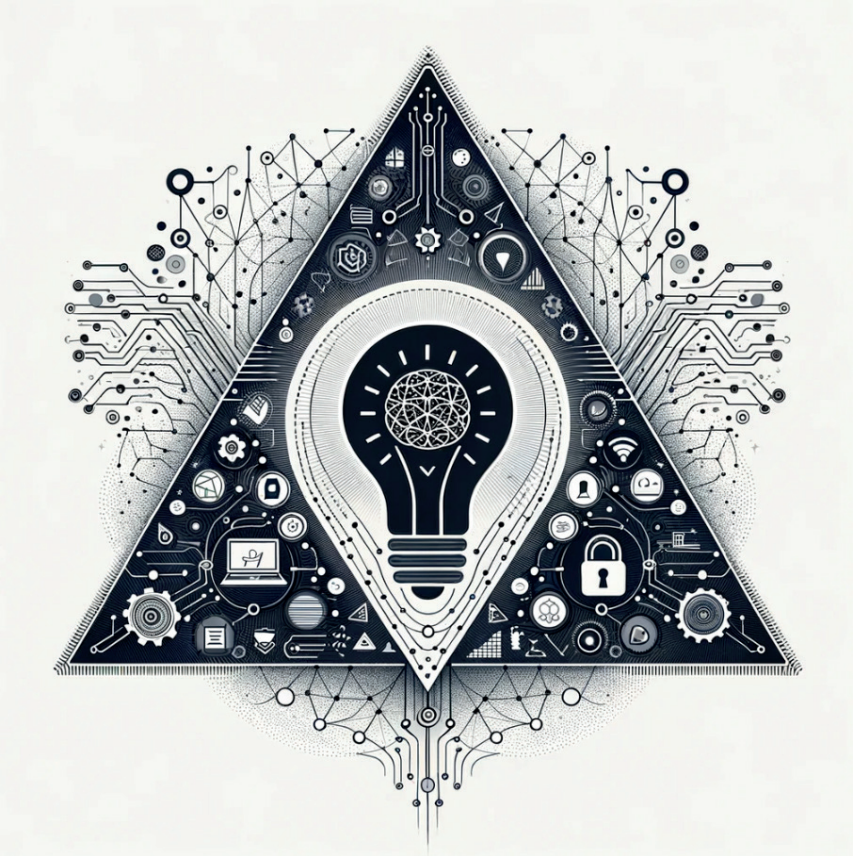
7 Computer Science Research Program 2023-2027

THIS

Towards High-quality and Intelligent Systems



Department of Computer Science, Open Universiteit

7.1 Overview of the department

7.1.1 History

The first research plan of the computer science department was created by the School of Computer Science at the OU in May 2011: *Software Technology Research Plan 2010-2015* [30]. This research program contained two research lines on software technology: *Software Technology for Teaching and Learning* and *Software Technology for Quality Improvement*.

In 2014 the School of Computer Science was integrated into the Faculty of Management, Science & Technology (MST), and renamed to the Department of Computer Science. The MST research committee created an interdisciplinary research program in December 2014: *Learning and Innovation in Resilient Systems: MST Research Program 2015-2020* [22].

In 2017, the MST research program on *Learning and Innovation in Resilient Systems* was assessed in a midterm review over the period 2014-2016. A self-evaluation of the research program was written [23]. The midterm review followed the SEP 2015-2021 'Protocol for Research Assessments in the Netherlands' (amended version, September 2016). The assessment committee considered three assessment criteria: research quality, relevance to society, and viability. Furthermore, three additional aspects were considered: PhD programs, research integrity, and diversity. The assessment outcome was very positive [1].

In 2020 the structure of the OU was reorganized. Since then, the Department of Computer Science became part of the Faculty of Science. A research plan was created for 2020-2025 [29] and, together with the department of Information Science, i.e. C&IS was assessed in 2022 in a National Computer Science Research Assessment following SEP 2021-2027 together with all Dutch universities except Delft (TUD) and Groningen (RUG). The same three assessment criteria were considered: research quality, relevance to society, and viability. Other additional aspects were considered: Open Science, PhD Policy and Training, Academic Culture, and Human Resources Policy.

The outcome was positive, summarizing that C&IS is recognized for emphasizing practical relevance, particularly through engaging a significant number of external PhD and master's students and collaborating with industry partners. The review committee applauds C&IS for undertaking the assessment process despite the department being relatively new to research. The committee notes that C&IS's willingness to embrace scrutiny and feedback is fundamental for its research growth and advancement.

Taking these observations into account, the current document is the update of the research of the Computer Science department for 2023-2027.

7.1.2 Research focus of the department

The Department of Computer Science at the Open University is researching a multi-faceted approach based on four research lines. Each line, while distinct, interconnects to form a comprehensive strategy (see Fig. 1).

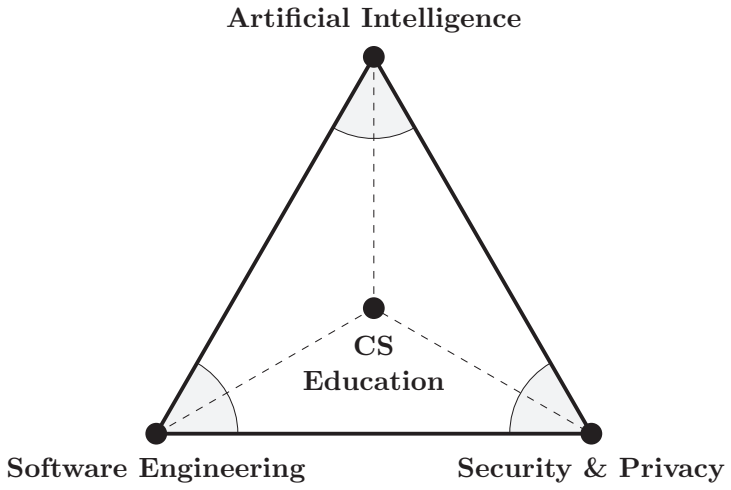


Figure 1: The four research lines at the Computer Science department

- *Software Engineering*, led by Prof. Dr. Tanja E.J. Vos, concentrates on enhancing software reliability through rigorous testing and formal verification methods.
- *Security & Privacy*, steered by Prof. Dr. Ir. Harald Vranken, aims to fortify software and computing systems against breaches, emphasizing measures for preventing, detecting and defending against security and privacy threats.
- *Artificial Intelligence*, under the guidance of Prof. Dr. Natasha Alechina, explores responsible, safe and efficient development of AI systems, in particular exploring how AI can further support and automate aspects of software engineering and security.
- *Computer Science Education*, led by Prof. Dr. Erik Barendsen, works on research supporting teaching and learning of computer science and digital skills, focusing in particular on programming, student-related factors and digital literacy.

Together, these lines form a robust and interdisciplinary network, striving not only to advance the practical aspects of software engineering but also to ensure the safe, secure, and enlightened development of future computing systems. They are shaping a new generation of computer science professionals well-prepared in the latest technologies and methodologies.

7.1.3 Size

Table 1 provides an overview of the amount of FTE involved in the research program.

Table 1: FTE in research lines

Research line	Staff	PhD/Postdoc	Total
Software Engineering	7,1	6,8	13,90
Security & Privacy	4,1	0	4,10
Artificial Intelligence	8,8	4	12,80
Computer science education	1,85	0	1,85

7.1.4 Embedding in the landscape in the Netherlands

7.1.4.1 ICT Research Platform Netherlands (IPN)

The Dutch university Computer Science community has a long tradition of national cooperation. The computer scientists are organized within the ICT Research Platform Netherlands (IPN)[1]. The members of IPN, are the general universities (including the OU) and the universities of the 4TU federation that have a substantial focus on computer science research and/or education, and the Centrum Wiskunde & Informatica (CWI).

Within the IPN, there are intensive collaborations on specific sub-areas in the 'Special Interest Groups (SIGs)':

- VERSEN[2] (Vereniging voor Software Engineering)

  Most members of our department are members of VERSEN. Moreover, the department head (Bastiaan Heeren) chairs the Workgroup on Education by the Dutch National Association for Software Engineering (VERSEN).

- SIG-Cyber Security (ACCSS)[3]
- SIG-Artificial Intelligence (SIGAI)[4]
- Data Science Platform Nederland (DSPN)[5]
- SIG- Future Computer Systems and Networking (FCSN)[6].

The annual national ICT.Open conference aims to bring together scientists from all ICT research disciplines and industries to meet, learn, and exchange ideas. It is jointly

[1] https://ict-research.nl/
[2] https://www.versen.nl
[3] https://accss.nl
[4] https://ict-research.nl/groups/special-interest-groups/sigai/
[5] http://www.datascienceplatform.org/index.shtml
[6] https://ict-research.nl/groups/special-interest-groups/fcsn/

organized by IPN, the HBO-ICT lecturers' network PRIO, and NWO. The department tries to assist these events yearly.

An exceptional initiative is the IPN EDI Working Group that strives to improve equity, diversity and inclusion in the Dutch ICT community. The group organizes concrete actions and events in this area and actively discusses EDI-related topics with policy makers, heads of departments and other relevant stakeholders. The working group holds plenary meetings four times per year. It includes representatives from all Dutch universities as well as CWI and NWO. In the year 2023, Prof. Dr. Tanja Vos from our department has taken on the role as chair of that working group.

7.1.4.2 Research schools

The three national computer science research schools (ASCI[7], IPA[8], SIKS[9]) collaborate closely. The PhDs of the department are member of the research schools that best fit their research topic.

7.1.4.3 Sectorplannen Informatica

In the Netherlands, plans for the sector of computer science, known as 'sectorplannen informatica'[10], have been developed as part of broader initiatives to enhance the quality of higher education and scientific research. These plans are part of a structural investment by the Dutch Government in scientific research across various science and engineering disciplines, including computer science, with a specific focus on strengthening the foundations of basic research.

In 2019, a concise computer science image was published for the Beta and Engineering domains. Within the Beta domain, the disciplines of Mathematics, Computer Science, Physics, and Chemistry were prioritized to participate in the first Beta and Engineering sector plan for 2018-2025 (referred to in the document as SectorPlan 1). The computer science department of the Open University did not participate in these sector plans.

The sector plan committee reported in an interim evaluation report at the end of May 2022 on the progress of this sector plan, which was then halfway through, to the Minister of Education, Culture, and Science (OCW). The conclusion was that due to the developments in society, science and innovation, and education, the joint efforts of Sector Plan 1 had not yet led to the desired stability and space for the sector. This was the reason for the Informatics Platform Netherlands (IPN), on behalf of the sector, to submit a second sector plan application to the Minister of OCW as part of the current round of national Sector Plans for 2023-2029. The computer science department of the Open University is participating in this sector plan for the first time.

[7] https://asci.tudelft.nl

[8] https://ipa.win.tue.nl/

[9] https://siks.nl

[10] https://www.nlsectorplannen.nl/bestanden-beta-ii

Within the 'Sectorplan Informatica' 2023, it is the intention to specifically strengthen research and education in the following key areas ('zwaartepunten'):

- Data Modeling and Analysis: This aims to address various fundamental questions concerning the modeling, organization, processing, storage, and analysis of big data. It also involves developing high-quality, flexible, and energy-efficient data-driven AI systems.
- Machine Learning: This involves the development of knowledge in learning computer systems that adapt their behavior based on data and have a wide range of industrial and societal applications. It covers learning patterns, large-scale energy-efficient machine learning systems, dealing with small datasets, and methodologies for fair, explainable, and reliable machine learning.
- Machine Reasoning and Interaction: The next step in the evolution of AI towards human-level intelligence is machine reasoning: the ability to apply learned knowledge to new situations. This enhances the quality of automated decision-making and enables AI systems to better support human intelligence.
- Algorithmics: This entails developing new methods to solve complex computational problems. New applications (energy networks, climate modeling, logistics, healthcare) and paradigms (quantum algorithms, programmable matter, distributed and streaming algorithms) present new fundamental challenges.
- Software: Software engineering is about systematically designing, developing, verifying, testing, and maintaining software. The four main challenges in this research area are: (1) how to guarantee the reliability of software systems, (2) how to improve the software development process, (3) how to enhance the flexibility, maintainability, and thus the sustainability of existing software systems, (4) how to train enough software engineers and scientists to meet societal needs.
- Security and Privacy: Within the sector plan, four fundamental research questions in cybersecurity and cryptography are central: (1) how can the (in)security of a system be (automatically) demonstrated, (2) how to design secure-by-design computer systems, (3) how to detect and repel attacks, and (4) how to ensure privacy and policy compliance in a rapidly developing digital world?
- Networked Computer and Embedded Systems: The scientific challenges here lie in the energy consumption, reliability, and sustainability of computer systems in cars, airplanes, medical equipment, smart buildings, robotics, etc. Their increasing complexity calls for the development of new, effective (software) solutions for the design, analysis, and optimization of these systems.

