



DAGSTUHL REPORTS

Volume 8, Issue 4, April 2018

Program Equivalence (Dagstuhl Seminar 18151) <i>Shuvendu K. Lahiri, Andrzej Murawski, Ofer Strichman, and Mattias Ulbrich</i>	1
Blockchains, Smart Contracts and Future Applications (Dagstuhl Seminar 18152) <i>Foteini Baldimtsi, Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl</i>	20
Visualization of Biological Data - Crossroads (Dagstuhl Seminar 18161) <i>Jan Aerts, Nils Gehlenborg, Georgeta Elisabeta Marai, and Kay Katja Nieselt</i>	32
Normative Multi-Agent Systems (Dagstuhl Seminar 18171) <i>Mehdi Dastani, Jürgen Dix, Harko Verhagen, and Serena Villata</i>	72
Algebraic Effect Handlers go Mainstream (Dagstuhl Seminar 18172) <i>Sivaramakrishnan Krishnamoorthy Chandrasekaran, Daan Leijen, Matija Pretnar, and Tom Schrijvers</i>	104
Towards Accountable Systems (Dagstuhl Seminar 18181) <i>David Eyers, Christopher Millard, Margo Seltzer, and Jatinder Singh</i>	126
Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research (Dagstuhl Seminar 18182) <i>Pekka Abrahamsson, Jan Bosch, Sjaak Brinkkemper, and Alexander Mädche</i>	164

ISSN 2192-5283

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <http://www.dagstuhl.de/dagpub/2192-5283>

Publication date

November, 2018

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

License

This work is licensed under a Creative Commons Attribution 3.0 DE license (CC BY 3.0 DE).



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Aims and Scope

The periodical *Dagstuhl Reports* documents the program and the results of Dagstuhl Seminars and Dagstuhl Perspectives Workshops.

In principal, for each Dagstuhl Seminar or Dagstuhl Perspectives Workshop a report is published that contains the following:

- an executive summary of the seminar program and the fundamental results,
- an overview of the talks given during the seminar (summarized as talk abstracts), and
- summaries from working groups (if applicable).

This basic framework can be extended by suitable contributions that are related to the program of the seminar, e. g. summaries from panel discussions or open problem sessions.

Editorial Board

- Gilles Barthe
- Bernd Becker
- Daniel Cremers
- Stephan Diehl
- Reiner Hähnle
- Lynda Hardman
- Hannes Hartenstein
- Oliver Kohlbacher
- Bernhard Mitschang
- Bernhard Nebel
- Bernt Schiele
- Albrecht Schmidt
- Raimund Seidel (*Editor-in-Chief*)
- Emanuel Thomé
- Heike Wehrheim
- Verena Wolf

Editorial Office

Michael Wagner (*Managing Editor*)
Jutka Gasiorowski (*Editorial Assistance*)
Dagmar Glaser (*Editorial Assistance*)
Thomas Schillo (*Technical Assistance*)

Contact

Schloss Dagstuhl – Leibniz-Zentrum für Informatik
Dagstuhl Reports, Editorial Office
Oktavie-Allee, 66687 Wadern, Germany
reports@dagstuhl.de
<http://www.dagstuhl.de/dagrep>

Digital Object Identifier: 10.4230/DagRep.8.4.i

Program Equivalence

Edited by

Shuvendu K. Lahiri¹, Andrzej Murawski², Ofer Strichman³, and
Mattias Ulbrich⁴

1 Microsoft Research – Redmond, US, shuvendu.lahiri@microsoft.com

2 University of Oxford, GB, andrzej.murawski@cs.ox.ac.uk

3 Technion – Haifa, IL, ofers@ie.technion.ac.il

4 KIT – Karlsruher Institut für Technologie, DE, ulbrich@kit.edu

Abstract

Program equivalence is the problem of proving that two programs are equal under some definition of equivalence, e.g., input-output equivalence. The field draws researchers from formal verification, semantics and logics.

This report documents the program and the outcomes of Dagstuhl Seminar 18151 “Program Equivalence”. The seminar was organized by the four official organizers mentioned above, and Dr. Nikos Tzevelekos from Queen-Mary University in London.

Seminar April 8–13, 2018 – <http://www.dagstuhl.de/18151>

2012 ACM Subject Classification Software and its engineering → Software verification, Software and its engineering → Semantics

Keywords and phrases program equivalence, regression-verification, translation validation

Digital Object Identifier 10.4230/DagRep.8.4.1


1 Executive summary

Shuvendu K. Lahiri

Andrzej Murawski

Ofer Strichman

Mattias Ulbrich

License  Creative Commons BY 3.0 Unported license

© Shuvendu K. Lahiri, Andrzej Murawski, Ofer Strichman, and Mattias Ulbrich

Program equivalence is arguably one of the most interesting and at the same time important problems in formal verification. It has attracted the interest of several communities, ranging from the field of denotational semantics and the problem of Full Abstraction, to software verification and Regression Testing. The aim of this meeting was to bring together the different approaches and techniques of the current state of the art and to facilitate the cross-pollination of research between these areas.

This interdisciplinary community met once before in the workshop on program equivalence in London (April 2016). There was a general agreement among the participants that a research community around this topic should be established in the form of a workshop and eventually a conference, and that the interest in this topic continuously grows around the world, including a growing interest in the industry. Furthermore, currently there is little overlap in the conferences that some of the key players attend, to the point that many participants were little aware of other participants’ work.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Program Equivalence, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 1–19

Editors: Shuvendu K. Lahiri, Andrzej Murawski, Ofer Strichman, and Mattias Ulbrich



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We were happy to witness that indeed participants learned greatly from this week, collaborations were established, and cross fertilization between the communities occurred. We hope to meet again in Dagstuhl in the future!

