Report from Dagstuhl Seminar 23411

Accountable Software Systems

Bettina Könighofer[*1], Joshua A. Kroll[*2], Ruzica Piskac[*3], Michael Veale[*4], and Filip Cano Córdoba[†5]

1 **TU Graz, AT.** bettina.koenighofer@iaik.tugraz.at
2 **Naval Postgraduate School – Monterey, US.** jkroll@jkroll.com
3 **Yale University – New Haven, US.** ruzica.piskac@yale.edu
4 **University College London, GB.** m.veale@ucl.ac.uk
5 **TU Graz, AT.** filip.cano@iaik.tugraz.at

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23411 "Accountable Software Systems". The seminar brought together an interdisciplinary group of researchers from the fields of formal methods, machine learning, philosophy, political science, law, and policy studies to address the critical issue of accountability in the development and deployment of software systems. As these systems increasingly assume roles within safety-critical domains of society, including transportation, healthcare, recruitment, and the judiciary, the seminar aimed to explore the multifaceted concept of accountability, its significance, and its implementation challenges in this context.

During the seminar, experts engaged deeply in discussions, presentations, and collaborative sessions, focusing on key themes such as the application of formal tools in socio-technical accountability, the impact of computing infrastructures on software accountability, and the innovation of formal languages and models to improve accountability measures. This interdisciplinary dialogue underscored the complexities involved in defining and operationalizing accountability, especially in light of technological advancements and their societal implications. The participants of the seminar reached a consensus on the pressing need for ongoing research and cross-disciplinary efforts to develop effective accountability mechanisms, highlighting the critical role of integrating socio-technical approaches and formal methodologies to enhance the accountability of autonomous systems and their contributions to society.

Seminar October 8–13, 2023 – https://www.dagstuhl.de/23411
2012 ACM Subject Classification Applied computing → Law, social and behavioral sciences
Keywords and phrases accountability, Responsible Decision Making, Societal Impact of AI
Digital Object Identifier 10.4230/DagRep.13.10.24

1 Executive Summary

Bettina Könighofer (TU Graz, AT)
Joshua A. Kroll (Naval Postgraduate School – Monterey, US)
Ruzica Piskac (Yale University – New Haven, US)
Michael Veale (University College London, GB)

Accountability in the context of software is an emerging area that has attracted interest in disparate fields from computing and information science to philosophy to political science to law and policy studies. Presently, there is an increasing number of autonomous agents

[*] Editor / Organizer
[†] Editorial Assistant / Collector

Accountable Software Systems, *Dagstuhl Reports*, Vol. 13, Issue 10, pp. 24–49
Editors: Bettina Könighofer, Joshua A. Kroll, Ruzica Piskac, and Michael Veale
Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

shifting into safety-critical aspects of society and our everyday lives. Autonomously driven vehicles share the roads with us. Computerized reasoning aids healthcare providers in disease diagnostics, recruiters in hiring, and even adjudicators in analyzing pretrial flight risks. While a common goal of these autonomous agents is to assist and improve human lives, it is a harsh reality that the opposite occurs far too frequently. Car accidents involving autonomously driven vehicles have been fatal, and bias-vulnerable autonomous decision-making has led to discriminatory assessment and abuse of individuals. As a result, accountability is started to be investigated in areas including AI, machine learning, control, systems development, data management, software engineering, programming languages, human factors/human-computer interaction, and formal verification communities.

Understanding accountability and the place of formal computing tools in assigning it has been recognized by funding agencies as an important research topic. Significant reforms are currently occurring in the law to the liability of autonomous systems, such as the proposed EU Artificial Intelligence Act; reforms to the EU Machinery Directive and Product Liability Directive; standardization processes around software and accountability at the ISO, IEEE, CEN/CENELEC, US NIST, and ETSI, among others; a range of proposals around reforms to intermediary liability and recommender systems in the US Congress and proposals from the White House Office of Science and Technology Policy, among many other ongoing changes and revisions to existing policies and recommendations for new policies around the world. Throughout the Dagstuh Seminar, participants engaged in a rich tapestry of discussions, presentations, and working groups, focusing on three main areas:

- Justifying Assurance: Opportunities and Limits of Formal Tools for Accountability in Sociotechnical Context
- The Role(s) of Computing and Infrastructures in Accountability for Software
- New Formal Specification Languages and Modeling Techniques for Accountability

Group discussions further enriched the seminar's discourse. For example, debates around the Software as a Service (SaaS) model and the potential for a public, democratically managed cloud infrastructure highlighted the evolving nature of software maintenance and the democratization of digital infrastructure. Another group delved into the intricacies of software complexity, distinguishing between accountability and responsibility, and discussing the challenges of regulatory adaptation to technological innovation.

By having short talks to inspire discussions followed by in-depth discussions in breakout groups, the seminar successfully forged new conversations between the communities with a focus on how to improve upon the state of the art and practice in designing software systems with respect to accountability. Participants collectively recognized the importance of socio-technical approaches and formal methods in developing robust accountability frameworks, advocating for synergies between different fields to refine the state of the art and practice.

The seminar concluded with a group exercise to map and group open questions, which are documented at the end of this report. This exercise naturally provided more questions than answers, but these questions were coalescing on deep and specific interdisciplinary challenges that participants intended to take forwards in their future work.

In conclusion, the Dagstuhl Seminar on "Accountable Software Systems" served as a pivotal forum for cross-disciplinary exchange, catalyzing new conversations and potential collaborations.

2 Table of Contents

3 Overview of Talks

3.1 Analyzing Intentional Behavior in Autonomous Agents under Uncertainty

Filip Cano Córdoba (TU Graz, AT)

Joint work of Filip Cano Córdoba, Samuel Judson, Timos Antonopoulos, Katrine Bjørner, Nicholas Shoemaker, Scott J. Shapiro, Ruzica Piskac, Bettina Könighofer

Main reference Filip Cano Córdoba, Samuel Judson, Timos Antonopoulos, Katrine Bjørner, Nicholas Shoemaker, Scott J. Shapiro, Ruzica Piskac, Bettina Könighofer: "Analyzing Intentional Behavior in Autonomous Agents under Uncertainty", in Proc. of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, pp. 372–381, ijcai.org, 2023.

URL https://doi.org//10.24963/IJCAI.2023/42

Principled accountability for autonomous decision-making in uncertain environments requires distinguishing intentional outcomes from negligent designs from actual accidents. We propose analyzing the behavior of autonomous agents through a quantitative measure of the evidence of intentional behavior. We model an uncertain environment as a Markov Decision Process (MDP). For a given scenario, we rely on probabilistic model checking to compute the ability of the agent to influence reaching a certain event. We call this the scope of agency. We say that there is evidence of intentional behavior if the scope of agency is high and the decisions of the agent are close to being optimal for reaching the event. Our method applies counterfactual reasoning to automatically generate relevant scenarios that can be analyzed to increase the confidence of our assessment. In a case study, we show how our method can distinguish between "intentional" and "accidental" traffic collisions.