Table 2 shows how the research lines in the computer science research program at the OU correspond to the focus areas. Each line corresponds closely with one or two focus areas. The focus area on *Data modeling and analysis* is not covered by the program lines of the Department of Computer Science, but is covered by the research of the Department of Information Science and Business Processes at the OU. The focus area on *Software*

is covered by the lines *Software Engineering* and *Teaching & Learning*. The latter has a clear focus on education and also considers the application of software technology and programming languages, the development of a generic software framework, and the study of problem domains related to programming. The focus area *Networked computing and embedded systems* is not covered by the program lines, although some aspects are addressed, in particular, research on sustainability and energy analysis in the *Software Engineering* line.

Table 2: Mapping of program lines on focus areas

Focus area	**Program line**
Data modelling and analysis	-
Machine learning	Artificial Intelligence
Machine reasoning and interaction	Artificial Intelligence
Algorithmics	-
Software	Software Engineering, CS Education
Security and privacy	Security & Privacy
Networked computing and embedded systems	-

7.2 Research Line: Software Engineering

The research line in software engineering focuses on the quality of software. As the dependence on software in our society increases rapidly, its quality is crucial. However, this quality is not always evident. Unreliable and faulty systems cost money, can disrupt society, and, in critical sectors, may cause death. Our research contributes to high-quality software systems that underpin essential services in our society, from healthcare and finance to transportation and communication. By advancing software quality and addressing vulnerabilities, our research enhances the overall reliability of our digital infrastructure. We advance the quality of present-day systems, and also that of future software systems:

This gives rise to two sublines: *Software Testing and Analysis* described in Section 7.2.1 and *Programming Languages* described in Section 7.2.2.

7.2.1 Software Testing and Analysis

The "Software Testing and Analysis" research group is at the forefront of addressing a critical and contemporary challenge in the world of software engineering: the analysis and testing of *closed-source software*. In an era where legacy systems, proprietary third-party components, and intricate software architectures are ubiquitous, the unavailability of source code poses a formidable barrier to understanding, securing, and optimizing these systems. Our research encompasses a spectrum of vital topics, including scriptless testing, decompilation and reverse engineering, model learning, and verification of third-party libraries. These investigations share a common goal:

Slogan

Empower researchers and professionals to glean insights, detect vulnerabilities, and enhance the quality of real-life present software systems without requiring source code.

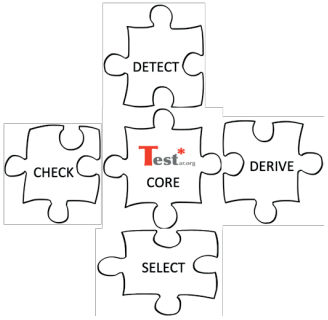
Our research group pioneers innovative, timely, and indispensable solutions to propel software engineering into the future, ensuring the integrity and resilience of software systems that underpin our digital world. In the next section, we describe each topic in more detail.

7.2.1.1 Scriptless test automation

End-to-end test automation at the Graphical User Interface (GUI) level is traditionally done by scripts that are designed to mechanize manual testing. However, conventional script-based tools are often rigid and fail to adapt to dynamic changes, necessitating substantial maintenance effort. In an era characterized by the accelerated development of applications, these limitations pose substantial challenges for testing teams, impeding their ability to stay aligned with the software developers.

Our research on *scriptless* end-to-end testing aims for a completely automated test approach. Instead of scripts, this approach is based on agents that implement various action selection mechanisms and test oracles. The underlying principle of this type of testing is very simple: generate test sequences of (state, action)-pairs by starting up the System Under Test (SUT) in its initial state and continuously selecting an action to bring the SUT to another state. The action selection characterizes the most basic problem of intelligent systems: what to do next.

We have implemented this approach in TESTAR. The tool implements the simple principle results in a core loop that continuously repeats:



- DETECT the GUI state
- DERIVE possible actions for the state
- SELECT an action to execute to go to the next state
- CHECK for failures

Research in this topic will concentrate on each of the 4 possible hooks that this simple approach can be extended and made more intelligent. This research will gradually shift the paradigm of end-to-end testing at the GUI level: from developing scripts to developing intelligent AI-enabled agents.

References and more

- Overview paper: Vos, T.E.J., Aho, P, Pastor, F, Rodriguez, O, Mulders, A. *TESTAR – scriptless testing through graphical user interface.* Software Testing Verification and Reliability 2021; 31:e1771. `https://doi.org/10.1002/stvr.1771`
- Open source repo: `https://github.com/TESTARtool/`

7.2.1.2 Decompilation

From legacy and abandonware to (smart) apps in virtual marketplaces, and from closed-source enterprise applications to embedded software; millions of binaries are being deployed and used daily, whose source code is inaccessible to stakeholders. Problematically, there is essentially nothing stakeholders can do to establish whether third-party binaries are indeed safe, except for just *believing* the original developers did a thorough job. But faith is brittle, and developers have shown—time and again—inadequacy in this area.

We aim to develop a novel holistic approach to decompilation and verification of binaries, by exploiting the synergy between new techniques for reverse-engineering and formal verification. To maximize impact, we are working on tools that target binaries originally written in C and C++. By placing particular emphasis on automation (the approach aims at minimal user interaction) and correctness (the produced result should be a

faithful semantic representation of the system under investigation), our vision is to allow *masses of stakeholders* to apply formal verification to *massively available binaries*.

References and more

- Freek Verbeek, Joshua Bockenek, Zhoulai Fu, Binoy Ravindran: *Formally verified lifting of C-compiled x86-64 binaries*. PLDI'22. `https://doi.org/10.1145/3519939.3523702`
- `https://ssrg-vt.github.io/FoxDec/`

7.2.1.3 Verification of standard libraries

Standard libraries are among the most commonly (re)used software components and a vital part of the ecosystem of any mainstream programming language. Yet, despite their importance, their safety and correctness are generally an open question.

Our initial work in this area has revealed uncovered decades-old bugs in the implementation of Java's sorting algorithm and list implementation. Our latest work has won an award from Google as part of a Google program that recognizes and rewards the often invisible and invaluable work of security researchers, such as finding and reporting programming bugs.

Research in this topic aims to develop new compositional techniques for deductive verification of (standard) libraries. Through the resulting provably safe and *massively used code*, our vision is to enable *masses of software engineers* to benefit from formal verification research, unbeknownst to them.

References and more

- First paper: Stijn de Gouw, Jurriaan Rot, Frank S. de Boer, Richard Bubel, and Reiner Hahnle. *Openjdk's java.utils.collection.sort() is broken: The good, the bad, and the worst case.* In CAV (1), volume 9206 of Lecture Notes in Computer Science, pages 273–289. Springer, 2015
- Google award winning paper: Hiep, H.D.A., Maathuis, O., Bian, J. et al. *Verifying OpenJDK's LinkedList using KeY (extended paper).* Int J Softw Tools Technol Transfer 24, 783–802 (2022). `https://doi.org/10.1007/s10009-022-00679-7`

7.2.1.4 Model learning

Model learning marks a major step forward to make formal verification methods more accessible to a wider range of researchers. Unlike traditional approaches where models are often handcrafted and may not accurately reflect the implemented system, model learning automates the creation of models that are consistent with the actual observable behavior of the system. These techniques are designed to facilitate researchers and

engineers in applying formal methods directly to actual systems without the laborious process of manual modeling. This is particularly relevant in our work on the analysis of attack surfaces within the Security & Privacy research line. Although this technique has been successful in small applications, it still requires expert knowledge about the system and it does not scale to larger systems.

The research in this topic, aims to address the above-mentioned limitations by improving and generalizing model learning techniques. The methodology we employ focuses on leveraging the modularity of the system under learning. By acknowledging and utilizing the modular design of systems (e.g., using communicating components), we aim to produce more scalable and accurate models. These models not only reflect the system more accurately but also align with the testing theories of finite state machines, enhancing the overall effectiveness of formal verification methods.

References and more

- Joshua Moerman. *Learning product automata*. In ICGI, volume 93 of Proceedings of Machine Learning Research. PMLR, 2018. `http://proceedings.mlr.press/v93/moerman19a.html`.
- Satuio: Software tool for generating adaptive distinguishing sequences and unique input/output sequences for finite state machines, 2022. `https://github.com/Jaxan/satuio`

7.2.1.5 Formal methods in biology

In some contexts, an accurate correlation between model and reality is extremely difficult or even impossible to achieve. This occurs for example when trying to unravel the communication networks that can be found inside living beings. As biological networks were not engineered, we cannot refer to any existing description or technical paper to ensure that a "biological communication protocol" is accurately and completely modeled. Still, also in these cases we want to make formal verification methods applicable and accessible to domain experts.

The objective behind the research in this topic is to allow biologists to apply the concepts of (theoretical, formal) modeling in their daily research without the need for a rigorous mathematical training. Thanks to a structured modeling approach, biologists are able to more clearly organize their data and theories about specific biological networks, and use the resulting models as a guide in discussions. In addition, the formal foundations of the models allow for the application of so-called *in silico* experiments based on simulations and model checking. This way, biologists can more easily test hypotheses and understand emergent behavior in complex models.

ANIMO (Analysis of Networks with Interactive MOdeling), one of the tools resulting from this research, makes the analysis power of Timed Automata available to biology researchers and students without the need for a training in formal methods.

References and more

- S. Khurana, S. Schivo et al. *An ECHO of Cartilage: In Silico Prediction of Combinatorial Treatments to Switch Between Transient and Permanent Cartilage Phenotypes With Ex Vivo Validation.* Frontiers in bioengineering and biotechnology, 2021. https://doi.org/10.3389/fbioe.2021.732917
- S. Schivo, S. Khurana et al. *ECHO, the executable CHOndrocyte: A computational model to study articular chondrocytes in health and disease.* Cellular Signaling, 2020. https://doi.org/10.1016/j.cellsig.2019.109471
- S. Schivo et al. *Computational modeling of complex protein activity networks.* Phosphorylation, 2017. https://doi.org/10.5772/intechopen.69804
- S. Schivo et al. *Modelling with ANIMO: between fuzzy logic and differential equations.* BMC Systems Biology, 2016. https://doi.org/10.1186/s12918-016-0286-z
- ANIMO - Analysis of Networks with Interactive MOdeling
  https://www.utwente.nl/en/tnw/dbe/research/post_lab/animo/
  https://fmt.ewi.utwente.nl/tools/animo/

7.2.2 Programming Languages

Where the subline on testing takes current software systems as a starting point to improve software quality, the subline on programming languages starts with the ideal future software program in mind, and creates tools to develop such systems. Such an ideal program is fault-tolerant, error-free, robust, and scalable. The common goal is:

Slogan

Creating tools, novel programming language features, or design and implement whole new programming languages, to support developers write their software.

The solutions that are proposed in this subline share a common trait in that the tools will enable the formal, and hopefully automated, proofs of program correctness. In the next sections, we will describe each topic in more detail.

7.2.2.1 Advanced Programming Languages

Software applications pop up in many different contexts. Writing a program for a specific context in a general purpose programming language tends to be a hard task. Developers need to take many recurring programming activities into account when creating such applications.

Our goal is to ease this development process while taking program correctness into account. Instead of creating an one-size-fits-all programming language, we take general

techniques from programming language design and create languages that are specially tailored to their domain.

We thoroughly define a language's *syntax and semantics* using techniques discussed below in Section 7.2.2.3. We need to add enough language features to ease the creation of new software. However, by adding advanced *type and effect systems* on top of our language, we restricting it in such a way that we can guarantee proper behavior of written programs. We can give mathematical proofs on *program safety*, and verify program properties using *symbolic execution*. These properties can be mechanized in *proof assistants*.

An example of our approach is a formal specification and programming language for task-oriented systems. The resulting language TopHat aids in faithfully and understandably modeling collaboration of people in the real world. It does so while taking away recurring programming activities for distributed and fault-tolerant applications with persistent data and interactive user interfaces.

Next steps are to apply similar techniques to design and create a programming language to write kernel extensions for the Linux operating system. These programs need to be correct by compilation, provably terminating, and provably run in a limited amount of memory space.

References and more

- First paper: Tim Steenvoorden, Nico Naus, and Markus Klinik. TopHat: A formal foundation for task-oriented programming. In *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 17:1–17:13, 2019.
- Idris proofs: `https://github.com/timjs/tophat-proofs`

7.2.2.2 Program generation

Problems with large IT projects persist in today's world despite technological advancements. We believe that formal methods can automate software design, development, and engineering, offering solutions to these persistent issues. Our focus is on languages that can specify business problems in such a way that they directly facilitate the generation of corresponding information systems. Our interest in such languages is motivated by practice, so collaboration with non-academic partners is imperative.