2 Table of Contents

Executive summary

Shuvendu K. Lahiri, Andrzej Murawski, Ofer Strichman, and Mattias Ulbrich . . . 1

Overview of Talks

Relational Logic with Framing and Hypotheses (status report)	
<i>Anindya Banerjee and David A. Naumann</i>	5
Semantic Differencing for HipHop Bytecode	
<i>Nick Benton</i>	5
Verification with Reusing Exchangeable Results (Conditions, Witnesses, Precisions)	
<i>Dirk Beyer</i>	5
Validating Optimizations of Concurrent C/C++ Programs	
<i>Soham Chakraborty</i>	6
Semantics-Parametric Program Equivalence	
<i>Stefan Ciobaca</i>	7
Abstract Semantic Diffing of Evolving Concurrent Programs	
<i>Constantin Enea</i>	7
Property Directed Equivalence via Abstract Simulation	
<i>Grigory Fedukovich</i>	8
A graph-rewriting refinement of the β law	
<i>Dan R. Ghica</i>	8
Regression Verification of Multi-Threaded Programs	
<i>Arie Gurfinkel and Ofer Strichman</i>	9
Model-checking contextual equivalence of higher-order programs with references	
<i>Guilhem Jaber</i>	9
Distinguishing between Communicating Transactions	
<i>Vasileios Koutavas</i>	10
DEEPSEC: Deciding Equivalence Properties in Security Protocols	
<i>Steve Kremer</i>	10
Interprocedural Relational Verification in SymDiff and Applications	
<i>Shuvendu K. Lahiri</i>	11
Polymorphic Game Semantics for Dynamic Binding	
<i>James Laird</i>	11
Program equivalences and program refinements for compiler verification	
<i>Xavier Leroy</i>	12
Client-Specific Equivalence Checking – An Overview	
<i>Yi Li and Julia Rubin</i>	13
Semantic Program Repair Using a Reference Implementation	
<i>Sergey Mechtaev</i>	13
An introduction to game semantics	
<i>Andrzej Murawski</i>	14

Trace Equivalence For Android Malware Detection	
<i>Julia Rubin</i>	14
Proofs for Performance	
<i>Rahul Sharma</i>	15
Program equivalence problems in computational science	
<i>Stephen Siegel</i>	15
Proving Mutual Termination	
<i>Ofer Strichman</i>	16
The Software Analysis Workbench	
<i>Aaron Tomb</i>	16
Nominal Games: A Semantics Paradigm for Effectful Languages	
<i>Nikos Tzevelekos</i>	17
Relational Equivalence Proofs Between Imperative and MapReduce Algorithms	
<i>Mattias Ulbrich</i>	17
A Behavioural Equivalence for Algebraic Effects: Logic with Modalities	
<i>Niels Voorneveld</i>	18
Participants	19

3 Overview of Talks

3.1 Relational Logic with Framing and Hypotheses (status report)

Anindya Banerjee (NSF – Alexandria, US) and David A. Naumann (Stevens Institute of Technology – Hoboken, US)

License © Creative Commons BY 3.0 Unported license
 © Anindya Banerjee and David A. Naumann
Joint work of Anindya Banerjee, David Naumann, Mohammad Nikouei

Relational properties arise in many settings: relating two versions of a program that use different data representations, noninterference properties for security, etc. The main ingredient of relational verification, relating aligned pairs of intermediate steps, has been used in numerous guises, but existing relational program logics are narrow in scope. We are investigating a logic based on novel syntax that weaves together product programs to express alignment of control flow points at which relational formulas are asserted. Correctness judgments feature hypotheses with relational specifications, discharged by a rule for the linking of procedure implementations. The logic supports reasoning about program-pairs containing both similar and dissimilar control and data structures. Reasoning about dynamically allocated objects is supported by a frame rule based on frame conditions amenable to SMT provers. In this talk we give an overview of the project ideas and status.

3.2 Semantic Differencing for HipHop Bytecode

Nick Benton (Facebook – London, GB)

License © Creative Commons BY 3.0 Unported license
 © Nick Benton

We describe a semantic differencing tool used to compare the bytecodes generated by two different compilers for Hack/PHP at Facebook. The tool is a prover for a simple relational Hoare logic for low-level code and is used in testing, allowing the developers to focus on semantically significant differences between the outputs of the two compilers.

3.3 Verification with Reusing Exchangeable Results (Conditions, Witnesses, Precisions)

Dirk Beyer (LMU München, DE)

License © Creative Commons BY 3.0 Unported license
 © Dirk Beyer

This presentation covers the topic of exchangeable verification results. First, we explain how the conditions of conditional model checking [1] can be used to pass information from one verifier to another, in particular, the first verifier describes in the condition the parts of the state space that it was able to successfully verify, while the second verifying can use the condition of the first verifier in order to concentrate on the state space that the first verifier did not succeed on [2]. Second, abstraction based approaches (cf. CEGAR) need to compute the abstract model, i.e., specify a precision that defines the level of abstraction (set of predicates for predicate abstraction, set of variables for value analysis). The precision


is a valuable piece of information that should be reused when verifying a similar program (as, e.g., in regression verification) [citePrecisionReuse]. Third, verification witnesses are exchangeable objects that contain information that another verification tool (validator) can use to re-establish the verification result [4, 5, 3]. Witnesses enable many new opportunities to improve the value of verification tools for the user, e.g., by supporting verification-based debugging [6].