3.2 Is Software Eaten by the Cloud?

Corinne Cath (TU Delft, NL & University of Cambridge, GB)

In 2011, Marc Andreessen well-known tech investor and inventor of the short-lived but influential Mosaic browser, proclaimed that "software is eating the world". This was a period of rapid digitization during which time more and more industries were running on software – and changing their business models to one of delivering online services rather than selling one-off products. A second, but perhaps more silent revolution accompanied by the glutenous expansion of software and its as-a-service business model, is happening today. This one is not being splashed across the front page of the Wall Street Journal. Namely the growing importance of cloud computing, as the infrastructure for the "agile" production and distribution of software [1]. In this talk, I focus on the growing importance of cloud computing in the context of the future of software ecosystems. In particular, I argue that the future of software lives on the cloud. Cloud computing is the on-demand availability of computing resources (such as storage and infrastructure), as services over the internet, at scale. It ostensibly eliminates the need for individuals and businesses to self-manage physical resources and allows them to pay for what they use. Many software companies buy computing resources from a small set of tech behemoths, including Google, Microsoft, and AWS. But this perceived convenience comes at a price. I draw on several recent case studies that outline

the importance of cloud computing for the recent boom in the development of generative AI (GenAI) products, like ChatGPT and Claude. I demonstrate that when we talk about GenAI, we implicitly assume that this software runs on the cloud, given GenAI's specific computational requirements. This cloud reliance creates distinct dependencies between the software ecosystem and cloud computing companies, their financial models and political priorities, which have accountability ramifications that require further research.

References

1 Gürses, Seda, and Joris van Hoboken. 2018. "Privacy after the Agile Turn." In The Cambridge Handbook of Consumer Privacy, edited by Evan Selinger, Jules Polonetsky, and Omer Tene, 579–601. Cambridge Law Handbooks. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781316831960.032.

3.3 Causal Explanations: What Can Computer Scientists Do for Accountability

Hana Chockler (King's College London, GB)

Computerised systems are increasingly opaque, due to their size and to the nature of the new AI systems, such as neural networks. Yet, we have to be able to reason about the correctness of these systems, debug them, fix errors, answer "what if" questions, and explain their decisions. In this talk, I present a framework for causal explanations of image classifiers, based on the notions of actual causality and explainability. Our tool ReX provides explanations that approximate minimal subsets of the image sufficient to get the same classification if the rest of the image is hidden. I describe our recent work in extending ReX to multiple explanations and survey the challenges in adapting the existing explainability tools to medical applications.

References

1 Chockler and Halpern. "Responsibility and Blame: A Structural-Model Approach". J. Artif. Intell. Res. 22: 93-115 (2004)
2 Chockler, Halpern, Kupferman. What causes a system to satisfy a specification? ACM Trans. Comput. Log. 9(3): 20:1-20:26 (2008)
3 Aleksandrowicz, Chockler, Halpern, Ivrii. "The Computational Complexity of Structure-Based Causality". AAAI'14: 974-980.
4 Alrajeh, Chockler, Halpern. "Combining Experts' Causal Judgments". Artif. Intell. (2020).
5 Sun, Chockler, Huang, Daniel Kroening. "Explaining Image Classifiers Using Statistical Fault Localization". ECCV'20: 391-406.
6 Chockler, Kroening, Sun. "Explanations for Occluded Images". ICCV'21: 1234-1243.
7 Pouget, Chockler, Sun, Kroening. "Ranking Policy Decisions". NeurIPS'21.
8 Chockler, Halpern. "On Testing for Discrimination Using Causal Models". AAAI'22.

3.4 What Does Data Erasure Mean? What Should Data Erasure Mean?

Aloni Cohen (University of Chicago, US)

Joint work of Aloni Cohen, Adam D. Smith, Marika Swanberg, Prashant Nalini Vasudevan
Main reference Aloni Cohen, Adam D. Smith, Marika Swanberg, Prashant Nalini Vasudevan: "Control, Confidentiality, and the Right to be Forgotten", in Proc. of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023, pp. 3358–3372, ACM, 2023.
URL https://doi.org//10.1145/3576915.3616585

Recent digital rights frameworks stipulate an ability to request that a data controller – roughly, any system that stores and manipulates personal information – "erase" or "delete" one's data (e.g., the "right to be forgotten" in the GDPR). We ask how deletion should be formalized in complex systems that interact with many parties and store derivative information. There are two broad principles at work in existing approaches to formalizing deletion: confidentiality and control. This talk briefly explores these questions in the specific context of machine learning models, and erasing training data therefrom.

3.5 Complexity Effects on a Highly-Accountable System Containing Safety-Critical Software

Misty Davies (NASA – Moffett Field, US)

This talk began with a quick overview of the regulatory landscape for aviation. Today, there are distinctly different regulatory process paths for hardware/software system components and for the human operators and their roles in the overall system. Today, the handling of almost all remaining uncertainty is pushed into the operational assurance processes. This means that hardware and software components are expected to be highly deterministic. As we increasingly push required system functionality from execution by people into execution by automation we are running into the limits of what our current processes allow us to assure. This is broadly perceived as a barrier to innovation.

One way forward could be a shift to assurance models that merge and blend our current paradigms for silicon-based and human-based assurance.

References

1 Brat, Guillaume P., Huafeng Yu, Ella Atkins, Prashin Sharma, Darren Cofer, Michael Durling, Baoluo Meng et al. Autonomy Verification & Validation Roadmap and Vision 2045. No. NASA/TM-20230003734. 2023.

3.6 Accountable Software Systems: Lessons from System Safety

Roel Dobbe (TU Delft, NL)

Main reference Roel Dobbe: "System Safety and Artificial Intelligence", in Proc. of the FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 – 24, 2022, p. 1584, ACM, 2022.
URL https://doi.org//10.1145/3531146.3533215

We draw inspiration from the field of system safety to understand challenges with accountability in the development, use and governance of software systems. System safety has dealt with accidents and harm in safety-critical systems subject to varying degrees of software-based automation, already for many decades. As a result, it has some crystallized lessons, tools and ontologies that establish conditions for accountability to promote or guarantee safety, as well as to inform the design of actual mechanisms for accountability. We covered two core concepts that capture such conditions and mechanisms. At the operational level, the process model which details the goals, constraints, actions, observations and models/knowledge needed to safely operate processes subject to software-based automation. At the organizational and institutional levels, the safety control structure combines, describes and relates all mechanisms contributing to safety, including the operational mechnisms, and extending towards managerial, maintenance and research and development. Moving further up, the control structure also includes various regulatory, policy, advocacy law-making, law-enforcing and judicial mechanisms. System safety methods allow for the integral analysis of unsafe or otherwise undesirable behaviors and outcomes in software systems, helping to understand how various mechanisms contribute to material outcomes, either directly in a causal sense or indirectly by providing the environmental factors for such system behavior and outcomes to emerge. We concluded with a few lessons written down in Nancy Leveson's magnum opus "Engineering a Safer World: Systems Thinking Applied to Safety", including the curse of flexibility and the crucial importance of culture in upholding accountability mechanisms.

References

1 Roel Dobbe. System Safety and Artificial Intelligence. In The Oxford Handbook of AI Governance. Oxford University Press, 2022, https://doi.org/10.1093/oxfordhb/9780197579329.001.0001.
2 Nancy G. Leveson. Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press, 2012. http://ebookcentral.proquest.com/lib/delft/detail.action?docID=3339365

3.7 Accountability in Computing: Concepts and Mechanisms

Joan Feigenbaum (Yale University – New Haven, US)

Joint work of Joan Feigenbaum, Aaron D. Jaggard, Rebecca N. Wright
Main reference Joan Feigenbaum, Aaron D. Jaggard, Rebecca N. Wright: "Accountability in Computing: Concepts and Mechanisms", Found. Trends Priv. Secur., Vol. 2(4), pp. 247–399, 2020.
URL https://doi.org//10.1561/3300000002

This firestarter talk was a brief "teaser" of the work in [1].