The research objective of this topic is to develop a comprehensive theory of information systems centered around the principle that business semantics alone should form the basis of system specifications. These specifications would then be used to automatically generate the system, aiming to reduce software errors and enhance project success rates.

Our methodology involves advancing the existing Ampersand framework, a platform already capable of generating information systems. We are working on enhancing this

framework with additional tooling for challenges such as data migration under evolving schemas and enabling incremental changes for more frequent releases.

Key to our implementation strategy is the development of systems that are composable, scalable, and cloud-native. We aim to introduce an associative, commutative, and idempotent union operator for system composition. Furthermore, we intend to validate our theory and tools through case studies in practical environments, emphasizing collaboration with non-academic partners to ensure real-world applicability.

References and more

- Stef Joosten, Relation Algebra as programming language using the Ampersand compiler, Journal of Logical and Algebraic Methods in Programming, Volume 100, 2018, Pages 113-129, ISSN 2352-2208, https://doi.org/10.1016/j.jlamp.2018.04.002.
- The Ampersand project: `https://ampersandtarski.github.io/`
- The tool: `https://rap.cs.ou.nl`
- The course: `https://www.ou.nl/-/IM0403_Rule-Based-Design`
- A blog `https://sjcjoosten.nl/1-research/information-systems/`

7.2.2.3 Metatheory

This topic delves into the mathematical underpinnings or metatheory of programming, a crucial aspect of understanding and advancing programming languages. It encompasses the study of the fundamental properties and theories that govern programming languages.

Semantics There are many different ways to formally define the semantics of a programming language. One approach that has proved to be quite effective is *operational semantics*, which describes the program behavior in terms of an abstract machine. Each flavor of semantics has its advantages, and studying those can yield insights about how to formalize existing languages, or develop new ones.

Program equivalence When the semantics of a programming language has been formalized, we can compare and analyze programs. We are particularly interested in *program equivalence*, i.e., whether or not two programs exhibit the same behavior. The ability to prove or automatically verify this is important, for instance, to check whether an optimized version of a program still calculates the same result.

Axiomatisation When proving facts about programs, we can start with several facts or *axioms* that we accept as true, and reason from there. When a set of axioms is sufficient to prove all semantically valid properties, it is called *complete*. Completeness is very

powerful because it means that no further primitive properties need to be considered when writing a proof. Nevertheless, establishing completeness for new systems remains hard, and so we study techniques that can help accelerate this process.

References and more

- Schmid, Todd, Tobias Kappé and Alexandra Silva. "A Complete Inference System for Skip-free Guarded Kleene Algebra with Tests." European Symposium on Programming (2023). https://doi.org/10.1007/978-3-031-30044-8_12
- Kappé, Tobias. "Completeness and the Finite Model Property for Kleene Algebra, Reconsidered." International Conference on Relational and Algebraic Methods in Computer Science (2023). https://doi.org/10.1007/978-3-031-28083-2_10
- Mohan, Anshuman, Yunhe Liu, Nate Foster, Tobias Kappé, and Dexter Kozen. "Formal abstractions for packet scheduling." Proceedings of the ACM on Programming Languages 7, no. OOPSLA2 (2023): 1338-1362. https://doi.org/10.1145/3622845

7.2.2.4 Code generation

Today, Large Language Models (LLMs) are mostly used as *databases*. Users ask questions, which the model answers based on their compressed world knowledge encoded in the model's weights. One popular application of LLMs as databases is to generate code, ranging from contemporary programming languages such as Java, Python, etc, to database queries in various dialects of SQL or other query languages. With the advent of multi-modal models, it is even possible to ask questions by drawing a picture of a UI and have the model generate all the necessary HTML, CSS, and JavaScript to create a full application.

However LLMs are increasingly supplied with tools that they can invoke to access real-time data such as Web search, or to perform actions such as looking up flights, order food, or in general run arbitrary code. By combining tool use with the code generation abilities, LLMs can be viewed as powerful *neural computers*.

The aim of this research topic is to create a powerful natural language-based programming language for these AI-powered neural computers, by adding functionalities for users to, for example, name and parameterize prompts. Programming AIs faces all the same Software Engineering challenges as programming traditional binary computers, including quality, correctness, and robustness. A key aspect of our implementation is the incorporation of *proof carrying code* where the model generates both code as well as a proof that the generated code is safe, correct, and efficient.

References and more

- J. Bader, S. Seohyun Kim, F. Sifei Luan, S. Chandra and E. Meijer, "AI in Software Engineering at Facebook," in IEEE Software, vol. 38, no. 4, pp. 52-61, July-Aug. 2021, doi: 10.1109/MS.2021.3061664.
- Max Tegmark and Steve Omohundro, "Provably safe systems: the only path to controllable AGI", arXiv:2309.01933, 2023

7.3 Research line: Security & Privacy

We live in a digital society, which brings huge advances and benefits to all aspects of our daily lives. This however makes us dependent on Information and Communication Technologies (ICT), and hence vulnerable to threats. Managing security and privacy risks is essential to safeguard our ICT infrastructures and data from malicious actors, which range from script kiddies to organized cybercriminals and even state actors. The impact of attacks affects individual citizens (eg., in banking fraud), companies and institutions (eg., in ransomware attacks), and even our national safety (eg., attacks on control systems for critical infrastructures such as the electricity grid or water barriers).

The research challenges on security and privacy in the Netherlands have been formulated in the National Cyber Security Research Agenda (NCSRA) in five pillars:

1. Design: applying security-by-design to prevent security problems of systems and services before they are deployed
2. Defence: taking measures to protect deployed systems through identifying assets, preventing and detecting attacks, responding to incidents, and mitigating the impact of attacks and recovery
3. Attack: understanding the attack surface of systems
4. governance: addressing (conformance with) policies and national and international regulatory frameworks
5. Privacy: addressing how to protect sensitive and personal data.

Our research mainly addresses technical aspects of security and privacy in the pillars design, defence, attack, and privacy. The governance pillar is addressed mainly by the Department of Information Science, with whom we closely cooperate.

The research line Security & Privacy focuses on three sub-lines:

1. analysis of attack surfaces: What are vulnerabilities that cause security and privacy threats, and when and why do they occur?
2. mitigation of security and privacy threats: How to prevent or defend against the identified threats?

3. the human factor, education and ethics: This sub-line addresses non-technical aspects. The human factor is the Achilles-heel of security, and we address this by studying human behaviour and useability aspects of security and privacy, supporting education on security and privacy, and considering ethical aspects.

7.3.1 Analysis of attack surfaces

We analyse the attack surfaces of ICT systems to better understand threats on security and privacy. These attack surfaces comprise all potential entry points and vulnerabilities that can be exploited by attackers to compromise systems, involving the hardware and software that build these systems as well as the people that use them. We consider this both at the level of the system design and architecture, as well as at the system implementation level.

Slogan

Analysing the attack surface of ICT systems to identify threats on security and privacy.

Our research addresses the attack surfaces of systems, software and computer networks in general, and the attack surfaces of AI systems, distributed systems, and the World-Wide Web as specific instances.

Systems We develop formal models of real-life systems and describe how they can be disrupted by malicious actors. Using an abstract model as a reference for a more complex system lets us understand its most important aspects. The use of formal models allows us to apply formal verification methods such as model checking, to obtain useful insights into how system weaknesses can be exploited and how the security of systems can be strengthened. We develop software tools for this purpose using advanced software engineering techniques, making the power of formal methods available to security experts without the need for additional formal training. An example is the modelling and verification of attack trees. (This research has close relations with topics on formal verification of standard libraries and model learning in the Software Engineering research line.)

References and more

- R. Kumar, S. Schivo, E. Ruijters, B. Yildiz, D. Huistra, J. Brandt, A. Rensink and M. Stoelinga, "Effective Analysis of Attack Trees: A Model-Driven Approach", in Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science 10802, pp. 56-73, 2018, doi:10.1007/978-3-319-89363-1_4
- N. Dong, H. Jonker and J. Pang, "Formal modelling and analysis of receipt-free auction protocols in applied pi", Computers & Security 65, 2017, pp. 405-432, doi:10.1016/j.cose.2016.09.002

Software We analyse the source code of software to identify vulnerabilities. We apply state-of-the-art AI methods to improve static code analysis. The main challenge is how to transform source code into a representation that can be input to an AI model. This transformation typically considers combining abstract syntax tree, control flow, and data flow representations into a graph model, and subsequently transforming the graph model into a numeric representation. This transformation should maintain sufficient information about the syntaxis and semantics related to vulnerabilities.

References and more

- J. Kronjee, A. Hommersom and H. Vranken, "Discovering software vulnerabilities using data-flow analysis and machine learning", in Proceedings of the International Conference on Availability, Reliability and Security, 2018.

Computer networks We analyse computer network traffic to detect anomalies and traces of malicious activity (such as botnets). We apply state-of-the-art AI methods to train classifiers that can distinguish anomalies and malicious traffic from regular network traffic. The challenge is to analyse network traffic in real time. This requires small models that operate at high speed and that also can be retrained quickly. Using small models also facilities explainability of the model and reducing false positive detections.

References and more

- D. Willems, K. Kohls, B. van der Kamp and H. Vranken, "Data Exfiltration Detection on Network Metadata with Autoencoders", in Electronics 2023, 12(12), 2584, doi:10.3390/electronics12122584.

AI systems We analyse security and privacy aspects that arise in the development and deployment of AI systems. We consider the security and privacy of both data and AI models. Data poisoning is an example of a data security threat, in which attackers attempt to bias or deceive AI systems by manipulating training data or injecting malicious training data. Protecting data privacy is needed to protect the privacy of individuals and sensitive data, and avoiding inadvertent disclosure or data breaches that can lead

to privacy violations and legal consequences. We address robustness of AI models by making them resistant to adversarial attacks during inference, in which attackers try to manipulate inputs to mislead the AI system. (This research has close relations with robust, safe and trustworthy AI in the AI research line.)

References and more

- M. Alishahi, V. Moghtadaiee and H. Navidan, "Add noise to remove noise: Local differential privacy for feature selection", in Computers & Security 123, 2022, 102934, doi:/10.1016/j.cose.2022.102934

Distributed systems We analyse security and privacy aspects of distributed systems. We focus on distributed systems in which consensus algorithms are used to agree on a global state. Prime examples are decentralized Web3 applications that are built on blockchain, and decentralized systems such as cryptocurrencies and distributed storage systems. Consensus mechanisms such as proof-of-work and proof-of-stake provide security by validating and authenticating transactions, that are next stored on a blockchain. We study the electricity consumption and environmental footprint due to consensus algorithms, as well as incentive mechanisms to reduce this footprint. We also study other aspects, such as the security of smart contracts, and tracing transaction flows on blockchains.

References and more

- A.R. Sai and H. Vranken, "Promoting rigor in blockchain energy and environmental footprint research: A systematic literature review", in Blockchain: Research and Applications 2024, 5(1), 100169, Elsevier, doi:10.1016/j.bcra.2023.100169

World-Wide Web We analyse the security and privacy of the world-wide web. We focus on digital fingerprinting, which is used to uniquely identify and track devices that connect to a website or online service. Digital fingerprinting considers various device characteristics and configurations, such as IP address, browser type and version, screen resolution, time zone, installed fonts, and language settings. The combination of these attributes creates a unique identifier for the device. Digital fingerprinting offers a means to track or identify visitors across websites and services, enabling the creation of user profiles. Although digital fingerprinting can improve security by identifying suspicious or potentially fraudulent activities, it also raises privacy concerns as it can be used to track users without their consent, even when they disallow cookies or use private browsing modes. We study fingerprinting techniques and countermeasures, their spread and impact, as well as ethical and regulatory aspects. For example, we investigate how the reliability of web measurements through web scrapers is affected by browser fingerprinting that detects web scrapers. To improve upon this, we design and implement countermeasures that enable large-scale web measurements whose results are not marred

by scraper detection.

References and more

- S. Calzavara, H. Jonker, B. Krumnow and A. Rabitti, "Measuring Web Session Security at Scale", in Computers & Security 111, 2021, 102472, doi:10.1016/j.cose.2021.102472

7.3.2 Mitigation of security and privacy threats

We apply methods and techniques in computer science to mitigate security and privacy threats. We provide security-by-design and privacy-by-design to prevent threats, and we provide methods and techniques to defend against threats. Our research focuses on the application of AI and cryptography to address threats, and on mitigating threats for AI systems.

Slogan

Providing security-by-design and privacy-by-design to mitigate threats on security and privacy.