References

- 1 D. Beyer, T. A. Henzinger, M. E. Keremoglu, and P. Wendler. 2012. Conditional Model Checking: A Technique to Pass Information between Verifiers. In *Proc. FSE*. ACM, Article 57, 57:1–57:11 pages. ISBN:978-1-4503-1614-9, <https://doi.org/10.1145/2393596.2393664>
- 2 Dirk Beyer, Marie-Christine Jakobs, Thomas Lemberger, and Heike Wehrheim. 2018. Reducer-Based Construction of Conditional Verifiers. In *Proc. ICSE*. ACM. <https://doi.org/10.1145/3180155.3180259>
- 3 D. Beyer and P. Wendler. 2013. Reuse of Verification Results: Conditional Model Checking, Precision Reuse, and Verification Witnesses. In *Proc. SPIN LNCS 7976*. Springer, 1–17. https://doi.org/10.1007/978-3-642-39176-7_1
- 4 D. Beyer, M. Dangel, D. Dietsch, M. Heizmann, and A. Stahlbauer. 2015. Witness Validation and Stepwise Testification across Software Verifiers. In *Proc. FSE*. ACM, 721–733. ISBN: 978-1-4503-3675-8. <https://doi.org/10.1145/2786805.2786867>
- 5 D. Beyer, M. Dangel, D. Dietsch, and M. Heizmann. 2016. Correctness Witnesses: Exchanging Verification Results Between Verifiers. In *Proc. FSE*. ACM, 326–337. <https://doi.org/10.1145/2950290.2950351>
- 6 Dirk Beyer and Matthias Dangel. 2016. Verification-Aided Debugging: An Interactive Web-Service for Exploring Error Witnesses. In *Proc. CAV (2) LNCS 9780*. Springer, 502–509. https://doi.org/10.1007/978-3-319-41540-6_28

3.4 Validating Optimizations of Concurrent C/C++ Programs

Soham Chakraborty (MPI-SWS – Kaiserslautern, DE)

License  Creative Commons BY 3.0 Unported license
© Soham Chakraborty

Compilation of C/C++ shared memory concurrent programs faces many challenges. On the one hand, C/C++ concurrency enable multiple transformations on shared memory accesses and fences. On the other hand, not all transformations which are correct for sequential programs are correct in the concurrent setting. Thus, compiler writers have to perform careful analysis to determine which transformations are correct.

In this talk I will present our work on validating the optimizations of LLVM, a state-of-the-art C/C++ compiler. Our work has revealed some previously unknown bugs in LLVM concerning the compilation of concurrent C/C++ programs.

3.5 Semantics-Parametric Program Equivalence

Stefan Ciobaca (University AI. I. Cuza – Iasi, RO)

License © Creative Commons BY 3.0 Unported license
© Stefan Ciobaca

Joint work of Stefan Ciobaca, Dorel Lucanu

The operational semantics of any programming language can be modeled as a set of constrained rewrite rules of the form “ l rewrites into r if b ”, where l and r are terms representing program configurations and where b is a logical constraint. The rewrite rules are interpreted in an algebra of program configurations and the reduction relation generated by the rewrite rules is the one-step transition relation.

Using this encoding, we can prove program equivalence in a semantics-parametric manner. We build an equivalence checker $E(P, Q, L, R)$ that takes as input not only two programs P and Q that we want to prove equivalent, but also the operational semantics L and R of the programming languages of P and Q .

We implement the equivalence checker and show that it works on several examples, which cover both imperative and functional languages. This is work-in-progress.

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI – UEFISCDI, project number PN-III-P2-2.1-BG-2016-0394, within PNCDI III.

3.6 Abstract Semantic Diffing of Evolving Concurrent Programs

Constantin Enea (University Paris-Diderot, FR)

License © Creative Commons BY 3.0 Unported license
© Constantin Enea

Joint work of Ahmed Bouajjani, Constantin Enea, Shuvendu K. Lahiri

Main reference Ahmed Bouajjani, Constantin Enea, Shuvendu K. Lahiri: “Abstract Semantic Diffing of Evolving Concurrent Programs”, in Proc. of the Static Analysis – 24th International Symposium, SAS 2017, New York, NY, USA, August 30 – September 1, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10422, pp. 46–65, Springer, 2017.

URL http://dx.doi.org/10.1007/978-3-319-66706-5_3

We present an approach for comparing two closely related concurrent programs, whose goal is to give feedback about interesting differences without relying on user-provided assertions. This approach compares two programs in terms of cross-thread interferences and data-flow, under a parametrized abstraction which can detect any difference in the limit. We introduce a partial order relation between these abstractions such that a program change that leads to a “smaller” abstraction is more likely to be regression-free from the perspective of concurrency. On the other hand, incomparable or bigger abstractions, which are an indication of introducing new, possibly undesired, behaviors, lead to succinct explanations of the semantic differences.

3.7 Property Directed Equivalence via Abstract Simulation

Grigory Fedyukovich (Princeton University, US)

License © Creative Commons BY 3.0 Unported license
© Grigory Fedyukovich

Joint work of Grigory Fedyukovich, Arie Gurfinkel, Natasha Sharygina

Main reference Grigory Fedyukovich, Arie Gurfinkel, Natasha Sharygina: “Property Directed Equivalence via Abstract Simulation”, in Proc. of the Computer Aided Verification – 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 9780, pp. 433–453, Springer, 2016.

URL http://dx.doi.org/10.1007/978-3-319-41540-6_24

Numerous versions of software have to be designed, developed, and verified before the product is ready for a release. Each version suffers from bugs which have to be fixed and should not appear again in the future. Once a version is formally verified for safety, its proof should be made available for verification of the coming versions. However, vast majority of verification tools are tailored to verification of each new program version in isolation from the version history. We present an approach for incremental verification based on constrained Horn clauses that lifts the proofs across program modifications. The key idea behind our approach is to establish a property directed equivalence between pairs of program versions, and we propose a way to do it through synthesis of simulation relations. We present the implementation and evaluation of the algorithm supporting our hypothesis that incremental verification can be performed efficiently, even if the program modifications are non-trivial. In cases when the complete proof lifting is impossible, our tool lifts the proof partially, which further allows the generation of the missing parts of the proof, or the calculation of a change impact certificate.