Accountability is a widely studied but amorphous concept, used to mean different things across different disciplines and domains of application. Here, we survey work on accountability in computer science and other disciplines. We motivate our survey with a study of the myriad

ways in which the term "accountability" has been used across disciplines and the concepts that play key roles in defining it. This leads us to identify a temporal spectrum onto which we may place different notions of accountability to facilitate their comparison. We then survey accountability mechanisms for different application domains in computer science and place them on our spectrum. Building on this broader survey, we review frameworks and languages for studying accountability in computer science. Finally, we offer conclusions, open questions, and future directions.

References

1 Joan Feigenbaum, Aaron D. Jaggard, and Rebecca N. Wright (2020). Accountability in Computing: Concepts and Mechanisms. Foundations and Trends in Privacy and Security: Vol. 2: No. 4, pp 247-399.

3.8 AI is a Mushroom

Jake Goldenfein (The University of Melbourne, AU)

Main reference Jake Goldenfein: "Privacy's Lose Grip: Law and the Operational Image" in Zalnierute et al (eds) Cambridge Handbook on the Regulation of Facial Recogntion in the Modern State (CUP 2024) (forthcoming)

A number of scholars ahve identified the complexity of locating organisational accountability in AI products because of complex industrial arrangements. But there are similar complexities in the dataset supply chain for AI models. Datasets used to train commercial and noncommercial models are produced and refined by numerous organisations with different characteristics and for different purposes. These inter-organisational forms of coordination and strategic partnership result in complex "ecologies" of data flow subject to complex forms of regulatory arbitrage. But what kind of "ecology" is this? Inspired by the work of Anna Tsing, the goals here is to understand or conceptualise these mutualistic relations and multi-organisational associations, the data flows they coordinate, and the products they produce as a type of mushroom, whose arrangement depends on a broader political economy.

3.9 Accountable Legal Decision Support?

Thomas T. Hildebrandt (University of Copenhagen, DK)

Joint work of Amine Abbad Andaloussi, Lars Rune Christensen, Søren Debois, Nicklas Pape Healy, Hugo A. López, Morten Marquard, Naja L. Holten Møller, Anette C. M. Petersen, Tijs Slaats, Barbara Weber

Main reference Thomas T. Hildebrandt, Amine Abbad Andaloussi, Lars Rune Christensen, Søren Debois, Nicklas Pape Healy, Hugo A. López, Morten Marquard, Naja L. Holten Møller, Anette C. M. Petersen, Tijs Slaats, Barbara Weber: "EcoKnow: Engineering Effective, Co-created and Compliant Adaptive Case Management Systems for Knowledge Workers", in Proc. of the ICSSP '20: International Conference on Software and System Processes, Seoul, Republic of Korea, 26-28 June, 2020, pp. 155–164, ACM, 2020.

URL https://doi.org//10.1145/3379177.3388908

Case management and business processes regulated by law are increasingly being digitalized, in the public as well as the private sector. At the same time, laws are continuously changed and new regulations are introduced. Moreover, law needs to be open for interpretation

and re-interpretation. This calls for new methods to ensure accountability of the legal decision support. We present the DCR Graph tools and notation developed and tested in the EcoKnow.org project, which allows domain experts to map the law to a DCR Graph (which is essentially a temporal logic knowledge graph) and validate the meaning through simulation. Via the tools developed by the research-based company DCR Solutions, the graphs can be used to provide dynamic guidelines, case management support and automation, and is used widely in the public sector in Denmark via the KMD WorkZone Enterprise Information Management system.

References

1. Christoffer Olling Back, Tijs Slaats, Thomas Troels Hildebrandt, Morten Marquard: DisCoveR: accurate and efficient discovery of declarative process models. Int. J. Softw. Tools Technol. Transf. 24(4): 563-587, 2022
2. Vlad Paul Cosma, Thomas T. Hildebrandt, Christopher H. Gyldenkærne, Tijs Slaats. *BERMUDA: Participatory Mapping of Domain Activities to Event Data via System Interfaces.* ICPM Workshops 2022: 127-139, 2022
3. Asbjørn Ammitzbøll Flügge, Thomas T. Hildebrandt, Naja L. Holten Møller. *Street-Level Algorithms and AI in Bureaucratic Decision-Making: A Caseworker Perspective.*Proc. ACM Hum. Comput. Interact. 5(CSCW1): 40:1-40:23, 2021
4. Anette C. M. Petersen, Lars Rune Christensen, Richard Harper, Thomas T. Hildebrandt, *"We Would Never Write That Down": Classifications of Unemployed and Data Challenges for AI.* Proc. ACM Hum. Comput. Interact. 5(CSCW1): 102:1-102:26, 2021
5. Amine Abbad Andaloussi, Francesca Zerbato, Andrea Burattin, Tijs Slaats, Thomas T. Hildebrandt, Barbara Weber. *Exploring how users engage with hybrid process artifacts based on declarative process models: a behavioral analysis based on eye-tracking and think-aloud.* Softw. Syst. Model. 20(5): 1437-1464. 2021
6. Håkon Norman, Søren Debois, Tijs Slaats, Thomas T. Hildebrandt. *Zoom and Enhance: Action Refinement via Subprocesses in Timed Declarative Processes.* BPM 2021: 161-178
7. Thomas T. Hildebrandt, Håkon Normann, Morten Marquard, Søren Debois, Tijs Slaats. *Decision Modelling in Timed Dynamic Condition Response Graphs with Data.* Business Process Management Workshops 2021: 362-374, 2021
8. Anette Chelina Møller Petersen, Lars Rune Christensen, Thomas T. Hildebrandt: *The Role of Discretion in the Age of Automation.* Comput. Support. Cooperative Work. 29(3): 303-333, 2020
9. Hugo A. López, Søren Debois, Tijs Slaats, Thomas T. Hildebrandt. *Business Process Compliance Using Reference Models of Law.* FASE 2020: 378-399, 2020
10. Asbjørn William Ammitzbøll Flügge, Thomas T. Hildebrandt, Naja L. Holten Møller. *Algorithmic Decision Making in Public Services: A CSCW-Perspective.* GROUP (Companion) 2020: 111-114, 2020
11. Søren Debois, Hugo A. López, Tijs Slaats, Amine Abbad Andaloussi, Thomas T. Hildebrandt. *Chain of Events: Modular Process Models for the Law.* IFM 2020: 368-386, 2020
12. Naja L. Holten Møller, Irina Shklovski, Thomas T. Hildebrandt. Shifting Concepts of Value: Designing Algorithmic Decision-Support Systems for Public Services. NordiCHI 2020: 70:1-70:12, 2020
13. Thomas T. Hildebrandt, et al. *EcoKnow: Engineering Effective, Co-created and Compliant Adaptive Case Management Systems for Knowledge Workers.* ACM, Proceedings of ICSSP '20: International Conference on Software and System Processes, Seoul, Republic of Korea, 26-28 June, 2020

3.10 Platforms, Sovereignty, and Software Accountability

Divij Joshi (University College London, GB)

Main reference Cohen, Julie E.: "Law for the platform economy." UCDL Rev. 51 (2017): 133.
URL https://scholarship.law.georgetown.edu/facpub/2015/

This talk examines why the "platform" model is crucial to thinking about software accountability. Software production, deployment and use is increasingly intermediated through the organisational form of the "platform" – generally private entities which control access to computational resources as well as structure and organise the "market" for software products. Through varying levels of control over elements of a software production or deployment "stack", the platform model is crucial for thinking about accountability in software production.

As platforms increasingly determine access to computational resources and the ability to run softwares, they exercise power over users or populations in ways that may be undemocratic or unjust. Smartphone OS providers can arbitrarily expand the surveillance capabilities of mobile devices or IoT devices, and "App Stores" can similarly chose to unilaterally force changes to software production business models, as we have seen in the recent past.