Application of AI for security and privacy We apply AI methods and techniques to improve security and privacy in several different ways. AI has been applied in the last decades by both academic researchers and industry practitioners to address security challenges and problems related to security and privacy. This ranged from proactively protecting and defending systems and services to responding to security incidents after their occurrence, as well as analysing their impact and decision-making support. On the technical level, we study how AI can help in automated repair and patching of software vulnerabilities, and analysis of anomalies in computer network traffic. On the operational and management levels, we study the application of hybrid AI techniques, which combine human expertise with machine learning and deep learning, for improving security, such as structuring information on threats and solutions, metadata analysis, impact assessment and decision-making support for military Cyber/Information Operations, and assessing and strengthening user's security behaviour in relation to different types of security incidents.

References and more

- H. Vranken and H. Alizadeh, "Detection of DGA-Generated Domain Names with TF-IDF", in Electronics 2022, 11(3), 414, doi:10.3390/electronics11030414
- W. de Kraker, H. Vranken and A. Hommersom, "GLICE: Combining Graph Neural Networks and Program Slicing to Improve Software Vulnerability Detection", in Proceedings IEEE European Symposium on Security and Privacy, 2023, doi:10.1109/EuroSPW59978.2023.00009
- A. Chockalingam and C. Maathuis, "Assessing Cascading Effects of Cyber-Attacks in Interconnected Critical Infrastructures", in Proceedings European Safety and Reliability Conference, 2022, doi:10.3850/978-981-18-5183-4_S23-04-521-cd

Mitigating threats of AI systems Ensuring the robustness of AI models involves a combination of techniques, including data preprocessing, model design, regularization, training with adversarial examples, and monitoring. Furthermore, trained AI models may be considered as intellectual property, and hence AI models themselves should be secured against stealing, while also the usage of AI models by unauthorized parties may lead to information leaks. We study scenarios for differential privacy in which training data is distributed among multiple entities without entities sharing their original data. To achieve this, we explore the usage of federated learning, in which each participant locally trains an AI model using its own data and sends model updates to a central server that aggregates the model updates from all participants. This ensures that raw data stays locally with each participant, avoids the need for transferring large volumes of data to the central server, and is robust to dropout or failures of participants. Additionally, we study how to incorporate fairness considerations into AI systems.

References and more

- M. Sheikhalishahi and F. Martinelli, "Privacy preserving clustering over horizontal and vertical partitioned data", in IEEE Symposium on Computers and Communications, 2017, doi: 10.1109/ISCC.2017.8024694

Cryptography We study real-world application of cryptography to improve security and privacy. We focus on the development and implementation of privacy-enhancing technologies (PETS) to protect information related to a person's identity. A first research topic is the usage of attributes to generalise identities in attribute-based authentication, signature, and access control. We contribute to the implementation of attribute-based credentials in the IRMA and Yivi applications. A second research topic is the usage of polymorphic pseudonymisation to hide identities in the context of big data. A third research topic is practical cryptographic protection of medical data (PEP), and attribute-based encryption schemes to protect data stored at cloud services.

References and more

- G. Alpár, F. van den Broek, B. Hampiholi, B. Jacobs, W. Lueks and S. Ringers, "IRMA : practical , decentralized and privacy-friendly identity management using smartphones", in Proceedings Workshop on Hot Topics in Privacy Enhancing Technologies, 2017
- F. van den Broek, B. Hampiholi and B. Jacobs, "Securely Derived Identity Credentials on Smart Phones via Self-enrolment", in Security and Trust Management, Lecture Notes in Computer Science 9871, 2016, Springer, doi:10.1007/978-3-319-46598-2_8

7.3.3 Human factor, ethics, and education

The human factor is considered to be the Achilles-heel of security. We address this by studying human behaviour and useability aspects of security and privacy, by supporting education on security and privacy, and by considering ethical aspects.

Slogan

Addressing the human factor and ethics in security and privacy by considering how humans interact with ICT systems, and how to educate students and users.

Attention for the human factor and ethics plays an important role in all our research activities in the analysis- and mitigation-sublines. In addition, we carry out small case studies on topics related to the human factor and ethics, such as security of digital exams, privacy in digital forensics, compliancy with security and privacy regulations, and fraud in scientific publishing.

In education, we recognise that beyond teaching theoretical concepts and principles of security and privacy, equipping students with hands-on experience in labs enhances their understanding and deepens their knowledge. We provide this by researching, developing, and applying virtual labs, in which students can practice both defensive and offensive techniques in a realistic yet simulated environment. We conduct research not only about the technical infrastructure of distributed virtual labs to enable groups of remote students working together, but also about the way such labs can be applied for security education in distance teaching.

We explore the challenges and possibilities offered by generative AI. We build tools for comprehensive and active defence solutions that combine the generation and detection of mechanisms like disinformation for security awareness and learning purposes. In addition, for educational purposes, we are building educational games for young students and non-STEM experts for raising security awareness and learning support in relation to social media manipulation mechanisms like disinformation and deep fakes.

References and more

- J. Haag, H. Vranken and M. van Eekelen, "A Virtual Classroom for Cybersecurity Education", in Transactions on Edutainment XV, LNCS 11345 (2019): 173–208, Springer, doi:10.1007/978-3-662-59351-6_13
- C. Maathuis and S. Chockalingam, "Responsible Digital Security Behaviour: Definition and Assessment Model", in Proceedings European Conference on Cyber Warfare and Security, 2022, doi:10.34190/eccws.21.1.203
- C. Maathuis and S. Chockalingam, "Modelling Responsible Digital Security Behaviour for Countering Social Media Manipulation", in Proceedings European Conference on Social Media, 2023, doi:10.34190/ecsm.10.1.1079

7.4 Research line: Artificial Intelligence

Artificial intelligence (AI) is acquiring increasing importance in society and in business, and research into AI is expanding in the computer science department. National and European research agendas in AI strongly promote both technical advances in AI (such as deep learning) and research that ensures that AI technologies are beneficial to society: that AI systems are robust, safe, trustworthy, and are developed and applied ethically and responsibly. At the CS department, we are actively involved in the technical development of AI, in conducting AI research involving both humans and AI in creating new systems and solutions, and in ensuring AI conformance to safety requirements and ethical values.

This research line can be subdivided into ensuring trustworthiness of AI systems, and improving their effectiveness in collaborating with humans. There are clearly interactions between the two research sub-lines.

7.4.1 Robust, safe and trustworthy artificial intelligence

Slogan

Develop techniques for safe, robust and trustworthy AI systems

Robustness and safety of artificial intelligence can be ensured by using verification techniques applied to already existing systems, or generating (synthesizing) systems that are guaranteed to be correct by construction. Transparent and explainable systems are more amenable to rigorous analysis and providing guarantees for the system's behavior. A significant amount of research in the AI group focuses on symbolic systems and explicit knowledge representation

References and more

- Natasha Alechina, Giuseppe De Giacomo, Brian Logan, and Giuseppe Perelli. Automatic synthesis of dynamic norms for multi-agent systems. In Gabriele Kern-Isberner, Gerhard Lakemeyer, and Thomas Meyer, editors, *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, pages 12–21. ijcai.org, 2022.
- Raphaela Butz, Renée Schulz, Arjen Hommersom, and Marko C. J. D. van Eekelen. Investigating the understandability of XAI methods for enhanced user experience: When bayesian network users became detectives. *Artif. Intell. Medicine*, 134:102438, 2022.
- Giso H. Dal, Alfons W. Laarman, Arjen Hommersom, and Peter J. F. Lucas. A compositional approach to probabilistic knowledge compilation. *Int. J. Approx. Reason.*, 138:38–66, 2021.
- Jesse Heyninck, Gabriele Kern-Isberner, Tjitze Rienstra, Kenneth Skiba, and Matthias Thimm. Revision, defeasible conditionals and non-monotonic inference for abstract dialectical frameworks. *Artif. Intell.*, 317:103876, 2023.

7.4.1.1 Integration of symbolic and subsymbolic approaches

Our strategy for the future is to develop techniques for achieving robust and safe AI that combine symbolic and subsymbolic approaches, reasoning, and learning. Examples of existing work in this direction explain the behavior of learned systems; declaratively specifying constraints on the outcome of learning; and using symbolic techniques for ensuring safety in reinforcement learning.

References and more

- Raphaela Butz, Arjen Hommersom, and Marko van Eekelen. Explaining the most probable explanation. In *International Conference on Scalable Uncertainty Management*, pages 50–63. Springer, 2018.
- Giovanni Varricchione, Natasha Alechina, Mehdi Dastani, Giuseppe De Giacomo, Brian Logan, and Giuseppe Perell. Pure past action masking. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Safe, Robust and Responsible AI (SRRAI) Track*, 2024.

In order to make techniques from knowledge representation amenable to integration with sub-symbolic techniques, it is also important to investigate how to make the processing of symbolic knowledge more efficient (e.g. by breaking down the knowledge base in modular parts and to give a principled account of how to combine symbolic knowledge with sub-symbolic knowledge (e.g. in the form of probabilities, fuzzy values or plausibilities. Specifically, the interaction between symbolic and sub-symbolic systems, such as large language models (LLMs), we can reduce the knowledge engineering bottleneck associated with symbolic systems, while at the same time enhancing the capabilities of

LLMs towards performing higher-order reasoning and planning.

References and more

- Jesse Heyninck, Gabriele Kern-Isberner, Thomas Andreas Meyer, Jonas Philipp Haldimann, and Christoph Beierle. Conditional syntax splitting for non-monotonic inference operators. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023*, pages 6416–6424. AAAI Press, 2023.
- Jesse Heyninck, Gabriele Kern-Isberner, Tjitze Rienstra, Kenneth Skiba, and Matthias Thimm. Revision, defeasible conditionals and non-monotonic inference for abstract dialectical frameworks. Artif. Intell., 317:103876, 2023.
- Vishal Pallagani, Bharath Muppasani, Biplav Srivastava, Francesca Rossi, Lior Horesh, Keerthiram Murugesan, Andrea Loreggia, Francesco Fabiano, Rony Joseph, Yathin Kethepalli. Plansformer tool: demonstrating generation of symbolic plans using transformers. In *IJCAI*, volume 2023, pages 7158–7162. International Joint Conferences on Artificial Intelligence, 2023.

7.4.1.2 Explainable AI

We aim to contribute to human-centered aspects of explainable-AI (XAI) methods, for example, by evaluating the understandability of explanations. This is for example very relevant in medicine, where there is an urgent need for better understanding the requirements of such XAI systems to obtain better user acceptability, actions taken based on the results from the system and overall impact on clinical practice.

References and more

- Raphaela Butz, Renée Schulz, Arjen Hommersom, and Marko C. J. D. van Eekelen. Investigating the understandability of XAI methods for enhanced user experience: When bayesian network users became detectives. *Artif. Intell. Medicine*, 134:102438, 2022.
- C. Combi, B. Amico, R. Bellazzi, A. Holzinger, J. H. Moore, M. Zitnik, and J. H. Holmes. A manifesto on explainability for artificial intelligence in medicine. *Artificial Intelligence in Medicine*, 133:102423, 2022.

7.4.1.3 Robust, private and safe AI

We develop robust models through Bayesian methods, that allows for including prior knowledge into the learning process. For example, such learning methods have been developed by our group in the context of continuous-time Bayesian networks, and we aim to further investigate this in dynamic systems, particularly by incorporating various

types of knowledge in dynamic treatment regimes. We are also addressing privacy aspects in machine learning and data analysis in general.

References and more

- Manxia Liu, Arjen Hommersom, Maarten van der Heijden, and Peter JF Lucas. Learning parameters of hybrid time bayesian networks. In *Conference on Probabilistic Graphical Models*, pages 287–298. PMLR, 2016.
- Milan Lopuhaä-Zwakenberg, Mina Alishahi, Jeroen Kivits, Jordi Klarenbeek, Gert-Jan van der Velde, and Nicola Zannone. Comparing classifiers' performance under differential privacy. In *SECRYPT*, pages 50–61, 2021.
- Mina Sheikhalishahi, Andrea Saracino, Fabio Martinelli, and Antonio La Marra. Privacy preserving data sharing and analysis for edge-based architectures. *International Journal of Information Security*, pages 1–23, 2022.

7.4.1.4 AI and Cybersecurity

AI is shaping the trajectory of progress in all societal domains by providing insights and decision-making support in a wide range of activities. A fundamental pillar of building robust, safe, and trustworthy AI systems is to assure their transparency in relation to data used, systems' behavior, and decisions made in a way that is explainable, interpretable, and human-centered to human needs, goals, and expectations. To this end, extensive research is conducted in the military domain for building safe, responsible, and trustworthy AI systems for conducting military operations in a way that accounts, respects, and protects civilian lives and infrastructure and provides relevant military decision-making support. We are also collaborating with the Security and Privacy research line on detection of software vulnerabilities using machine learning.