3.8 A graph-rewriting refinement of the β law

Dan R. Ghica (University of Birmingham, GB)

License © Creative Commons BY 3.0 Unported license
© Dan R. Ghica

Joint work of Dan R. Ghica, Koko Muroya, Todd Waugh Ambridge

The newly developed Dynamic Geometry of Interaction is a graph-rewriting abstract machine based on Girard’s semantics of linear logic proofs. It can model in a unified setting all the reduction strategies of the lambda calculus, also giving accurate cost models for execution. Using it we can refined the standard beta law of the lambda calculus into four, simpler, graph-rewriting laws.

3.9 Regression Verification of Multi-Threaded Programs

Arie Gurfinkel (University of Waterloo, CA) and Ofer Strichman (Technion – Haifa, IL)

- License** © Creative Commons BY 3.0 Unported license
© Arie Gurfinkel and Ofer Strichman
- Joint work of** Arie Gurfinkel, Sagar Chaki, Ofer Strichman
- Main reference** Sagar Chaki, Arie Gurfinkel, Ofer Strichman: “Regression verification for multi-threaded programs (with extensions to locks and dynamic thread creation)”, *Formal Methods in System Design*, Vol. 47(3), pp. 287–301, 2015.
- URL** <http://dx.doi.org/10.1007/s10703-015-0237-0>
- Main reference** Sagar Chaki, Arie Gurfinkel, Ofer Strichman: “Regression Verification for Multi-threaded Programs”, in *Proc. of the Verification, Model Checking, and Abstract Interpretation – 13th International Conference, VMCAI 2012, Philadelphia, PA, USA, January 22-24, 2012. Proceedings, Lecture Notes in Computer Science*, Vol. 7148, pp. 119–135, Springer, 2012.
- URL** http://dx.doi.org/10.1007/978-3-642-27940-9_9

Regression verification is the problem of deciding whether two similar programs are equivalent under an arbitrary yet equal context, given some definition of equivalence. So far this problem has only been studied for the case of single-threaded deterministic programs. We present a method for regression verification to establish partial equivalence (i.e., input/output equivalence of terminating executions) of multi-threaded programs. Specifically, we develop two proof-rules that decompose the regression verification between concurrent programs to that of regression verification between sequential functions, a more tractable problem. This ability to avoid composing threads altogether when discharging premises, in a fully automatic way and for general programs, uniquely distinguishes our proof rules from others used for classical verification of concurrent programs.

3.10 Model-checking contextual equivalence of higher-order programs with references

Guilhem Jaber (ENS – Lyon, FR)

- License** © Creative Commons BY 3.0 Unported license
© Guilhem Jaber

This talk will present SyTeCi, a general automated tool to check contextual equivalence for programs written in a typed higher-order language with references (i.e. local mutable states), corresponding to a fragment of OCaml. After introducing the notion of contextual equivalence, we will see on some examples why it is hard to prove such equivalences (reentrant calls, private states). Then, we will introduce SyTeCi, a tool to automatically check such equivalences. This tool is based on a reduction of the problem of contextual equivalence of two programs to the problem of reachability of “error states” in a transition system of memory configurations. Contextual equivalence being undecidable (even in a finitary setting), so does the non-reachability problem for such transition systems. However, one can apply model-checking techniques (predicate abstraction, analysis of pushdown systems) to check non-reachability via some approximations. This allows us to prove automatically many non-trivial examples of the literature, that could only be proved by hand before. We will end this talk by the presentation of a prototype implementing this work.

3.11 Distinguishing between Communicating Transactions

Vasileios Koutavas (Trinity College Dublin, IE)

License © Creative Commons BY 3.0 Unported license
© Vasileios Koutavas

Joint work of Vasileios Koutavas, Maciej Gazda, Matthew Hennessy

Main reference Vasileios Koutavas, Maciej Gazda, Matthew Hennessy: “Distinguishing between communicating transactions”, *Inf. Comput.*, Vol. 259(1), pp. 1–30, 2018.

URL <http://dx.doi.org/10.1016/j.ic.2017.12.001>

Communicating transactions is a form of distributed, non-isolated transactions which provides a simple construct for building concurrent systems. We will explore the observable behaviour of such systems through different nominal modal logics which share standard communication modalities, but have distinct past and future modalities involving transactional commits. We will discuss how, although quite different, the distinguishing power of these logics is identical. Furthermore, they are equally expressive because there are semantics-preserving translations between their formulae. Using the logics we can clearly exhibit subtle example inequivalences between communicating transactions, shedding light on the behaviour of such constructs.

3.12 DEEPSEC: Deciding Equivalence Properties in Security Protocols

Steve Kremer (INRIA Nancy – Grand Est, FR)

License © Creative Commons BY 3.0 Unported license
© Steve Kremer

Joint work of Vincent Cheval, Steve Kremer, Itsaka Rakotonirina

Main reference Vincent Cheval, Steve Kremer, Itsaka Rakotonirina: “DEEPSEC: Deciding Equivalence Properties in Security Protocols – Theory and Practice”. In *Proc. of the 39th IEEE Symposium on Security and Privacy (S&P’18)*, pp. 525–542, IEEE Computer Society Press, San Francisco, CA, USA, May 2018.

URL <http://doi.ieeecomputersociety.org/10.1109/SP.2018.00033>

Automated verification has become an essential part in the security evaluation of cryptographic protocols. Recently, there has been a considerable effort to lift the theory and tool support that existed for reachability properties to the more complex case of equivalence properties. In this talk I will report on our recent advances in theory and practice of this verification problem. We establish new complexity results for static equivalence, trace equivalence and labelled bisimilarity and provide a decision procedure for these equivalences in the case of a bounded number of sessions. Our procedure is the first to decide trace equivalence and labelled bisimilarity exactly for a large variety of cryptographic primitives—those that can be represented by a subterm convergent destructor rewrite system. We implemented the procedure in a new tool, DEEPSEC. We showed through extensive experiments that it is significantly more efficient than other similar tools, while at the same time raises the scope of the protocols that can be analysed.