This also places limits on how we can democratically and legitimately control computation and software production or use that is increasingly "infrastructural", i.e. a common, shared resource that is vital for conducting an array of activities. Given that these infrastructures are globally distributed and usually privately controlled, nation states or other legitimate publics have little insight or control into how governance decisions are made on platforms. At the level of the state, this is also giving rise to calls for expanding upon "digital sovereignty" through a number of measures, including regulation, or creating public alternatives.

3.11 "Put the Car on the Stand": SMT-based Oracles for Investigating Decisions

Samuel Judson (Yale University – New Haven, US)

Joint work of Samuel Judson, Matthew Elacqua, Filip Cano Córdoba, Timos Antonopoulos, Bettina Könighofer, Scott J. Shapiro, Ruzica Piskac
Main reference Samuel Judson, Matthew Elacqua, Filip Cano Córdoba, Timos Antonopoulos, Bettina Könighofer, Scott J. Shapiro, Ruzica Piskac: "'Put the Car on the Stand': SMT-based Oracles for Investigating Decisions", CoRR, Vol. abs/2305.05731, 2023.
URL https://doi.org//10.48550/ARXIV.2305.05731

Principled accountability in the aftermath of harms is essential to the trustworthy design and governance of algorithmic decision making. Legal philosophy offers a paramount method for assessing culpability: putting the agent "on the stand" to subject their actions and intentions to cross-examination. We show that under minimal assumptions automated reasoning can rigorously interrogate algorithmic behaviors as in the adversarial process of legal fact finding. We use the formal methods of symbolic execution and satisfiability modulo theories (SMT) solving to discharge queries about agent behavior in factual and counterfactual scenarios, as adaptively formulated by a human investigator. We implement our framework and demonstrate its utility on an illustrative car crash scenario.

3.12 Algorithmic Systems Through an Ethnographic Lens

Daan Kolkman (TU Eindhoven, NL)

Although algorithms are imbued with a sense of objectivity and reliability, numerous high-profile incidents have demonstrated their fallibility. Despite their ubiquity, access to algorithms and algorithmic systems is typically policed, meaning that our understanding of what it is that people who develop algorithms "do" and why algorithms fail in practice remains largely unexplored. In a way, these systems that are designed for structuring and ordering, themselves often resist straightforward categorization and control. This observation echoes Seaver's concept [3] of software systems as "culturally enacted" through user interaction, challenging the notion of these systems as fixed technical objects. My approach, while similar in nature to Seaver's, took a slightly different point of departure. Specifically, I built on studies by Beunza and Garud [1], Beunza and Stark [2], and Spears [4] that – like Seaver – center on understanding the socio-material context of algorithms. However, such work in the Social Studies of Finance domain also focuses on understanding the "technical" aspects of algorithms themselves. In a recent case study, I investigate the development – and demise – of an algorithmic system that used Wi-Fi data as an input for footfall measurement in the Netherlands. The study utilizes an ethnographic approach to track the development and eventual abandonment of this system that measured footfall in 1100 locations across the Netherlands. Based on the case study I conclude that, in comparison to the manual collection of footfall measurement, the transition to more digital modes of data collection has several unique characteristics that matter for the processes of knowledge production: 1. The speed and volume of these new modes of data collection is much higher; 2. The new modes of data collection are relatively untried, hard to calibrate, and in general more error-prone; 3. New modes of data collection can be – and are – contested on the basis of non-digital observations. In unison this leads to the – attempted- alignment of divergent measures of footfall and integration previously isolated epistemic cultures. In pursuit of more understanding of – and more influence over – algorithmic systems, ethnographic studies like these may contribute by shedding light on the mundane practices of those involved in their development and use.

References

1 Beunza, D., & Garud, R. (2007). Calculators, lemmings or frame-makers? The intermediary role of securities analysts. The Sociological Review, 55 (2 suppl), 13-39
2 From dissonance to resonance: Cognitive interdependence inquantitative finance. Economy and society, 41(3), 383-417
3 Algorithms as culture: Some tactics for the ethnography of algorithmic systems. Big Data & Society, 4 (2), 1-12
4 Engineering value, engineering risk: what derivatives quants know and what their models do. PhD Thesis

3.13 Verification of Accountability in Protocols with Tamarin

Robert Künnemann (CISPA – Saarbrücken, DE)

Joint work of Robert Künnemann, Kevin Morio, Ilkan Esiyok
Main reference Kevin Morio, Robert Künnemann: "Verifying Accountability for Unbounded Sets of Participants", in Proc. of the 34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021, pp. 1–16, IEEE, 2021.
URL https://doi.org//10.1109/CSF51468.2021.00032

The Tamarin [1] protocol verification tool (`https://tamarin-prover.github.io/`) integrates support for verifying protocol accountability in the following sense: "protocol provides accountability for property iff, at all times, the protocol can correctly determine the parties causing violation of this property". We first explain that we need causation to define this property, and that, more precisely, causes ought to be the fact whether or whether not a party deviated from protocol behavior. Then we outline how this is done, and how this technique developed to support an unbounded number of parties. We talk about the case studies we performed and demonstrate the Tamarin tool and its web GUI.

References

1 Meier, Simon, et al. "The TAMARIN prover for the symbolic analysis of security protocols." Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. Springer Berlin Heidelberg, 2013.

3.14 Accountability Lessons Learned from the Design and Deployment of Digital Contact Tracing

Wouter Lueks (CISPA – Saarbrücken, DE)

Main reference Carmela Troncoso, Dan Bogdanov, Edouard Bugnion, Sylvain Chatel, Cas Cremers, Seda F. Gürses, Jean-Pierre Hubaux, Dennis Jackson, James R. Larus, Wouter Lueks, Rui Oliveira, Mathias Payer, Bart Preneel, Apostolos Pyrgelis, Marcel Salathé, Theresa Stadler, Michael Veale: "Deploying decentralized, privacy-preserving proximity tracing", Commun. ACM, Vol. 65(9), pp. 48–57, 2022.
URL https://doi.org//10.1145/3524107

With the onset of the global pandemic in early 2020 came a massive push to deploy digital technologies to aid in pandemic mitigation. In this talk, we looked back at how the DP3T project proposed a privacy-friendly approach to digital contact tracing [1], eventually leading to its adoption by Google and Apple. This adoption was an essential enabler of the use of these technologies, but also surfaces questions of accountability. What does it mean to design solutions on such a short timescale? How can citizens convince themselves that the privacy protections are implemented as designed? And what does it mean for Google and Apple to control what health applications can and cannot do [2]?

References

1 Carmela Troncoso, Dan Bogdanov, Edouard Bugnion, Sylvain Chatel, Cas Cremers, Seda F. Gürses, Jean-Pierre Hubaux, Dennis Jackson, James R. Larus, Wouter Lueks, Rui Oliveira, Mathias Payer, Bart Preneel, Apostolos Pyrgelis, Marcel Salathé, Theresa Stadler, Michael Veale: *Deploying decentralized, privacy-preserving proximity tracing.* Commun. ACM 65(9): 48-57 (2022)
2 James R. Larus: *Whose smartphone is it?* Commun. ACM 64(9): 41-42 (2021)

3.15 Accountability and Explainability of French Housing Benefits Computation

Denis Merigoux (INRIA – Paris, FR)

Joint work of Marie Alauzen, Émile Rolley, Louis Gesbert, Justine Banuls

How French government agencies explain decisions taken by an IT system following the law? Typical example is the amount of taxes and social benefit you get/pay once you've filled the form. With several other authors, we have done : a state of the art, interviews with social workers and social benefits agency agents about what use they have for explainability, how this plays out with respect to accountability, and prototyping automatic generation of law-based explanations using the Catala programming language.