References and more

- Clara Maathuis. On explainable AI solutions for targeting in cyber military operations. In *International Conference on Cyber Warfare and Security*, volume 17, pages 166–175, 2022.
- Clara Maathuis. On the road to designing responsible AI systems in military cyber operations. In *European Conference on Cyber Warfare and Security*, volume 21, pages 170–177, 2022.
- Clara Maathuis. Human centered explainable AI framework for military cyber oper-ations. In *MILCOM 2023-2023 IEEE Military Communications Conference (MIL-COM)*, pages 260–267. IEEE, 2023.
- Clara Maathuis, Wolf Pieters, and Jan van den Berg. Decision support model for effects estimation and proportionality assessment for targeting in cyber operations. *Defence Technology*, 17(2):352–374, 2021.
- Wesley De Kraker, Harald Vranken, and Arjen Hommmersom. Glice: Combining graph neural networks and program slicing to improve software vulnerability detection. In 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pages 34–41. IEEE, 2023.

7.4.2 Effective Human-Centered AI

Open University places a strong emphasis on the concept of co-creation with stakeholders, including the context of human-AI collaboration in which pre-trained AI systems are leveraged to create new AI models for the sake of human knowledge advancement. Co-creation, in our context, also involves the active involvement of human stakeholders in defining research directions that revolve around methodologies for the practical application of AI in industry and society.

Slogan

Empower researchers and users to effectively co-create with AI systems

The co-creation aspect of our research line is built upon our extensive experience in applying AI within industrial environments, exemplified by partnerships with companies like DHL and APG through the experience of the Brightland Smart Services Campus AI hub as a community aiming at applied research with industry and society. This implies that, in addition to algorithmic development, our department is dedicated to developing methodologies for the practical application of AI technologies in the industry, by modeling systems that can create relevant insights and business value for industrial stakeholders. Human-centered AI will therefore be the preferred context of our efforts, to ensure that, following the European tradition concerning human values, AI systems are created to foster human rights, and to the benefit of human society as a whole, as opposed to a race towards the most advanced AI system, the focus will be on the

development of the fairest and most inclusive possible type of AI.

In the pursuit of such objectives, our research group plans to focus on topics such as deep learning natural language processing, reinforcement learning and neurosymbolic reasoning.

References and more

- Ben Shneiderman. Human-centered AI. Oxford University Press, 2022.
- Bea Waelbers, Stefano Bromuri, and Alexander P Henkel. Comparing neural networks for speech emotion recognition in customer service interactions. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.
- Thomas van Dongen, Gideon Maillette de Buy Wenniger, and Lambert Schomaker. Schubert: Scholarly document chunks with bert-encoding boost citation count prediction. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 148–157, 2020.
- Giovanni Varricchione, Natasha Alechina, Mehdi Dastani, and Brian Logan. Synthesising reward machines for cooperative multi-agent reinforcement learning. In *European Conference on Multi-Agent Systems*, pages 328–344. Springer, 2023.
- Jesse Heyninck, Gabriele Kern-Isberner, Tjitze Rienstra, Kenneth Skiba, and Matthias Thimm. Revision, defeasible conditionals and non-monotonic inference for abstract dialectical frameworks. *Artif. Intell.*, 317:103876, 2023.

7.4.2.1 AI and Digital Twins

Other examples of building such human-AI collaboration systems are digital twins, i.e., digital representations of physical objects, systems, and processes involved when conducting military operations in a safe and responsible way. Another example is building AI solutions that assess and enhance responsibility of digital security behavior in relation to social media manipulation mechanisms like disinformation and misinformation. Countering disinformation is another important line of research that is conducted at OU by means of both AI and gaming technologies. To this end, Maathuis et al. build a hybrid deep learning system for generating and detecting disinformation in relation to ongoing societal crises, such as pandemic and conflict.

References and more

- Clara Maathuis. An outlook of digital twins in offensive military cyber operations. In *European Conference on the Impact of Artificial Intelligence and Robotics*, volume 4, pages 45–53, 2022.
- Clara Maathuis and Sabarathinam Chockalingam. Modelling responsible digital security behaviour for countering social media manipulation. In *ECSM 2023 10th European Conference on Social Media*. Academic Conferences and publishing limited, 2023.
- Clara Maathuis, Iddo Kerkhof, Rik Godschalk, and Harrie Passier. Design lessons from building deep learning disinformation generation and detection solutions. In *European Conference on Cyber Warfare and Security*, volume 22, pages 285–293, 2023.

7.4.2.2 Decision making in Industry 4.0

Conventional planning models are strong considering the complexity of the problem they are tackle. However, these models have a common aspect of using strong assumptions and statistical parameters, since they lack tools to cope with intrinsic uncertainties of the processes. Fortunately, this does not become a bothering issue in long time horizons. However when it comes to plan in relatively shorter time horizon, as in case of make-to-order or just-in-time type production or service conventions, then relaxing the strong assumptions by learning the complex processing system parameters turns out to be the only way to generate realistic outputs. Here, AI comes as a crucial discipline to improve conventional planning methods in several ways. Sometimes AI tools work with conventional planning models as a component in the solution framework with both supportive and active roles. Predicting case-specific parameters is an example of former role and making real-time decisions is of latter. Some AI tools may also be embedded into planning models as one way of enhancing the capability of these models for more complex decisions.

References and more

- Murat Firat, Julie De Meyere, Tugce Martagan, and Laura Genga. Optimizing the workload of production units of a make-to-order manufacturing system. *Computers & Operations Research*, 138:105530, 2022.
- Murat Firat, Guillaume Crognier, Adriana F Gabor, Cor AJ Hurkens, and Yingqian Zhang. Column generation based heuristic for learning classification trees. *Computers & Operations Research*, 116:104866, 2020.
- D. Wely. Integrating machine learning in the project scheduling of a multi-skilled workforce. Master's thesis, Data Science in Business and Entrepreneurship, Jheronimus Academy of Data Science, 2022.

7.4.2.3 Interactivity

Interactivity plays a pivotal role in the development and deployment of machine learning models, particularly in the context of human-in-the-loop systems. There are two common avenues for incorporating interactivity into the model, interactivity during training and interactivity during inference.

Interactivity During Training One avenue for incorporating interactivity is during the training phase, exemplified by approaches like active learning or reinforcement learning with human feedback (RLHF). In active learning, the model actively selects the most informative instances to ask a human to label. This process optimizes the model's learning efficiency by focusing on challenging examples, thereby improving its performance with fewer labeled examples. Human input becomes an integral part of the learning loop, steering the model toward a better understanding of complex patterns and nuances in the data.

Another example, popularized by large language models, is RLHF. In this setting, humans provide additional feedback to guide the learning and outcomes of the model. This feedback can include reward shaping, preference information, or explicit corrections to the model's actions. It is powerful paradigm that bridges the gap between the capabilities of machine learning models and the nuanced, contextual understanding that humans bring to complex decision-making scenarios. It allows the model to adapt to human preferences as well as domain-specific knowledge, making it well-suited for applications where human subjectivity or adherence to certain constraints is crucial.

References and more

- Gabrielle Kaili-May Liu. Perspectives on the social impacts of reinforcement learning with human feedback, arXiv:2303.02891, 2023.

Interactivity During Inference Another avenue of interactivity arises during inference time, where models are endowed with control mechanisms that enable users to influence or steer the outputs. This interactive paradigm empowers users to guide the model's decision-making, fostering a collaborative and adaptive system that aligns more closely with user preferences and requirements. These control mechanisms can take on various forms of user inputs, such as text prompts, point clicks, constraint specification, and preference indication. This transforms machine learning models from static tools to adaptive systems that respond to user needs in real-time. Users become active participants in the decision-making process, leveraging the model's capabilities while retaining control over the final outcomes. This collaborative approach ensures that AI systems are not black boxes but rather tools that users can shape and trust.

Together, these dual facets of interactivity reinforce the symbiotic relationship between human intelligence and machine learning models. By involving human input during both

training and testing, we move beyond traditional one-size-fits-all models to more flexible, adaptive, and user-centric AI systems that better serve the diverse and evolving needs of users in various domains. We have been active in this area for example for interactive object counting, image retargeting, and style transfer.

References and more

- Tadhg McCarthy, John Jethro Virtusio, Jose Jaena Mari Ople, Daniel Stanley Tan, Divina Amalin, and Kai-Lung Hua. Macnet: Mask augmented counting network for class-agnostic counting. *Pattern Recognition Letters*, 169:75–80, 2023.
- Adrienne Francesca O Soliven, John Jethro Virtusio, Jose Jaena Mari Ople, Daniel Stanley Tan, Divina Amalin, and Kai-Lung Hua. Conconet: Class-agnostic counting with positive and negative exemplars. *Pattern Recognition Letters*, 171:148–154, 2023.
- Jilyan Bianca Dy, John Jethro Virtusio, Daniel Stanley Tan, Yong-Xiang Lin, Joel Ilao, Yung-Yao Chen, and Kai-Lung Hua. Mcgan: mask controlled generative adversarial network for image retargeting. *Neural Computing and Applications*, 35(14):10497–10509, 2023.
- John Jethro Virtusio, Jose Jaena Mari Ople, Daniel Stanley Tan, Muhammad Tanveer, Neeraj Kumar, and Kai-Lung Hua. Neural style palette: A multimodal and interactive style transfer from a single style image. *IEEE Transactions on Multimedia*, 23:2245–2258, 2021.
- John Jethro Virtusio, Daniel Stanley Tan, Wen-Huang Cheng, Mohammad Tanveer, and Kai-Lung Hua. Enabling artistic control over pattern density and stroke strength. *IEEE Transactions on Multimedia*, 23:2273–2285, 2020.

7.5 Research line: Computer Science Education

The Computer Science Education research field focuses on teaching and learning of CS-related topics, from the perspective of both teachers and learners. The field investigates education on a wide range of computing topics, including (but not limited to) programming education, design, discrete mathematics (e.g. propositional logic), algorithms, and software engineering. Traditionally, much attention is focused on introductory programming courses, since writing code is a notoriously hard skill to master, which often makes these courses a major stumbling block for students. Compared to other educational research fields such as mathematics education and science education, CS education is relatively young.

The international CS Education research community has organized itself into a Special Interest Group on Computer Science Education (SIGCSE) hosted by ACM, with associated conferences that are held annually, a newsletter, and a list of topics. Table 3 lists the four topic areas that are defined, which are used by the OU researchers for

Topic areas	*Description*	*5.1*	*5.2*	*5.3*
Computing Topics	These topics relate to different content areas within computing education.	×		
Broadening Participation in Computing	These topics relate to efforts to make CS education a more equitable space for all and improve diversity and inclusion in computer science.		×	
Education and Experience	These topics relate to different pedagogical concerns in the teaching and learning of computing.	×	×	×
Curriculum	These topics address different programmatic themes.			×

Table 3: Four topic areas that are defined by the ACM's Special Interest Group on Computer Science Education (SIGCSE). The last three columns indicate how the three sublines (5.1 to 5.3) are connected to these topic areas.

positioning the departmental research line.[11]

Several reasons motivate having a CS education research line in the CS department. Firstly, it connects the department's research activities with educational programs. Teaching practices inspire new research goals to pursue. Reversely, research outcomes such as tools, instructional designs, and best practices can be included in CS courses or inform CS curricula. Secondly, a substantial part of the MSc students works as CS teacher in higher education, generally for a university of applied sciences. This group is often interested in selecting an educational topic with an element of computing for their graduation assignment. Thirdly, the OU has a long-standing tradition in research on educational technology, with research groups that are located in different faculties. This research line connects disciplinary practices with this tradition.

The research line consists of three sublines that are interrelated. The first subline (Section 7.5.1) studies several aspects of programming education including the design, construction, refactoring, and testing of programs, as well as tool support for each of these aspects (e.g. for generating automated feedback). The second subline (Section 7.5.2) explores human factors that influence CS education: this includes collaborative and socio-technical learning, but also efforts to improve diversity and inclusion in computer science. The third subline (Section 7.5.3) focuses on digital literacy.

7.5.1 Programming education

Programming is one of the core areas of Computer Science. Although introductory programming courses are typically positioned in the first semester of CS curricula, research shows that many students struggle with completing these courses [18]. This makes programming education, which focuses on learning and teaching how to program, an important research area. The emphasis of our department's research line is on higher education, even though many excellent initiatives exist that target young kids and teenagers [35, 10].