3.13 Interprocedural Relational Verification in SymDiff and Applications

Shuvendu K. Lahiri (Microsoft Research – Redmond, US)

- License** © Creative Commons BY 3.0 Unported license
© Shuvendu K. Lahiri
- Main reference** Shuvendu K. Lahiri, Kenneth L. McMillan, Rahul Sharma, Chris Hawblitzel: “Differential assertion checking”, in Proc. of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE’13, Saint Petersburg, Russian Federation, August 18-26, 2013, pp. 345–355, ACM, 2013.
URL <http://dx.doi.org/10.1145/2491411.2491452>
- Main reference** Chris Hawblitzel, Ming Kawaguchi, Shuvendu K. Lahiri, Henrique Rebêlo: “Towards Modularly Comparing Programs Using Automated Theorem Provers”, in Proc. of the Automated Deduction – CADE-24 – 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings, Lecture Notes in Computer Science, Vol. 7898, pp. 282–299, Springer, 2013.
URL http://dx.doi.org/10.1007/978-3-642-38574-2_20
- Main reference** Shuvendu K. Lahiri, Chris Hawblitzel, Ming Kawaguchi, Henrique Rebêlo: “SYMDIFF: A Language-Agnostic Semantic Diff Tool for Imperative Programs”, in Proc. of the Computer Aided Verification – 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012. Proceedings, Lecture Notes in Computer Science, Vol. 7358, pp. 712–717, Springer, 2012.
URL http://dx.doi.org/10.1007/978-3-642-31424-7_54
- Main reference** Francesco Logozzo, Shuvendu K. Lahiri, Manuel Fähndrich, Sam Blackshear: “Verification modulo versions: towards usable verification”, in Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI ’14, Edinburgh, United Kingdom – June 09 – 11, 2014, pp. 294–304, ACM, 2014.
URL <http://dx.doi.org/10.1145/2594291.2594326>
- Main reference** Shaobo He, Shuvendu K. Lahiri, Zvonimir Rakamaric: “Verifying Relative Safety, Accuracy, and Termination for Program Approximations”, J. Autom. Reasoning, Vol. 60(1), pp. 23–42, 2018.
URL <http://dx.doi.org/10.1007/s10817-017-9421-9>

In this talk, I describe the SymDiff tool that is a verifier for proving properties of program differences. Differential program verification concerns with proving interesting properties over program differences, as opposed to the program itself. Such properties include program equivalence, but can also captures more general differential/relational properties. SymDiff provides a specification language to state such differential (two-program) properties using the concept of mutual summaries that can relate procedures from two versions. It also provides proof system for checking such differential specifications along with the capability of generating simple differential invariants.

We describe applications of SymDiff towards interprocedural equivalence checking, cross-version compiler validation, differential assertion checking, checking the safety of approximate transformations and for semantic change impact analysis.

3.14 Polymorphic Game Semantics for Dynamic Binding

James Laird (University of Bath, GB)


- License** © Creative Commons BY 3.0 Unported license
© James Laird
- Main reference** James Laird: “Polymorphic Game Semantics for Dynamic Binding”, in Proc. of the 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France, LIPIcs, Vol. 62, pp. 27:1–27:16, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.
URL <http://dx.doi.org/10.4230/LIPIcs.CSL.2016.27>

We present a game semantics for an expressive typing system for block-structured programs with late binding of variables and System F style polymorphism. As well as generic programs and abstract datatypes, this combination may be used to represent behaviour such as dynamic dispatch and method overriding.

We give a denotational models for a hierarchy of programming languages based on our typing system, including variants of PCF and Idealized Algol. These are obtained by extending polymorphic game semantics to block-structured programs. We show that the categorical structure of our models can be used to give a new interpretation of dynamic binding, and establish definability properties by imposing constraints which are identical or similar to those used to characterize definability in PCF (innocence, well-bracketing, determinacy). Moreover, relaxing these can similarly allow the interpretation of side-effects (state, control, non-determinism) – we show that in particular we may obtain a fully abstract semantics of polymorphic Idealized Algol with dynamic binding by following exactly the methodology employed in the simply-typed case.

3.15 Program equivalences and program refinements for compiler verification

Xavier Leroy (INRIA – Paris, FR)

License  Creative Commons BY 3.0 Unported license
© Xavier Leroy

Verifying the soundness of a compiler means proving that the generated code behaves as prescribed by the semantics of the source program. There are many definitions of interest for “behaves as prescribed”. Observational equivalence is appropriate for well-defined source languages such as Java. However, for C and C++, observational equivalence cannot be guaranteed because several evaluation orders are allowed for source programs, while the compiled code implements one of those evaluation orders. Moreover, C and C++ treat run-time errors such as integer division by zero or out-of-bound array accesses as undefined behaviors, meaning that the compiled code is allowed to perform any actions whatsoever, from aborting the program to continuing with random values to opening a security hole.

The CompCert compiler verification project builds on a notion of program refinement that enables the compiler to choose one among several possible evaluation orders, making the program “more deterministic”, and also to optimize source-level undefined behaviors away, making the program “more defined”. An example of the latter dimension of refinement is the elimination of an integer division $z = x / y$ if z is unused later: if y is 0, the original program exhibits undefined behavior (division by zero), but not the optimized program.