Investigated model of transparency, in french law you have to publish the source code. In the case of housing benefits distributed by CNAF, cobol code from 1995, unreadable. So that is a problem. French law forces them to publish a summary of what the algorithm is doing but you can't condense 300 pages in 2-pages that they published, if you add all the edge cases together you get the entire population. Last requirement, for each decision that is made by an algo you should provide a detailed, individualize, and legible explanation. We looked at all the govt systems that should provide these, but none of them do. From there, we wanted to investigate deeper on the housing benefits case and interviewed social workers and CNAF agents about their use of explainability of decisions. Findings: low-level social workers completely trust the algorithm, higher-level workers sort of understand the principles of how it works but without being able to link the behavior to the law that specifies it. To contest it, detect an algorithm error or debug it you need link between law and code through a detailed explanation of how thing was computed. However at CNAF there's a lenghty process between law and code with two different specification/documentation sets written by different groups of people. Recommendation : you should be able to link the law, the code and the explanations of how the code executed, while automatically producing explanations and sharing them externally to have more accurate contestations from the outside for maybe better debugging of the algorithm.

3.16 An AI Transparency Register for the Public Sector

Matthias Spielkamp (AW AlgorithmWatch – Berlin, DE)

Joint work of Michele Loi, Anna Mätzener, Angela Müller, Matthias Spielkamp
Main reference Michele Loi, Anna Mätzener, Angela Müller, Matthias Spielkamp: "Automated Decision-Making Systems in the Public Sector – An Impact Assessment Tool for Public Authorities". AlgorithmWatch, Berlin, Germany, 2021
URL https://algorithmwatch.org/en/wp-content/uploads/2021/09/2021_AW_Decision_Public_Sector_EN_v5.pdf

When using automated decision-making systems (ADM systems) in the public sector, authorities act in a unique context and bear special responsibilities towards the people affected. Against this background, the use of ADM systems by public administrations should be subject to stringent transparency mechanisms – including public registers and mandatory impact assessments.

Without the ability to know whether ADM systems are being deployed, all other efforts for the reconciliation of fundamental rights and ADM systems are doomed to fail. Legally mandatory public registers of all ADM systems used by public administrations – at communal, regional, national, and supranational levels – should therefore be created.

These registers should come with the legal obligation for those responsible for the ADM system to disclose information on the underlying model of the system, its developers and deployers, the purpose of its use, and the results of the algorithmic impact assessment.

In order to make a difference in practice, ethical reflection must be translated into ready-to-use tools, providing authorities with the means for conducting such an analysis. To this end, we have developed a practical, user-friendly, and concrete impact assessment tool, enabling the evaluation of an ADM system throughout its entire life cycle.

If you'd like to add bibliographic references to your abstract, please use the thebibliography-environment (bibtex-files are not supported):

3.17 Responsibility and Liability regarding Software and AI

Rüdiger Wilhelmi (Universität Konstanz, DE)

Looking on responsibility and liability regarding software and especially non-symbolic artificial intelligence from the legal perspective, I start with two distinctions. The first one regards the different branches of law. Public law and especially administrative law concentrate on prohibitions and permissions and criminal law on prohibitions as well. Both necessarily involve the state as such and are mainly statute law. Private law relates to private individuals. Contract law allows the private shaping of law. Tort law regulates compensation for damages on a non-contractual basis. The second distinction is between general and special purpose law. With regard to software there is law aiming at software or digitalization at large like – in the context of the EU – the General Data Protection Regulation, the Digital Markets Act and the Digital Service Act as well as the proposals for an AI Act and an AI Liability Directive. There is also special law in more general statutes like computer fraud in the criminal code or the provisions regarding consumer contracts on digital products in the civil code, that lead to a convergence between software contracts based on the law of sale and those based on rental or service contracts. But software and digitalization at large are also covered by law not directly aiming at them like competition law, ordinary fraud, most of contract law or tort law. Regarding software there are legal relationships between a lot of actors. In general, it starts with a contract between the producer and the user, very often intermediated by a provider or dealer. Often, there is another contract between the user and his client, like a medical practitioner and his patient. On the other end of the chain there are contracts between the producer and his suppliers, like the programmer, the data supplier or trainers. In case of damage, there are non-contractual relationships based on tort governing the compensation. They exist in parallel to the contractual relationships but also independent to these and to especially third parties. My talk concerns responsibility and liability regarding software and especially artificial intelligence concentrated on tort law. Tort law is the branch of private law concerned with the award of compensation for damages even if there is no contractual relationship between the damaging and the damaged party. Basically, tort law is no law designed especially for information technology and digitalization,

but general-purpose law. As such it is the part of law very often concerned first with new developments. The most relevant branch of tort law in this context is product liability. The conditions for product liability based on negligence in essential are an infringement of legal interests, a breach of duty of care or conduct and causation. Based on strict liability they are an infringement of legal interest, being active in a specific domain, like road traffic or gene technology, and causation. The difference is, that there is no need to define a duty under strict liability. But there exist exemptions to strict liability with similar effects but being much more defined. Product liability claims can be based on both, negligence and strict liability. They require a defective product that does not provide the safety to expect and causes damages to specified legal interest not comprising pure economic loss. In our context the most relevant defects are design defects and instruction defects. Design defects can be identified form a bird's eye perspective by comparing the product to other products or from a frog perspective by looking whether a specific feature could have been better. Instruction defects fail to inform the users properly, especially to warn of risks. An interesting question is, whether and when flaws in the design can be compensated for by instructions or warnings. In this respect, design and instruction defects are a sort of communicating vessels between the producer and the user and could shift risks between them. The level of safety depends on what is technically possible and on the risks involved. The more risky a product is, the higher the product safety requirements. In extreme cases, the necessary safety level cannot be achieved. This mechanism is also known in other areas of law and is also the basis of the AI Act. In the end, the concrete level of safety to be provided has to be determined by the courts. Standards set by standardization bodies or evolving out of the industry are of limited effect in the sense, that they normally constitute the minimum level, but do not prevent the courts requiring higher levels, what they do. The main problems assorted with software and artificial intelligence are the lack of transparency and related to this of predictability and reliability. But this no new problem to product liability where it is solved by duties to disclosure of evidence and a shift of the burden of proof, in the end making the producer accountable for the lack of transparency, predictability and reliability. There is some indication, that the product liability can cover the problems for responsibility and liability regarding software and artificial intelligence, maybe requiring some minor adaptions. The current discussion about introducing a new (strict) liability regime lacks a sufficient analysis of whether, where and why the existing regime is insufficient.

3.18 Scenic: A Probabilistic Scenario Description Language

Beyazit Yalcinkaya (University of California – Berkeley, US)

Main reference Daniel J. Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia: "Scenic: a language for scenario specification and data generation", Mach. Learn., Vol. 112(10), pp. 3805–3849, 2023.
URL https://doi.org//10.1007/S10994-021-06120-5

This presentation focuses on the integral role of expressive environment modeling tools within the realm of formal methods, underpinning the pursuit of verified AI. In this context, our research group at UC Berkeley has introduced Scenic, a probabilistic scenario description language designed to facilitate the realization of this goal. Scenic serves as a probabilistic programming language, enabling the formulation of distributions that characterize scenes and scenarios, the former representing configurations of physical objects and autonomous agents,

and the latter encompassing distributions over these scenes and the dynamic behaviors of their constituent agents over time. This novel language is distinguished by its succinct and comprehensible syntax for spatiotemporal relationships, accompanied by the capacity to declaratively enforce both hard and soft constraints on the scenario. Furthermore, by considering accountability as an inherent system property, we find the application of Scenic crucial for precisely specifying the environmental conditions under which we can hold a software system accountable. In doing so, we illuminate the potential of Scenic as a means to enhance the accountability of software systems.