[11] See `https://www.sigcse2024.org/info/topics`: a fifth topic area that is used for identifying research methods of submitted papers is excluded here.

Writing a program consists of several phases, including problem analysis, design, coding, testing, and debugging. From a research perspective, each phase introduces challenges and new concepts that have to be learned. Also combining these phases adds complexity for students and teachers. In this research line, we consider multiple programming paradigms, each with their unique characteristics: imperative programming (at the method level), object-oriented programming (at the class level), and functional programming. For example, the correct application of OO-patterns can be investigated for the design phase, code-quality issues can be raised at the method level and may lead to refactorings, and program synthesis techniques can assist programmers in completing their code.

In particular, the research group studies tool support and technology-enhanced learning, often in the context of programming education. We develop automated feedback and analysis tools, and we investigate what the effects of such tools are in a classroom setting. These tools can support students with step-wise feedback for solving a problem, detect misconceptions and common mistakes, and provide a personalised and adaptive learning environment (also called "Intelligent Tutoring System").

Slogan

We explore tools and investigate teaching practices and students' learning, thus supporting the teaching and learning of all phases associated with program development.

Refactoring. Refactoring is the process of enhancing the structure of software, preserving the observable behavior, to make it easier to understand, modify, and extend [8]. With a growing volume of program code and an increasing demand for new functionalities, the importance of refactoring is also on the rise. The question of how to guarantee behavior preservation [31] still remains unresolved to a satisfactory degree [19].

One way to ensure behavior preservation is by formally proving the program's semantics in advance. In the case of languages with formally defined semantics, it is possible to demonstrate that certain refactorings maintain the program's semantics [34]. However, for highly complex languages such as Java, accomplishing this task is exceedingly challenging, if not infeasible [40].

Another well-known technique is 'testing' [19]. With testing, predefined tests that specify the desired behavior are used to determine whether the intended relationship between input and output remains intact during and after the refactoring process [8]. One problem with testing is that verification occurs after the refactoring has been executed, errors can go unnoticed if the test cases do not sufficiently cover all the desired behaviors, or if the behavior is not adequately specified. Furthermore, existing test code must be in sync with the refactored code to be tested [32]. Moreover, while failed test cases do indicate the absence of certain desired behaviors, they do not always provide a clear indication of the specific code location responsible for the issue.

A third known technique for achieving behavior preservation is by specifying 'refactoring preconditions' that must be satisfied before a refactoring is executed [31]. This approach is employed in various refactoring engines, including Eclipse and NetBeans. However, issues arise with this method, as refactoring engines may impose preconditions that are either too lenient or excessively restrictive, leading to situations where incorrect transformations are permitted while correct ones are hindered [21]. Furthermore, when potential problems arise, the range of available solutions is often limited.

In this research, we try to develop a technique that allows for an identification of risks before carrying out refactoring and presenting them to the programmer. In the second step, we aim to create a mechanism that provides recommendations for preventing or resolving these risks. Instead of the necessity of initially establishing specifications for a program to be refactored, such as defining input-output tests or refactoring preconditions, the technique examines the smallest changes at the code level (microsteps) and derives conclusions about potential changes in behaviour.

The tool we want to develop is interesting for both experienced programmers as well as computer science students.

References and more

- Ebrahim Rahimi, Harrie Passier and, Sylvia Stuurman, Exploring Factors Influencing the Satisfaction of Adult Software Engineering Students with Teamwork in Distance Education, 23rd Koli Calling International Conference on Computing Education Research, Helsinki, Finland
- Sylvia Stuurman, Harrie Passier, Erik Barendsen, Analyzing Students' Software Redesign Strategies, in Koli Calling '16: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Finland, ACM, 2016
- Lex Bijlsma, Arjan Kok, Harrie Passier, Harold Pootjes and Sylvia Stuurman, Evaluation of design pattern alternatives in Java, in Software: Practice and Experience, 2021 (12), Wiley Online Library.
- M. Lawende, H. Passier, G. Alpár. Reproduction for Insight: Towards Better Understanding the Quality of Students Tests. In Proceedings of ITiCSE 2021. ITiCSE 2021

Program synthesis. The goal of program synthesis is to automatically construct (part of) a program that satisfies a given specification. Such a specification can be a combination of different forms: a formal specification expressed in logic, input-output examples that illustrate the program's desired behaviour, a type signature, etc. Program synthesis is essentially a search problem, where the large search space makes the problem difficult. Two approaches exist and both are studied:

- using the programming language's semantics and type system for propagating top-level specifications to sub-parts, and to reason with these local properties;
- using generative-AI technology on big data of programming solutions for finding

candidate programs (e.g. GitHub's Copilot).

For both cases, a valid research question is how such technology can be applied usefully to support learning.

Automated feedback. We continue the long-standing research line on automated feedback generation, which has resulted in the IDEAS framework[12] and several intelligent programming tutors that are based on this framework and that have been tested in class (e.g., Ask-Elle [9], the collection of Logic Tools [17], and the Refactor Tutor [15]). The framework offers several advanced concepts that allow for rapid prototyping and extend the general knowledge of building intelligent tutoring systems (ITS): the specification of problem-solving procedures as a domain-specific language for generating step-wise feedback, hybrid solutions that mix the model tracing and and constraint-based modeling ITS paradigms, and buggy rules for describing common mistakes.

The IDEAS framework is based on state-of-the-art knowledge about ITSs. It supports VanLehn's inner and outer feedback loops that structures the sequencing of tasks and the step-wise feedback within such a task [42]. It also follows the traditional four-component architecture [28], which prescribes how the system should be decomposed into smaller parts. One such component, which is the domain reasoner or expert module, has received ample attention so far. We plan to investigate how the student model component can add to the feedback possibilities (e.g. by supporting an Open Learning Model) and make the system more adaptive. Computations by these components are offered to front-end UI systems as feedback services: these services are typically modelled after feedback types found in literature [25].

References and more

- Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education, 19(1):1–43, 2019.
- Bastiaan Heeren and Johan Jeuring. Automated feedback for mathematical learning environments. Proceedings of the 14th International Conference on Technology in Mathematics Teaching – ICTMT 14: Essen, Germany, 17-25, 2020.
- Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas Binsbergen. Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. International Journal of Artificial Intelligence in Education, 27(1):65–100, 2017.

Software Testing. Testing is regarded as a crucial activity in software development. It is unclear, however, how acquisition of testing skills can be combined with learning how to program. The inherent complexity of programming, together with known conceptual and strategic difficulties of novice programmers, makes the integration of testing in an early stage of computer science curricula a non-trivial issue. We develop and analyze

[12] The software framework has been released at `https://hackage.haskell.org/package/ideas`.

testing education approaches aiming to align seamlessly with early steps in introductory programming. Moreover, we investigate how to foster an inquiry-based approach to testing in programming education, in line with strategies applied by testing professionals.

References and more

- Doorn, N., Vos, T., Marín, B., & Barendsen, E. (2023). Set the right example when teaching programming: Test Informed Learning with Examples (TILE). In 2023 IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 269-280). IEEE.
- Doorn, N., Vos, T., Marín, B., Bockisch, C., Dick, S., & Barendsen, E. (2023, May). Domain TILEs: Test Informed Learning with Examples from the Testing Domain. In International Conference on Research Challenges in Information Science (pp. 501-508). Cham: Springer Nature Switzerland.

7.5.2 Human factors in CS education

Software is produced by people for people. We investigate and facilitate human and social factors in Computer Science and programming education. We approach human factors in CS education from three perspectives: social and collaborative learning, personalized learning, and diversity and inclusion.

Slogan

We introduce and implement progressive educational methods and tools in programming education, emphasizing human factors such as social learning, personalization, and diversity recognition.

Social and collaborative learning. Our approach to social and collaborative learning in CS education is multi-faceted, including:

- investigate how students co-learn the CS and programming topics;
- encourage teamwork and peer-based learning and incorporate group projects; and
- fostering a supportive and motivating learning environment.

By researching and addressing these aspects, we aim at harnessing the potential of social and collaborative learning to create interactive learning environments for programming education. In these learning environments, students learn from and help each other, improving their programming as well as soft skills and problem-solving skills through (co-)development of team-based programming projects. This process provides them with ample opportunities to share their ideas, problems, solutions, and exchange feedback.

Personalized learning. Regarding personalized learning, our goal is to facilitate and motivate a more student-centered learning experience for CS students. This includes identifying and catering to individual learning needs, interests, difficulties, and competencies of students in the context of CS education. To this end, we intend to explore and leverage the potential of game-based learning, technology-enhanced learning, and generative AI to support and motivate a more student-centered approach to learning programming and computer science.

Diversity and inclusion. Diversity and inclusion are becoming essential principles in computer science education. These principles aim to ensure fair access, learning experience, participation, and success for all CS students, regardless of their gender, educational background, race, or other individual differences and characteristics. Supporting these principles in CS and programming education calls for a multi-faceted approach, encompassing various aspects. This includes designing and implementing an inclusive curriculum and developing inclusive teaching methods and course materials, taking into account the diverse learning needs of underrepresented students.

7.5.3 Digital literacy

Digital literacy refers to knowledge, skills and attitudes necessary to apply digital technology in a variety of contexts, including daily life, other (non-computing) subjects, and professions. There are several ways to characterize digital literacy skills. A framework currently used in Dutch primary and secondary educational settings distinguishes basic digital skills, media literacy, information literacy and computational thinking.

Slogan

We investigate students' learning, instructional strategies and teachers' knowledge and skills in order to support teaching and learning of digital competencies across curricula in a variety of contexts.

Computational thinking. The term Computational Thinking refers to a set of problem solving skills that make use of concepts and methods stemming from computer science. Computational Thinking is the thought process used to understand and formulate problems in such a way that they can be solved in terms of computations. Common elements in characterizations of Computational Thinking are decomposition, abstraction, algorithmic thinking, evaluation and generalization. We investigate students' understanding and instructional strategies for Computational Thinking, especially with respect to abstraction skills of primary school students. Special attention is given to teachers' knowledge and skills required for incorporating Computational Thinking in their lessons.

References and more

- Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education*, 23(2), 159-187.
- Yeni, S., Grgurina, N., Saeli, M., Hermans, F., Tolboom, J., & Barendsen, E. (2023). Interdisciplinary Integration of Computational Thinking in K-12 Education: A Systematic Review. *Informatics in Education*.
- Faber, H. H., Koning, J. I., Wierdsma, M. D., Steenbeek, H. W., & Barendsen, E. (2019). Observing abstraction in young children solving algorithmic tasks. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 95-106). Cham: Springer International Publishing.
- Yeni, S., Nijenhuis-Voogt, J., Hermans, F., & Barendsen, E. (2022). An Integration of Computational Thinking and Language Arts: The Contribution of Digital Storytelling to Students' Learning. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education* (pp. 1-10).
- Barendsen, E., Gorissen, P., Van Rens, C., & Coetsier, N. (2023). *Computational thinking: veelzijdiger dan enkel programmeren.* Onderwijskennis, NRO. `https://www.onderwijskennis.nl/kennisbank/computational-thinking-veelzijdiger-dan-enkel-programmeren`

Information literacy. A key competence to prepare students to deal with complex problems in a professional or learning context is information problem solving (IPS), also known as information literacy (IL), commonly defined as the ability to find, access, evaluate, synthesize, and use information. Despite its significance, research has repeatedly shown that many students are underdeveloped in IPS. This lack of IPS can be problematic for students in terms of accessing information and knowledge. It can result in them making decisions based on inaccurate, one-sided, or out-of-date information. On the other hand, information literate students know how to use and produce digital information to actively support their learning, further increasing their chances of academic and societal success. We investigate learning environments aimed to foster IPS in a higher education setting.

7.6 Impact

The research lines of the department are strategically positioned to make a profound impact on various fronts, including technological, educational, industrial, and societal.