Program refinement is proved using simulation diagrams between the labeled transition systems that define the semantics of the original and transformed program. In full generality a so-called backward simulation diagram is needed, relating every transition of the transformed program with zero, one or several transitions of the original program, provided the original program is at a safe state (a state that cannot silently reach undefined behavior). For compilation passes that preserve the amount of nondeterminism, a simpler proof is possible as a forward simulation diagram, relating transitions of the original program with sequences of transitions of the transformed program.

This notion of program refinement and the associated proof techniques have served CompCert well so far, but can be difficult to extend to aggressive optimizations, other properties of interest, or other language features of interest. For example, loop optimizations such as loop exchange or loop blocking change control flow in a non-local manner, renewing interest in more denotational or more relational alternatives to simulation diagrams. Interesting program properties that we would like to see and to prove preserved during compilation

include constant-time cryptography, i.e. the fact that secret data is never used as argument to conditional branches, memory addressing, or other operations whose execution time depend on the value of the arguments. Finally, shared-memory concurrency is a challenge for compiler verification. Many compiler optimizations and code generation scheme that are valid for sequential programs remain valid for concurrent programs that are free of data races. Controlled data races, as supported by the low-level atomics of C and C++ 2011, raise many more challenges and are only starting to be understood semantically.

3.16 Client-Specific Equivalence Checking – An Overview

Yi Li (University of Toronto, CA) and Julia Rubin (University of British Columbia – Vancouver, CA)

License © Creative Commons BY 3.0 Unported license
© Yi Li and Julia Rubin

Joint work of Yi Li, Federico Mora, Marsha Chechik, Julia Rubin

Software is often built by integrating components created by different teams or even different organizations. Changes in one component may trigger a sequence of updates to its downstream clients. To avoid dealing with updates, developers often delay upgrades, negatively affecting correctness and robustness of their systems. In this work, we investigate the effect of component changes on the behaviour of their clients. We observe that changes in a component are often irrelevant to a particular client and thus can be adopted without any delays or negative effects. Following this observation, we formulate the notion of client-specific equivalence checking (CSE), lay out particular challenges and opportunities, and discuss possible solutions for checking such equivalence. We also present our early findings and propose some promising directions for further exploration.

3.17 Semantic Program Repair Using a Reference Implementation

Sergey Mechtaev (National University of Singapore, SG)

License © Creative Commons BY 3.0 Unported license
© Sergey Mechtaev

Joint work of Sergey Mechtaev, Manh-Dung Nguyen, Yannic Noller, Lars Grunske and Abhik Roychoudhury

Main reference Sergey Mechtaev, Manh-Dung Nguyen, Yannic Noller, Lars Grunske and Abhik Roychoudhury: “Semantic Program Repair Using a Reference Implementation” In Proc. of 40th Int’l Conference on Software Engineering, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE ’18)

URL <https://doi.org/10.1145/3180155.3180247>


The goal of program repair is to automatically modify a given incorrect program to eliminate the observable failures. One of the key challenges of this technology is that a formal specification of the intended behavior is typically not available in practice, and the use of a test suite as a correctness criteria often leads to the generation of incorrect patches that merely overfit the tests.

Semantic program repair aims to understand the meaning of software defects by means of semantic program analysis. This approach has two advantages over previous syntactic techniques. First, semantic analysis helps to efficiently navigate the conceptually large search space of patches. Second, semantic analysis helps to compensate the lack of correctness specification in real-world software.

In this talk, we discuss a semantic approach of generating program patches using a reference implementation. Specifically, this approach extracts specification from a reference implementation and generates a patch that enforces conditional equivalence of the patched and the reference programs w.r.t. a user-defined inputs condition. We demonstrate the effectiveness of this techniques in our experiments with GNU Coreutils and Busybox that implement the same set of UNIX utilities. We also discuss how this technique contributes to a broader vision of a general-purpose program repair system that will be able to address many types of defects in commodity software and have applications in software development, security and education.

3.18 An introduction to game semantics


Andrzej Murawski (University of Oxford, GB)

License  Creative Commons BY 3.0 Unported license
© Andrzej Murawski

I will give an introductory talk on game semantics, which is a modelling theory for higher-order programming languages based on the metaphor of game playing. Over the last 25 years, game semantics has been used to obtain the first full abstraction results for a wide spectrum of programming languages (full abstraction means that interpretations of two programs coincide exactly when the programs are equivalent). More recently, game models have been exploited to classify decidable (wrt program equivalence) fragments of various programming languages based on their type signatures. I will give a brief survey of the results and mention the kinds of automata that have turned out useful in capturing the dynamics of game models.

3.19 Trace Equivalence For Android Malware Detection

Julia Rubin (University of British Columbia – Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Julia Rubin

Joint work of Khaled Ahmed, Mieszko Lis, Julia Rubin

In this talk, we will present a novel approach we propose for efficiently detecting Android malware at runtime. Our approach relies on monitoring application execution, collecting execution traces, and then reasoning about equivalent vs. different traces. We will discuss characteristics of Android malware and identify a notion of trace equivalence that helps detect such malware. We will then show that the optimal notion of equivalence is impractical to implement due to the event-driven and multi-threaded nature of the Android system, and examine other possible solutions.

3.20 Proofs for Performance

Rahul Sharma (Microsoft Research India – Bangalore, IN)

License © Creative Commons BY 3.0 Unported license
© Rahul Sharma

Joint work of Eric Schkufza, Berkeley Churchill, Alex Aiken

Main reference Rahul Sharma, Eric Schkufza, Berkeley R. Churchill, Alex Aiken: “Data-driven equivalence checking”, in Proc. of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2013, part of SPLASH 2013, Indianapolis, IN, USA, October 26-31, 2013, pp. 391–406, ACM, 2013.