3.19 Towards a Framework for Certification of Reliable Autonomous Systems

Neil Yorke-Smith (TU Delft, NL)

Joint work of Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, Neil Yorke-Smith
Main reference Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, Neil Yorke-Smith: "Towards a framework for certification of reliable autonomous systems", Auton. Agents Multi Agent Syst., Vol. 35(1), p. 8, 2021.
URL https://doi.org//10.1007/S10458-020-09487-2

A computational system is called autonomous if it is able to make its own decisions, or take its own actions, without human supervision or control. The capability and spread of such systems have reached the point where they are beginning to touch much of everyday life. However, regulators grapple with how to deal with autonomous systems. We view certification of the behaviour of a system as, in appropriate cases, a component of accountability. This talk proposed to analyse what is needed in order to provide verified reliable behaviour of an autonomous system, analyse what can be done as the state-of-the-art in automated verification, and pointed to a roadmap towards developing regulatory guidelines, including articulating challenges to researchers, to engineers, and to regulators.

4 Working groups

4.1 Working groups topic discussions

We had two sessions where the participants were divided into different working groups to work on the same topic.

4.1.1 Forms of (Un)Accountability in Contemporary Software Ecosystems

The different groups discussed around the following key questions:

- How is software (un)accountable today? To whom, and with what consequences?
- How do different disciplines understand the concept of accountability?
- What are the main practical barriers to accountability?

4.1.2 Software Ecosystem Futures

The different groups discussed around the following key questions:

- How might software be designed, delivered, deployed and maintained in 5, 10, 15+ years? What are the main consequences for accountability?
- How are supply chains and value chains in software being formed and reformed, and what does this mean about which actors could and should be held responsible?
- Is the changing nature or a certain trajectory of software development and change inevitable? What are the main leverage points for driving it in different directions? What different directions exist?

4.2 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 1

Wouter Lueks (CISPA – Saarbrücken, DE) and Scott Shapiro (Yale University – New Haven, US)

In group 1, we discussed different ways of looking at accountability. We realized that the typical computer science notion of accountability as "matching the specification" might not be sufficient. Real systems have to deal with users, and users might not follow the rules. One alternative way of looking at accountability, then is, to consider what you do when users break the rules. Accountability therefore doesn't always have to be about software. In particular, in distributed systems, policy violations cannot always be prevented, instead such systems might resort to identify misbehaving users, or punish them, for example by reducing their utility.

Following this path, we realized that accountability is multiple ambiguous, and we'd have to consider different aspects, all of which relate to accountability:

- Ability: Is it even possible to hold "the thing" to account?
- Attributability: Can you assign credit or blame to a specific individual or even a line of code?
- Answerability: can you give an answer or account that explains the intention
- Liability: can we determine who has to be punished?

In light of these, we looked at the example of machine learning systems. There, answerability might be a challenge. Many systems cannot "explain" (or be made to explain) the choices that they made. And these systems are often operated by large companies. Should we attribute violations to individuals, or to the company as a whole?

We questioned the term "accountable systems" because systems cannot be held accountable, instead maybe "accountability of software" is more appropriate.

One particular challenge with accountability is to attribute problems. For one, this might not always be possible directly. For example, a child who knocks over a precious vase is usually not held accountable. Although their parents might. And a person suffering a acute mental distress might be held accountable for their actions, but not blamed.

In the world of objects, there are often processes in place to assign blame or draw lessons from failures. For example, after an aircraft crash, there is an investigation by an authorized body. Such body does not exist for software (unless it leads to physical harm).

4.3 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 2

Roel Dobbe (TU Delft, NL)

The prompt for this breakout was: How does software creation affect accountability?

Rather than answering this big question head on, the group decided to surface important distinctions that may help to understand the various forms software may take that are relevant to study the role of and impact on accountability.

The distinctions that we identified were:

- Software running in the cloud vs Software running on premise
- Open source vs Closed source
- Software-as-a-service vs Shrink-wrap software (with associated licensing)
- Infrastructure-as-a-service (one-stop shop) vs Open infrastructures and ecosystems for software development
- Manual requirement generation vs Automated generation of requirements
- General purpose/multi-task software vs Specialized/single-task software
- Centralized control over software vs decentralized control over software
- Software as a utility vs non-utility
- Big tech break up vs Further consolidation of software industry
- Universal software (standards) vs Plural software (standards)
- Quantum-based computing vs Bit-based computing
- Scrutable software architectures vs Inscrutable software architectures

4.4 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 3

Bettina Könighofer (TU Graz, AT) and Neil Yorke-Smith (TU Delft, NL)

In our group discussion, we considered that software complexity acts as an accelerant in the realm of technology, yet it isn't the initial catalyst for technological advancements. It's widely recognized that autonomous systems are on the horizon, and rather than resisting their emergence, we should focus on understanding and shaping their integration responsibly.

A key point of debate was the distinction between accountability and responsibility. While one can be responsible without being blameworthy, accountability often carries the weight of blame, especially in situations where the complexity of a system obfuscates the lines of direct responsibility. This distinction becomes particularly poignant in discussions about how to regulate technologies that are still evolving and the challenges of preparing the public sector through adequate training in data and digital literacy.

The conversation also touched on practical challenges, such as the costly repercussions of fixing bugs in software systems, exemplified by an incident in Los Angeles' air traffic control where a countdown timer bug led to operational chaos. This highlighted the broader issue of how audits, transparency registers, and other mechanisms for oversight struggle to keep

pace with dynamic software systems. The consensus emerged that understanding faults is a prerequisite for attributing blame or punishment, underlining the need for integrating such accountability measures throughout the software lifecycle.

Accountability was thus framed as an organizational challenge rather than purely a technical one, with parallels drawn to self-regulation models like film ratings in the U.S. or the medical association's role in certifying doctors. The discussion acknowledged the universal nature of software as akin to a Turing machine, raising questions about the feasibility of ensuring safety, security, and accountability simultaneously.

Identifying liability within complex software systems presents its own set of challenges, especially when all components function correctly yet the system as a whole fails. The concept of group responsibility was dissected, acknowledging its complexity and distributed nature. Questions were raised about how software can "make it right," including bearing punishment, making restitution, and ensuring justice.

The group considered examples such as rule-based parole systems demonstrating bias, underscoring the "stupidity" of computers in their lack of reasonableness, and the need for human-like foresight in technology design. The "Miracle on the Hudson" incident was cited as a case where software limitations impeded human judgment, suggesting a need for regulatory frameworks that can adapt to the unique challenges of software within various domains.

Finally, the discussions circled back to the foundational problem of software complexity and its role in exacerbating these accountability and regulatory challenges. The conclusion pointed towards the necessity of domain-specific expertise in creating regulatory frameworks for software, questioning what aspects of a system can be automatically checked by another system and what requires more nuanced, human oversight. Through these discussions, the group navigated the intricate web of ethical, technical, and regulatory challenges facing the future of software development and integration.

4.5 Forms of (Un)Accountability in Contemporary Software Ecosystems: Group 4

Beyazit Yalcinkaya (University of California – Berkeley, US)

In our discussion, we approached the forms of (un)accountability in contemporary software ecosystems from various angles. Essentially, our discussion was rooted in a computer science perspective due to the background of the majority of participants. Our responses to specific questions are summarized below.