7.6.1 Technological impact

Advancements in technology are at the core of our research. We aim to drive technological impact by creating and extending publicly available **tools, systems and libraries**. Our research results in the development of new tools, systems, and methodologies for software testing, analysis, and verification. For example:

- The TESTAR tool is an open source BSD3 tool (`https://github.com/TESTARtool`) for scriptless testing at the GUI level. In our pursuit of industrial impact, we organize hands-on sessions, or inspiration sessions, to provide companies with firsthand experience in using our specialized testing tool. This approach not only fosters a deeper understanding of our tools but also encourages industries to integrate them into their testing workflows.
- FoxDec is an open-source disassembler and decompiler (`https://ssrg-vt.github.io/FoxDec/`) that aims at retrieving a model from a binary that has been formally proven correct. FoxDec has been used to address challenges created by Raytheon BBN (`https://www.rtx.com/who-we-are/we-are-rtx/transformative-technologies/bbn`) for the DARPA research institute. These challenges have been used for the evaluation of FoxDec in an industrial setting.
- TopHat [39, 38] is a domain specific programming language (DSL) to describe workflow systems and business processes in a paradigm we call task-oriented programming. These specifications can be visualized and introduced a structured way to gradually develop such visualizations. It allows us to symbolically execute workflows to verify their correctness [27]. We also use this symbolic execution engine to generate next-step hints for end users [26]. Besides that, we developed a way to reason about the equivalence of task-oriented programs [16].
- The Ampersand project has yielded a compiler that generates information systems based on a formal specification (`https://ampersandtarski.github.io/`). It solves the problem of large IT projects by automating the design of information systems. Ampersand [13] takes a semantic approach by formalizing the semantics of the business (domain semantics) and generating fully functional prototypes from it. This facilitates incremental design and build approaches such as DevOps. Making an Ampersand model has proven useful in the requirements elicitation stage for yielding conceptual models that are ready to build. Generating systems has proven useful in the design stage for making prototypes that allow co-creation with users. Generating functional specifications has demonstrated its use in relieving architects from unpopular documentation work.
- Satuio is a (prototype) tool for generating adaptive distinguishing sequences and unique input/output sequences for finite state machines (`https://github.com/Jaxan/satuio`). It can generate short tests which verify in which state a system is. Such tests can also be used as a basis for conformance testing of finite state machines.

- We develop new techniques and extensions to the open-source KeY theorem prover (`https://www.key-project.org/`). KeY is tailored for formal verification of Java programs. New techniques are needed to optimize and make scalable the activities of the human proof architect and further enable verification of complex software written in the Java programming language.
- Library functions are used as the building blocks for millions of programs. If our analysis of these libraries reveals bugs, we will publish a fixed (and verified) version of the library and aim to incorporate this new version in the standard library. This way, all programs that use these libraries and the software engineering community at large benefit automatically from our new technology.
- We expect to be contributing to the development of state of the art AI technologies by e.g. developing new types of neural networks, new approaches to logic programming with predicates defined through representation learning, the definition of new approaches to training neural networks, to reinforcement learning conforming to declarative specifications, the specification of neurosymbolic models for planning in industry settings is also an area where we expect we can create impact.
- The educational tools that we develop advance the technological state-of-the-art and are often based on programming language technologies. For example, the intelligent tutoring systems that are based on the Ideas framework use techniques from generic programming (e.g. for rewriting and traversals), domain-specific languages, and advanced type system extensions. Similarly, the refactoring tool uses program analyses and is offered as a plug-in for integrated development environments (IDEs).

7.6.2 Academic impact

We aim to publish high-impact scientific papers on conferences and journals. In addition, we seek collaboration with academic research partners, both in the field of computer science and other disciplines including information science, law, and social sciences. For proof of work show a graph from PURE showing the publications starting from 2016?

7.6.3 Educational impact

The educational impact is reached by incorporating our research findings, expertise and tools into our educational programs. Especially the tools we develop have impact on our education. Moreover, graduation assignments help to further improve the tools. For example:

- Logic Tools for practicing equational, axiomatic, and inductive proofs: these tools are used in bachelor and premaster courses.
- The virtual security lab, of which we researched both the technical infrastructure

and application for education, that is used in the bachelor course on security.

- The TESTAR tool is used in our course on Software verification and testing (`https://www.ou.nl/opleiding?sku=im0903`).
- The course Rule Based Design [14] (`https://www.ou.nl/-/IM0403_Rule-Based-Design`) teaches how to design information systems with rules rather than code. Students are exposed to Ampersand as an example of a tool that does rule-based design. It uses a platform called RAP (`https://rap.cs.ou.nl`), which has been built in Ampersand itself [20]. RAP is the first application that Ampersand has generated to run in production.
- The ANIMO tool [36], [37] is being used in biomedical education at the University of Twente and at the Leiden University Medical Center as an instrument to model and analyze complex biological networks.
- The KeY tool (`https://key-project.org/`) for which we developed extensions and applications to major case studies, is used extensively in different universities in several countries, such as TU Darmstadt, Karlsruhe Institute of Technology, Chalmers, Carnegie Mellon, Uppsala and Oslo. In Germany alone, yearly more than 500 students learn to use KeY.

Moreover, we want to improve the success we have had with publishing Bachelor and Master thesis results, i.e. [6], [2], [41], [24], [4],[33], [3], [11], [5], [7], [12].

7.6.4 Industrial impact

Our research does not stop at academia; it has significant practical applications within companies and industries. The industrial impact of our work includes:

- **Software development:** Our research contributes to the development of better software products, benefiting software companies, startups, and tech industries.
- **Process optimization:** We provide insights and tools for optimizing software development and testing processes, reducing costs and development time.

This industrial impact is achieved through close collaboration with industry partners, technology transfer, and the incorporation of our research findings into industrial practices. We will actively engage with industry partners to collaboratively address real-world software quality challenges, making our research industry-driven.

Ampersand has been used in practice in various places, most notably at Ordina and TNO-ICT. Ordina uses Ampersand as a prototyping tool and a tool for semantic analysis of information systems. In the past, Ordina has used it during the design of large information systems in the public sector, e.g. INDIGO (Dutch Immigration Authority, IND) and DTV (Dutch Food Authority, NVWA). TNO-ICT, a major Dutch industrial research laboratory, has used Ampersand for research purposes, patent research, and demonstrator software. TNO-ICT has built several information systems that it now

maintains in production, in the context of the Semantic Treehouse project (`https://www.semantic-treehouse.nl/`).

The libraries that we analyze and improve are mainly developed in industry. For example, the Java Collection Framework is under development by Oracle and Google (for use in their Android platform). We will engage directly with industry by reporting discovered bugs, fixes, enhancements and test cases to the libraries to the official channels, such as the Java bug tracker. This way our improvements can be incorporated in the standard libraries that are in control and use by these companies and have direct impact on their software development, as well as millions of other software developers worldwide.

Another example is the Dagobert-project in which we researched the detection of botnets by analyzing network traffic by applying AI. This was done in cooperation with internet service providers and network infrastructure providers.

7.6.5 Social impact

Our Software Engienering research is dedicated to improving software quality, which directly translates to positive effects on society, communities, and specific groups. By providing insights and best practices for software quality, our research could influence policy decisions related to technology standards and regulations, ensuring that software systems are developed to the highest standards. Moreover, our work could directly contribute to preventing issues related to system failures and disruptions that could impact people's lives.

More specifically, society today relies on software, in particular the software libraries that form the building blocks of programs. For instance, the Java Collection Framework is used as the standard library on Android and cloud services and programs written in the mainstream Java programming language, which means that its library functions run on the devices of billions of users every day. By analyzing and improving such libraries, society thus directly benefits from using rigorously verified, trusted software components.

Our research on privacy directly relates to the privacy of citizens and organisations. We also research the impact of security measures, such as the electricity consumption and environmental footprint of security mechanisms in cryptocurrencies. Our research results has raised awareness, are being applied to enhance blockchain-based applications, and are used by governments and institutions to define legislation and policies. For instance, in our Econsensus project, we defined a code of conduct for researchers, we provided input for the US policy on cryptocurrency mining, and co-authored a report of the World Economic Forum.

The focus on both robust and safe AI, and co-creation with humans seeks to bridge the gap between theoretical advancements and real-world AI integration, ensuring that our research not only contributes to both academic knowledge but also the practical evolution of AI technologies in society and industry. It is therefore only natural that the approach taken by the Open University concerning AI research and education is that of

Open Science. This means that OU will focus on working with naturally replicable and open approaches to data and AI models, allowing open access to the publications, and working with open source as the preferred software development methodology.

The impact on society of the CS education research line consists of multiple aspects:

- Barendsen chaired the committee that was responsible for designing a new curriculum for the final exam subject Computer Science for secondary education (havo/vwo).
- Barendsen acts as an expert advisor in national curriculum revision and monitoring on digital literacy in primary and secondary education. He chairs the advisory board for the core curriculum ('kerndoelen') with respect to digital literacy.
- By publishing books on guidelines for inclusive education and autism (Autisme is geen puzzel [2021], Autisme-inclusief hoge onderwijs [to appear]), Stuurman informs a diverse audience about neurodiversity and helps teachers to make their education more accessible.

7.7 Organization of the department and meetings

The organizational structure of the department is depicted in Figure 2. It is a multifaceted structure consisting of several interconnected components that define the institution's governance and operational dynamics.

At the head of the department is the Department Management Team, which is lead by Dr. Bastiaan Heeren. This core team includes specialized roles such as the Faculty's Research Coordinator and the program leaders of the departments educational programs.

Diverging from the management core are two primary branches: Research Lines and Educational Programs, each with its own set of leaders and agendas. This structured approach not only emphasizes the institution's commitment to a broad spectrum of research but also underscores the foundational layer of education.

Strategic Meetings form the backbone of the department's communication and decision-making processes, fostering an environment of regular reflection and strategic planning. The department organizes different research meetings to achieve engagement and cohesion between the members of the department and their research results.

OUrsi is a two-weekly research seminar that aims to provide a platform for the researchers in the CS group as well as visitors to share their preliminary as well as mature research results with each other in an informal setting. The seminar seeks a balance between presentations from each of the research lines, as well as balancing talks from more senior group members with presentations from more junior members.

GenAI in CS Education is a quarterly symposium that aims to provide a platform for

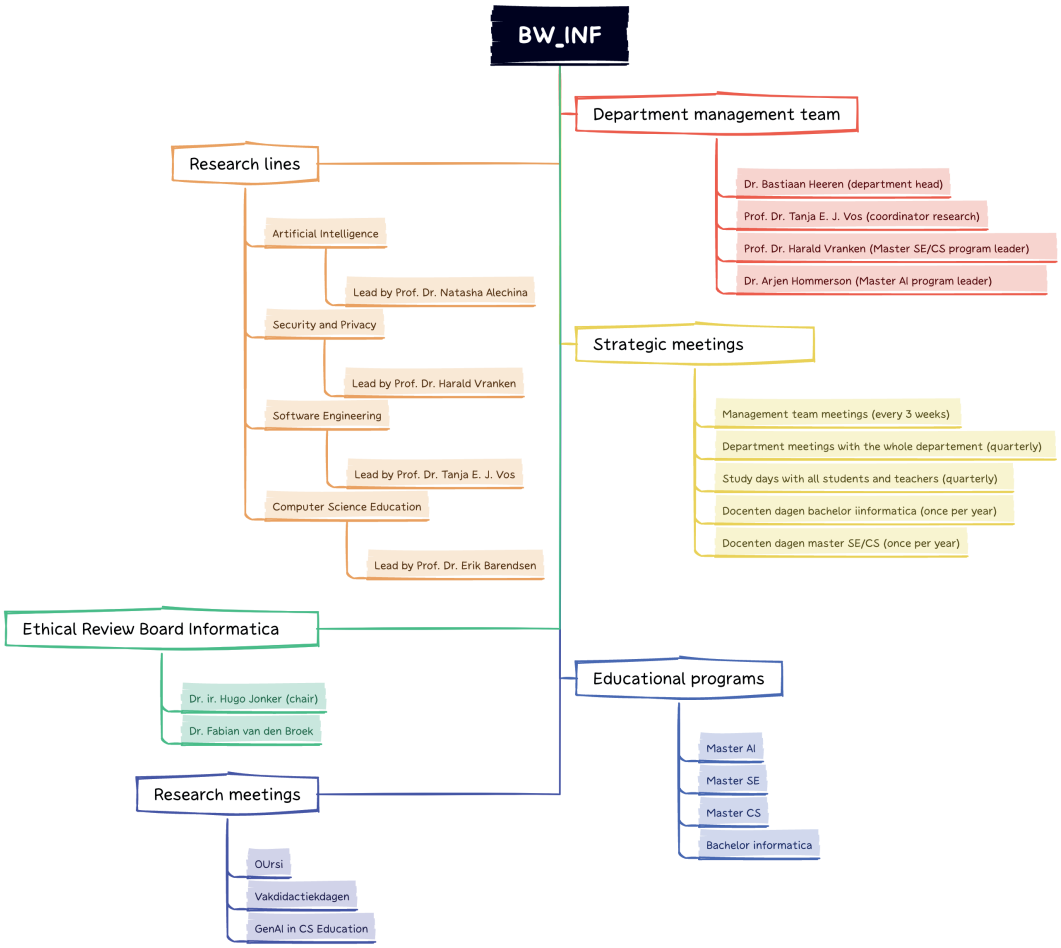


Figure 2: Organizational structure of the department

researchers and educators in computer science, software engineering, and programming education in higher education who are active or interested in the educational application of GenAI and ChatGPT within these disciplines. It is designed for them to share their research findings and good educational practices, collaborate on writing research proposals and grants, contribute to policymaking and regulations, and seek synergies with similar initiatives in GenAI within higher education.