URL <http://dx.doi.org/10.1145/2509136.2509509>

Automated formal reasoning has the potential to significantly improve the quality of compiler generated code. We describe a data-driven approach to such reasoning: the proof steps are assisted by analysis applied to data gathered from program executions. We show how data-driven equivalence checking proves the correctness of code generated by production compilers (such as GCC with all optimizations enabled) by generating a formal proof of equivalence between a C source and the compiler generated x86 binary. Moreover, this equivalence checker lets us generate provably correct code that is up to 70% faster than the compiler generated code. Furthermore, we show how data-driven precondition inference lets us generate code that can be multiple times faster than compiler generated code.

3.21 Program equivalence problems in computational science

Stephen Siegel (University of Delaware – Newark, US)

License © Creative Commons BY 3.0 Unported license
© Stephen Siegel

Main reference Stephen F. Siegel, Manchun Zheng, Ziqing Luo, Timothy K. Zirkel, Andre V. Marianiello, John G. Edenhofner, Matthew B. Dwyer, Michael S. Rogers: “CIVL: the concurrency intermediate verification language”, in Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, November 15-20, 2015, pp. 61:1–61:12, ACM, 2015.

URL <http://dx.doi.org/10.1145/2807591.2807635>

A number of interesting program equivalence problems arise in computational science. Many algorithms used in that domain have a straightforward implementation—e.g., matrix multiplication. But these straightforward implementations are then transformed in innumerable ways, e.g., to reduce the number of floating-point operations, to use cache more efficiently, and to take advantage of parallel hardware. The transformed programs are expected to be equivalent – in some sense – to the original simple versions. In this talk I will describe several examples of such problems, and our use of symbolic execution tools (CIVL and TASS) to solve them. In many cases, equivalence can be established within some small bounds (on the sizes of inputs, number of processes, etc.), but some progress has been made on proofs without such bounds.

3.22 Proving Mutual Termination

Ofer Strichman (Technion – Haifa, IL)

License © Creative Commons BY 3.0 Unported license
© Ofer Strichman

Joint work of Dima Elenbogen, Shmuel Katz, Ofer Strichman

Main reference Dima Elenbogen, Shmuel Katz, Ofer Strichman: “Proving mutual termination”, Formal Methods in System Design, Vol. 47(2), pp. 204–229, 2015.

URL <http://dx.doi.org/10.1007/s10703-015-0234-3>

Two programs are said to be mutually terminating if they terminate on exactly the same inputs. We suggest inference rules and a proof system for proving mutual termination of a given pair of procedures $\langle f, f' \rangle$ and the respective subprograms that they call under a free context. Given a (possibly partial) mapping between the procedures of the two programs, the premise of the rule requires proving that given the same arbitrary input in , $f(in)$ and $f'(in)$ call procedures mapped in the mapping with the same arguments. A variant of this proof rule with a weaker premise allows to prove termination of one of the programs if the other is known to terminate. In addition, we suggest various techniques for battling the inherent incompleteness of our solution, including a case in which partial equivalence (the equivalence of their input/output behavior) has only been proven for some, but not all, the outputs of the two given procedures. We present an algorithm for decomposing the verification problem of whole programs to that of proving mutual termination of individual procedures, based on our suggested inference rules. In this talk I will survey our work on proving mutual termination of programs and demo our prototype implementation of this algorithm.

3.23 The Software Analysis Workbench

Aaron Tomb (Galois – Portland, US)

License © Creative Commons BY 3.0 Unported license
© Aaron Tomb

Joint work of Robert Dockins, Adam Foltzer, Joe Hendrix, Brian Huffman, Dylan McNamee, Aaron Tomb

Main reference Robert Dockins, Adam Foltzer, Joe Hendrix, Brian Huffman, Dylan McNamee, Aaron Tomb: “Constructing Semantic Models of Programs with the Software Analysis Workbench”, in Proc. of the Verified Software. Theories, Tools, and Experiments – 8th International Conference, VSTTE 2016, Toronto, ON, Canada, July 17-18, 2016, Revised Selected Papers, Lecture Notes in Computer Science, Vol. 9971, pp. 56–72, 2016.

URL http://dx.doi.org/10.1007/978-3-319-48869-1_5

The Software Analysis Workbench (SAW) is a tool for transforming programs into models of their functional behavior, manipulating those models, and using various third-party reasoning tools to prove properties of those models. Although it is in principle more than a program equivalence checking tool, it is most highly tuned for proving equivalence.

SAW currently uses symbolic execution to construct program models, and uses path merging and path satisfiability checking to increase the class of programs for which it can generate a single, complete model. As a result, the current behavior is roughly an instance of bounded model checking, with the bounds provided by the program rather than some fixed constant.

SAW integrates closely with Cryptol, a domain-specific functional language originally designed for the high-level description of cryptographic algorithms, and generally well-suited to describing finite programs of the sort that are most amenable to analysis with SAT

and SMT. We have used SAW to prove functional equivalence between many imperative implementations of cryptographic algorithms and high-level specifications written in Cryptol.

This talk describes some of the techniques used in SAW along with some examples of the concrete implementations we have used it to verify.

3.24 Nominal Games: A Semantics Paradigm for Effectful Languages

Nikos Tzevelekos (Queen Mary University of London, GB)

License © Creative Commons BY 3.0 Unported license
© Nikos Tzevelekos

Joint work of Andrzej Murawski, Steven Ramsay, Dan Ghica, Guilhem Jaber, Thomas Cuvillier, Nikos Tzevelekos

Game semantics has been developed since the 90's as a denotational paradigm capturing observational equivalence of functional languages with imperative features. While initially introduced for PCF variants, the theory can nowadays express effectful languages ranging from ML fragments and Java programs to C-like code. In this talk we present recent advances in devising game models for effectful computation. Central in this approach is the use of names for representing in an abstract fashion different forms of notions and effects, such as references, higher-order values and polymorphism. We moreover look at automata models relevant to nominal games and how can they be used for model checking program equivalence.