How is software (un)accountable today? To whom, and with what consequences? Our consensus was that software today is essentially unaccountable as accountability has not been a major focus of software development. On top of that, we do not have a clear understanding of the possible behaviors of contemporary learning-based models. Moreover, even without any learning-based components in the software loop, we observe cases where the problem of accountability is ambiguous. Specifically, we discussed an example from the security literature in which a user interacts with various software services, and even though each software system is secure in itself, the information exposure between their interfaces causes a security vulnerability, so an attacker can capture sensitive information. This is an interesting example because none of the software services is the cause of the

security vulnerability; however, their composition results in an ecosystem with a security vulnerability. Therefore, it is not clear which software service should be held accountable. In general, the composition of software systems is not a trivial process. We concluded that the issue of accountability of software has to be studied well from various aspects in today's complex software ecosystem.

How do different disciplines understand the concept of accountability? Our consensus on this issue was that the term accountability is more of an umbrella term that can be understood from the perspective of various system properties that we care about. These properties mainly include safety, reproducibility, and certifiability. Specifically, the issue of certifiability is a main concern for holding an entity accountable for a software system's behavior. For example, one needs some sort of certification before using the output of a large language model to hold the system accountable for any complication that might occur, such as a copyright issue. Software services should provide a certification to their users so that the user can use this certification in court as a piece of evidence that the material was generated by the specific software service that they used. We concluded that certifiability has to be an interdisciplinary understanding for the accountability of software systems.

What are the main practical barriers to accountability? We started our discussion on this issue by critiquing the idea of transparency being a sufficient condition for the accountability of software systems. Our consensus was that the speed and scale of today's contemporary software systems cannot be handled by any human-level bureaucratic produce that can be accomplished through transparency. Therefore, we need systems that monitor/check themselves, and the processes for accountability must be automated as much as possible. However, to achieve this goal, we need new results from the computer science community to provide what can be formalized (and therefore automated) and what cannot be formalized so that the proper tools for automation can be developed. Then in light of these results and tools, policymakers can provide guidelines for system developers, and therefore better practices for accountable software systems can be enforced. We concluded that to make an accountable software ecosystem possible, policymakers and computer scientist should communicate their needs and capacities clearly to each other.

4.6 Software Ecosystem Futures: Group 1

Divij Joshi (University College London, GB)

Our group discussed how new technologies and organisational forms structure software production and accountability.

We began the discussing how or whether Large Language Models will change software production. There was some disagreement about the impact of these changes, with some participants stressing that automation of programmer's capabilities was already affecting software production, and that this leads to challenges to accountability. One participant spoke about how accountability in the supply chain or production might be becoming irrelevant with systems like zero-trust procurement (which the US government is moving towards).

Next, participants discussed the role of platforms and platform power among software production and use supply chains. They spoke about whether accountability needs to be located within software production processes, or from the perspective of end-users. From this perspective, which actors are "choke points" and which ones can be used to leverage accountability across the ecosystem.

Finally, participants discussed whether and how software production itself might be made more accountable through regulation.

4.7 Software Ecosystem Futures: Group 2

Filip Cano Córdoba (TU Graz, AT)

In Group 2, we did not focus our discussion around large language models (LLMs) or AI, but more on the software as a service model (SaaS). We agreed that how software is maintained will not change that much, perhaps more of it will be moving to the cloud and the Saas model. So, in the future, the user will not own anything, but this paradigm will probably not change much how software is to be developed and maintained. As an interesting side thought, we made the thought experiment of what would happen if instead of the big players that currently dominate the market, we created a public democratically managed cloud infrastructure that would serve as a non-for-profit realistic alternative. This entity would escape the power of both huge for-profit companies and governments, so the profit and the power would be in the hands of the people. We proposed the Wikipedia Foundation as a viable example to follow, albeit the Wikipedia Foundation is a much smaller organization, with a much more focused objective.

On supply chains, we discussed how when catastrophic fails happens, it is typically not the cause of a component-level failure, but a failure at system level. System level properties are generally more difficult to specify and verify, so they consist of an important challenge. We also discussed extensively about dependency-tracking projects, like NIX or the "software bill of materials". While transparency is not all, and full transparency may be infeasible, having a clear description of software dependencies will surely help make software systems accountable along their production and value chains.

About change and its innevitability, we first wanted to point out that nothing is inevitable, almost everything can happen if there is enough will in the public. Maybe the only inevitability is climate change, that will pressure current political systems and regulations to somehow limit the amount of resources dedicated to computing. We discussed how we should stirr this policies so that limited resources get allocated to valuable objectives.

4.8 Software Ecosystem Futures: Group 3

Neil Yorke-Smith (TU Delft, NL)

This breakout group discussed the future of software development, aided by the fresh air of the countryside around Schloss Dagstuhl.

- AI-aided, man-machine co-generation of software is already used at Google, one participant pointed out. The group discussed whether such co-generation will be more participatory and more sustainable in the future.
- The "code local" movement was discussed: developing software closer to the point of use; supply and value chains; the voice of citizens.
- Software projects that arose during the COVID-19 epidemic received discussion, both how software development continued or changed, and how new projects arose. One participant pointed out how "interesting" consortiums came together; critical comments arose about governments and how they procure software.

- One participant suggested that a few people can change the status quo. An example is the few who raised concerns about bias in AI, whereas this is now mainstream.
- The group thought that future software will be more flexible, making accountability more tricky.
- Lastly, the group discussed the role of "big tech". It was noted that software in schools in the Netherlands is highly dependent on Google.

5 Open problems

5.1 Concluding Exercise: Open Questions

Michael Veale (University College London, GB), Thomas Arnold (Tufts University – Medford, US), Filip Cano Córdoba (TU Graz, AT), Corinne Cath (TU Delft, NL & University of Cambridge, GB), Hana Chockler (King's College London, GB), Aloni Cohen (University of Chicago, US), Misty Davies (NASA – Moffett Field, US), Roel Dobbe (TU Delft, NL), Joan Feigenbaum (Yale University – New Haven, US), David Fuenmayor (Universität Bamberg, DE), Ashish Gehani (SRI – Menlo Park, US), Jake Goldenfein (The University of Melbourne, AU), Thomas T. Hildebrandt (University of Copenhagen, DK), Divij Joshi (University College London, GB), Samuel Judson (Yale University – New Haven, US), Daan Kolkman (TU Eindhoven, NL), Bettina Könighofer (TU Graz, AT), Joshua A. Kroll (Naval Postgraduate School – Monterey, US), Robert Künnemann (CISPA – Saarbrücken, DE), Stefan Leue (Universität Konstanz, DE), Wouter Lueks (CISPA – Saarbrücken, DE), Rupak Majumdar (MPI-SWS – Kaiserslautern, DE), Kira Matus (The Hong Kong Univ. of Science & Technology, HK), Denis Merigoux (INRIA – Paris, FR), Ruzica Piskac (Yale University – New Haven, US), Scott Shapiro (Yale University – New Haven, US), Jatinder Singh (University of Cambridge, GB), Matthias Spielkamp (AW AlgorithmWatch – Berlin, DE), Rüdiger Wilhelmi (Universität Konstanz, DE), Beyazit Yalcinkaya (University of California – Berkeley, US), and Neil Yorke-Smith (TU Delft, NL)

In the final session, we all wrote open questions on cards and then grouped them together in emergent, but unnamed groups.