Network meetings on Computing Education research ('Vakdidactiekdagen') These network meetings are organized four times a year. They connect a broad country-wide community of researchers with interests in computing education. The events serve as a platform to share research results, get in touch with colleagues and potential project partners, and to pitch new ideas. Moreover, the meetings offer a safe and constructive environment in which beginning researchers (in particular bachelor's and master's students and PhD candidates) benefit from feedback on their plans and preliminary work. Traditionally, for each event a recent research paper is selected to be read by the participants and discussed during the event, thus contributing to the role of the network as a learning community.

Finally, following the advice from the KNAW report "Ethical and Legal Aspects of Computer Science Research" (2017), the Computer Science department has established an Ethical Review Board for Informatics (ERBi). This body aims to provide easily accessible ethical advice on computer science-related research. The procedure for formal ethical approval for research remains unchanged and goes through the CETO.

7.8 Scientific and societal partners and collaborations

We have a strong collaboration with some entities that we would like to enforce and extend in the future.

7.8.1 Radboud University

We are closely connected with the Institute for Computing and Information Sciences (ICIS) at Radboud University. This holds in particular for the Digital Security group in ICIS: about 10 of our researchers have a position as guest researcher or have a secondment in this group. Of these researchers, some visit Radboud University regularly, while others even have their office at Radboud University. This strong physical presence facilitates contacts and cooperation, both for research and education. These researchers also act as supervisors for thesis projects at Radboud University. Since these projects are often done externally with industry or governmental institutions, this also provides a direct connection for researchers to private and public organisations.

To add: there are also links with the software science and data science groups at ICIS.

7.8.2 Virginia Tech

An active and fruitful collaboration has been maintained with Virginia Polytechnic Institute and State University (Virginia Tech) for years. The SSRG research group op prof. Ravindran has close ties to several researchers of the OU computer science group, which has lead to collaborative research, resource sharing, and joint funded research proposals. Most notably, Virginia Tech and OU share a funded DARPA proposal, and a funded NSF proposal. The objective of this research is to combine knowledge available at a US technical university (low-level OS programming, detailed hardware knowledge) with the mathematical and formal knowledge available at OU. This has lead to several joint publications, exchange of personnel and sharing of ideas and funding opportunities.

7.8.3 Technical University of Valencia

The collaboration with the Technical University of Valencia (Universidad Politécnica de Valencia (UPV)) is part of the Software Testing research led by Tanja Vos. This collaboration is unique in that this group is spanning two locations in two countries: the Open Universiteit (OU) in Heerlen and the Universidad Politécnica de Valencia (UPV) in Spain. The group was portrayed in the July edition of the I/O magazine[13] published by the ICT Research Platform Nederland (IPN).

7.8.4 University of Twente

An ongoing research collaboration with the University of Twente focuses on the application of formal modeling techniques in the field of biomedical research. The work is part of a larger quantitative biology research effort aimed at understanding cell fate decisions, with diagnostic and therapeutic applications. Teams from Leiden University Medical Center and Radboud University are also integral part of the research effort, which has been leading to a number of joint publications. Ideally, in the future the collaboration would entail shared research projects including funding and PhD supervision.

References

[1] Assessment Committee (Rik Leemans, Jos Benders, Michel van Eeten, and Wim Lambrechts). Midterm review Learning and Innovation in Resilient Systems 2014-2016. Technical report, Open Universiteit, October 31, 2017.

[2] Axel Bons, Beatriz Marín, Pekka Aho, and Tanja E.J. Vos. Scripted and scriptless gui testing for web applications: An industrial case. *Information and Software Technology*, 158:107172, 2023.

[13] https://ict-research.nl/wordpress/wp-content/uploads/2023/06/IO-magazine-NR2-2023_vWEB.pdf

[3] Jelle Bouma, Stijn de Gouw, and Sung-Shik Jongmans. Multiparty session typing in java, deductively. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*, volume 13994 of *Lecture Notes in Computer Science*, pages 19–27. Springer, 2023.

[4] Hatim Chahim, Mehmet Duran, Tanja E. J. Vos, Pekka Aho, and Nelly Condori Fernandez. Scriptless testing at the gui level in an industrial setting. In Fabiano Dalpiaz, Jelena Zdravkovic, and Pericles Loucopoulos, editors, *Research Challenges in Information Science*, pages 267–284, Cham, 2020. Springer International Publishing.

[5] Martin de Boer, Stijn de Gouw, Jonas Klamroth, Christian Jung, Mattias Ulbrich, and Alexander Weigl. Formal specification and verification of jdk's identity hash map implementation. In Maurice H. ter Beek and Rosemary Monahan, editors, *Integrated Formal Methods - 17th International Conference, IFM 2022, Lugano, Switzerland, June 7-10, 2022, Proceedings*, volume 13274 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2022.

[6] F. de Gier, D. Kager, S. de Gouw, and Tanja E. J. Vos. Offline oracles for accessibility evaluation with the testar tool. In *2019 13th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12, May 2019.

[7] Erwin de Jager and Stijn de Gouw. Hybrid analysis of BPEL models with grammars. In Yannis Manolopoulos, George A. Papadopoulos, and Theodoros Tzouramanis, editors, *Proceedings of the SOFSEM 2020 Doctoral Student Research Forum colocated with the 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020), Limassol, Cyprus, January 20-24, 2020*, volume 2568 of *CEUR Workshop Proceedings*, pages 73–84. CEUR-WS.org, 2020.

[8] Martin Fowler, Steven Fraser, Kent Beck, Bil Caputo, Tim Mackinnon, James Newkirk, and Charlie Poole. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[9] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas Binsbergen. Ask-elle: an adaptable programming tutor for haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, 27(1):65–100, 2017.

[10] Felienne Hermans. Hedy: a gradual language for programming education. In *Proceedings of the 2020 ACM conference on international computing education research*, pages 259–270, 2020.

[11] Hans-Dieter A. Hiep, Olaf Maathuis, Jinting Bian, Frank S. de Boer, and Stijn de Gouw. Verifying openjdk's linkedlist using key (extended paper). *Int. J. Softw. Tools Technol. Transf.*, 24(5):783–802, 2022.

[12] Hans-Dieter A. Hiep, Olaf Maathuis, Jinting Bian, Frank S. de Boer, Marko C. J. D. van Eekelen, and Stijn de Gouw. Verifying openjdk's linkedlist using key. In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part II*, volume 12079 of *Lecture Notes in Computer Science*, pages 217–234. Springer, 2020.

[13] Stef Joosten. Relation algebra as programming language using the ampersand compiler. *Journal of Logical and Algebraic Methods in Programming*, 100:113–129, 2018.

[14] Stef Joosten, Lex Wedemeijer, and Gerard Michels. *Rule Based Design*. Open Universiteit, Heerlen, 2013.

[15] Hieke Keuning, Bastiaan Heeren, and Johan Jeuring. A tutoring system to learn code refactoring. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, pages 562–568, New York, NY, USA, 2021. Association for Computing Machinery.

[16] Tosca Klijnsma and Tim Steenvoorden. Semantic equivalence of task-oriented programs in tophat. In Wouter Swierstra and Nicolas Wu, editors, *Trends in Functional Programming - 23rd International Symposium, TFP 2022, Virtual Event, March 17-18, 2022, Revised Selected Papers*, volume 13401 of *Lecture Notes in Computer Science*, pages 100–125. Springer, 2022.

[17] J.S. Lodder. *The Design and Use of Tools for Teaching Logic*. PhD thesis, September 2020.

[18] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 125–180. 2001.

[19] Tom Mens and Tom Tourwé. A survey of software refactoring. *IEEE Trans. Softw. Eng.*, 30(2):126–139, February 2004.

[20] Gerard Michels. *Development Environment for Rule-based Prototyping*. PhD thesis, Open University of the Netherlands, June 2015.

[21] Melina Mongiovi, Rohit Gheyi, Gustavo Soares, Márcio Ribeiro, Paulo Borba, and Leopoldo Teixeira. Detecting overly strong preconditions in refactoring engines. *IEEE Transactions on Software Engineering*, 44(5):429–452, August 2018.

[22] MST Research Committee (Carolien Kroeze, Marjolein Caniëls, Dave Huitema, and Harald Vranken). Learning and Innovation in Resilient Systems: MST Research Program 2015-2020. Technical report, Open Universiteit, December 18 2014.

[23] MST Research Committee (Petru L. Curseu, Marjolein Caniëls, Dave Huitema, Harold Krikke, Harald Vranken, and Annemarie Cremers). Learning and Innovation

in Resilient Systems 2014-2016. Technical report, Faculty of Management, Science and Technology (MST), Januari 10, 2017.

[24] Ad Mulders, Olivia Rodriguez Valdes, Fernando Pastor Ricós, Pekka Aho, Beatriz Marín, and Tanja E. J. Vos. State model inference through the gui using run-time test generation. In Renata Guizzardi, Jolita Ralyté, and Xavier Franch, editors, *Research Challenges in Information Science*, pages 546–563, Cham, 2022. Springer International Publishing.

[25] Susanne Narciss. Feedback strategies for interactive learning tasks. In J.M. Spector, M.D. Merrill, J.J.G. van Merriënboer, and M.P. Driscoll, editors, *Handbook of Research on Educational Communications and Technology*. Mahaw, NJ: Lawrence Erlbaum Associates, 2008.

[26] Nico Naus and Tim Steenvoorden. Generating next step hints for task oriented programs using symbolic execution. In Aleksander Byrski and John Hughes, editors, *Trends in Functional Programming - 21st International Symposium, TFP 2020, Krakow, Poland, February 13-14, 2020, Revised Selected Papers*, volume 12222 of *Lecture Notes in Computer Science*, pages 47–68. Springer, 2020.

[27] Nico Naus, Tim Steenvoorden, and Markus Klinik. A symbolic execution semantics for tophat. In Jurriën Stutterheim and Wei-Ngan Chin, editors, *IFL '19: Implementation and Application of Functional Languages, Singapore, September 25-27, 2019*, pages 1:1–1:11. ACM, 2019.

[28] Hyacinth S. Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, 1990.

[29] Department of Computer Science. Computer science research program 2020-2025. Technical report, Open Universiteit, 2020.

[30] School of Computer Science. Software Technology Research Plan 2010-2015. Technical report, Open Universiteit, May 9, 2011.

[31] William F Opdyke. *Refactoring object-oriented frameworks*. University of Illinois at Urbana-Champaign, 1992.

[32] Harrie Passier, Lex Bijlsma, and Christoph Bockisch. Maintaining unit tests during refactoring. In *Proceedings of the 13th International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*, PPPJ '16, pages 18:1–18:6, New York, NY, USA, 2016. ACM.

[33] Fernando Pastor Ricós, Arend Slomp, Beatriz Marín, Pekka Aho, and Tanja E.J. Vos. Distributed state model inference for scriptless gui testing. *Journal of Systems and Software*, 200:111645, 2023.

[34] Maurizio Proietti and Alberto Pettorossi. Semantics preserving transformation rules for prolog. *ACM SIGPLAN Notices*, 26(9):274–284, 1991.

[35] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Eve-

lyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[36] Stefano Schivo, Jetse Scholma, Paul E van der Vet, Marcel Karperien, Janine N Post, Jaco van de Pol, and Rom Langerak. Modelling with ANIMO: between fuzzy logic and differential equations. *BMC Systems Biology*, 10, July 2016.

[37] Jetse Scholma, Stefano Schivo, Ricardo A Urquidi Camacho, Jaco van de Pol, Marcel Karperien, and Janine N Post. Biological networks 101: computational modeling for molecular biologists. *Gene*, 533(1):379–84, January 2014.

[38] Tim Steenvoorden. *TopHat: Task-Oriented Programming with Style*. PhD thesis, Radboud University, Nijmegen, the Netherlands, 2022.

[39] Tim Steenvoorden, Nico Naus, and Markus Klinik. Tophat: A formal foundation for task-oriented programming. In Ekaterina Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 17:1–17:13. ACM, 2019.

[40] Lance Tokuda and Don Batory. Evolving object-oriented designs with refactorings. *Automated Software Engg.*, 8(1):89–120, jan 2001.

[41] Aaron van der Brugge, Fernando Pastor Ricos, Pekka Aho, Beatriz Marín, and Tanja E.J. Vos. Evaluating TESTAR's effectiveness through code coverage. In S. Abrahão Gonzales, editor, *XXV JISBD*. SISTEDES, 2021.

[42] Kurt VanLehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3):227–265, 2006.