- How to make enforceable/verifiable standards for traceability of requirements/components' provenance or system actors?
- How can we identify mechanisms of coordination in software supply chains?
- Can we imagine software supply chains which align responsibility and resources without handing over societal power?
- How to ensure accountability over Big Tech's use of data?

- How to manage drift in a world with fixed specifications and model structure?
- How universally can AI or ML be regulated?
- How can we do accountability for 'general purpose' machine learning software?

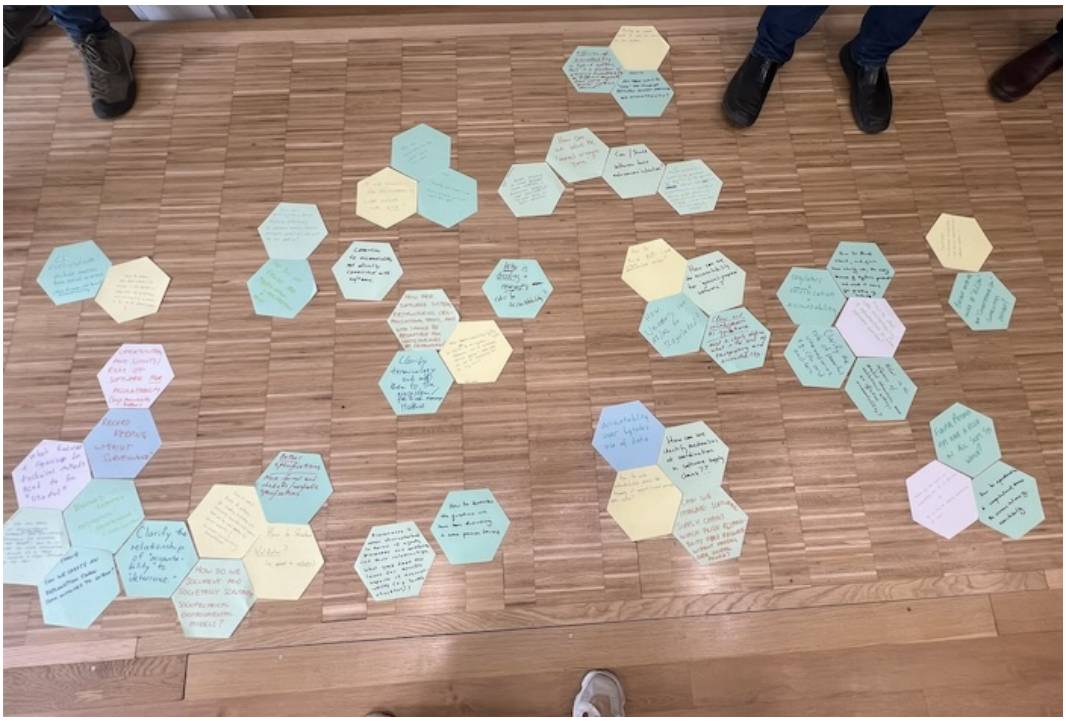

Figure 1 The grouping of the open questions from the final session of the seminar.

- How to write clear and unambiguous AI regulations that clearly define and delimit the level of needed transparency and accountability?

- Can the computer science curriculum include courses from social sciences which discuss, at least, accountability?
- How can we prepare citizens for future debates on the accountability of software systems in society?

- What are the opportunities and limits around the rise of software *for* accountability?
- Can we do record-keeping without opening the door to surveillance? What are the mechanisms for record keeping, and what are the governance mechanisms and implications?
- What features and capacities do technical accountability methods need to be successful?
- For key areas of accountability interest and key metrics describing the "goodness" of that property, how do we capture the state knowledge that allows us to trace the metrics and make decisions after the fact?
- Can we create an explanation chain from outcomes to intent?
- Clarify the relationship of "accountability" to deterrence
- How do we document and societally scrutinise sociotechnical environmental models?
- How do we structure validation (opposed to verification)?
- How can we make tools and methods for modelling, simulating and analysing sociotechnical systems that can be used by domain experts (to enable accountability and trustworthiness)
- Better specifications – more formal and checkable or verifiable specifications

- How do we ensure participation and broad perspectives?
- How can participatory design feature effectively in systems safety/hazard analysis practices, not just as an add on?

- If we focus on accountability, what might we lose?
- How can accountability be used and misused?
- Ex ante accountability impact assessments? Should be used to determine go or no-go for creating the software system.

- How are software systems restructuring organisational tasks and who should be responsible for the consequences of restructuring?
- Clarify terminology and maps it to the ecosystem or political economy
- What capacities or capabilities on the part of humans are required for the long term maintenance and use of accountable software systems? Which actors will hold the responsibility over the long term?
- Who is deciding and managing rules for accountability?

- Diffusion of accountability in hybrid systems, there is a question of assigning accountability to different components. Even worse when several systems interact!
- Building up component properties to system-level assessment, versus pure reductionism.
- Are there ways to "hack" the trade off between system complexity and accountabiilty?

- What divisions of labour between human operators and automation enable clear accountability? (for every sense of definition of accountability!)
- How can we solve the "moral crumple zone"?
- Can and should software have autonomous intention?
- How well do counterfactual scenarios as generated and specified by ML approaches fit with, improve, and sustain counterfactuals as questions between people determining responsibility for actions (what would you have done if "x" had happened instead?)

- Can we clarify the role of powerful intermediaries, e.g. certificate authorities and auditors?
- How to deal with accountability of the infrastructures upon which software operates?
- What is the influence of market structures and institutional settings on effective accountability?

- Formal methods may have a role in accountable software systems – but which?
- How to operationalise in computational terms the various notions of accountability
- What are the aspects of accountability what can be formalised and do we have the necessary tools and theories for these?

- Bridge with work by "AI and Law" and "Computational Law" communities – is it possible?
- How to make maintainable and accountable legal decision support or automation software systems?

Participants

- Thomas Arnold
Tufts University – Medford, US
- Filip Cano
TU Graz, AT
- Corinne Cath
TU Delft, NL & University of Cambridge, GB
- Hana Chockler
King's College London, GB
- Aloni Cohen
University of Chicago, US
- Misty Davies
NASA – Moffett Field, US
- Roel Dobbe
TU Delft, NL
- Joan Feigenbaum
Yale University – New Haven, US
- David Fuenmayor
Universität Bamberg, DE
- Ashish Gehani
SRI – Menlo Park, US
- Jake Goldenfein
The University of Melbourne, AU
- Thomas T. Hildebrandt
University of Copenhagen, DK
- Divij Joshi
University College London, GB
- Samuel Judson
Yale University – New Haven, US
- Bettina Könighofer
TU Graz, AT
- Daan Kolkman
TU Eindhoven, NL
- Joshua A. Kroll
Naval Postgraduate School – Monterey, US
- Robert Künnemann
CISPA – Saarbrücken, DE
- Stefan Leue
Universität Konstanz, DE
- Wouter Lueks
CISPA – Saarbrücken, DE
- Rupak Majumdar
MPI-SWS – Kaiserslautern, DE
- Kira Matus
The Hong Kong Univ. of Science & Technology, HK
- Denis Merigoux
INRIA – Paris, FR
- Ruzica Piskac
Yale University – New Haven, US
- Scott Shapiro
Yale University – New Haven, US
- Jatinder Singh
University of Cambridge, GB
- Matthias Spielkamp
AW AlgorithmWatch – Berlin, DE
- Michael Veale
University College London, GB
- Rüdiger Wilhelmi
Universität Konstanz, DE
- Beyazit Yalcinkaya
University of California – Berkeley, US
- Neil Yorke-Smith
TU Delft, NL