3.25 Relational Equivalence Proofs Between Imperative and MapReduce Algorithms

Mattias Ulbrich (KIT – Karlsruher Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license
© Mattias Ulbrich

Main reference Bernhard Beckert, Timo Bingmann, Moritz Kiefer, Peter Sanders, Mattias Ulbrich, Alexander Weigl: “Relational Equivalence Proofs Between Imperative and MapReduce Algorithms”, CoRR, Vol. abs/1801.08766, 2018.

URL <http://arxiv.org/abs/1801.08766>

MapReduce frameworks are widely used for the implementation of distributed algorithms. However, translating imperative algorithms into these frameworks requires significant structural changes to the algorithm. As the costs of running faulty algorithms at scale can be severe, it is highly desirable to verify the correctness of the translation, i.e., to prove that the MapReduce version is equivalent to the imperative original. We present a novel approach for proving equivalence between imperative and MapReduce algorithms based on partitioning the equivalence proof into a sequence of equivalence proofs between intermediate programs with smaller differences. Our approach is based on the insight that two kinds of sub-proofs are required: (1) uniform transformations rewriting the controlflow structure that are mostly independent of the particular context in which they are applied; and (2) context-dependent transformations that are not uniform but that preserve the overall structure and can be proved correct using coupling invariants. I demonstrated the feasibility of our approach by applying it to the PageRank algorithm. The potential for automation has been discussed.

3.26 A Behavioural Equivalence for Algebraic Effects: Logic with Modalities

Niels Voorneveld (University of Ljubljana, SI)

License  Creative Commons BY 3.0 Unported license
© Niels Voorneveld

Joint work of Alex Simpson, Niels F. W. Voorneveld

Main reference Alex Simpson, Niels F. W. Voorneveld: “Behavioural Equivalence via Modalities for Algebraic Effects”, in Proc. of the Programming Languages and Systems – 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14–20, 2018, Proceedings, Lecture Notes in Computer Science, Vol. 10801, pp. 300–326, Springer, 2018.

URL http://dx.doi.org/10.1007/978-3-319-89884-1_11

In this talk we investigate behavioural equivalence between programs of a functional language extended with a signature of (algebraic) effect-triggering operations. Two programs are considered behaviourally equivalent if they enjoy the same behavioural properties. To formulate this, we define a logic whose formulas specify these behavioural properties. A crucial ingredient is a collection of modalities expressing effect-specific aspects of behaviour. The construction of the logic and the theory of such modalities are outlined.

We look at examples of effects and what modalities we may choose for them. These examples include: nondeterminism, error, probabilistic choice, global store and input/output. Moreover, we will briefly look at two technical conditions for such modalities which, if satisfied, makes the logically-specified behavioural equivalence a congruence. The given examples satisfy these properties. The induced behavioural equivalence is also related to a notion of Abramsky’s applicative bisimilarity.

Participants

- Anindya Banerjee
NSF – Alexandria, US
- Gilles Barthe
IMDEA Software – Madrid, ES
- Nick Benton
Facebook – London, GB
- Dirk Beyer
LMU München, DE
- Soham Chakraborty
MPI-SWS – Kaiserslautern, DE
- Stefan Ciobaca
University Al. I. Cuza – Iasi, RO
- Constantin Enea
University Paris-Diderot, FR
- Grigory Fedyukovich
Princeton University, US
- Dan R. Ghica
University of Birmingham, GB
- Arie Gurfinkel
University of Waterloo, CA
- Guilhem Jaber
ENS – Lyon, FR
- Vasileios Koutavas
Trinity College Dublin, IE
- Steve Kremer
INRIA Nancy – Grand Est, FR
- Shuvendu K. Lahiri
Microsoft Research – Redmond, US
- James Laird
University of Bath, GB
- Xavier Leroy
INRIA – Paris, FR
- Yi Li
University of Toronto, CA
- Sergey Mechtaev
National University of Singapore, SG
- Andrzej Murawski
University of Oxford, GB
- Kedar Namjoshi
Nokia Bell Labs – Murray Hill, US
- David A. Naumann
Stevens Institute of Technology – Hoboken, US
- Julia Rubin
University of British Columbia – Vancouver, CA
- Philipp Rümmer
Uppsala University, SE
- Neha Rungta
Amazon.com, Inc. – Palo Alto, US
- Chaked Saydoff
Technion – Haifa, IL
- Rahul Sharma
Microsoft Research India – Bangalore, IN
- Stephen Siegel
University of Delaware – Newark, US
- Marcelo Sousa
University of Oxford, GB
- Ofer Strichman
Technion – Haifa, IL
- Aaron Tomb
Galois – Portland, US
- Nikos Tzevelekos
Queen Mary University of London, GB
- Mattias Ulbrich
KIT – Karlsruher Institut für Technologie, DE
- Niels Voorneveld
University of Ljubljana, SI



Blockchains, Smart Contracts and Future Applications

Edited by

Foteini Baldimtsi¹, Stefan Katzenbeisser², Volkmar Lotz³, and
Edgar Weippl⁴

1 George Mason University – Fairfax, US, foteini@gmu.edu

2 TU Darmstadt, DE, skatzenbeisser@acm.org

3 SAP Labs France – Mougins, FR, volkmar.lotz@sap.com

4 Secure Business Austria Research, AT, eweippl@sba-research.org

Abstract

This report documents the Dagstuhl seminar 18152 “Blockchains, Smart Contracts & Future Applications”. While Bitcoin currently works well in practice, there are many open questions regarding the long-term perspective of blockchain technologies, for both public and private/permissioned blockchains. It is yet unclear how processes can be designed to work in predictive ways and how to embed security in the lifecycle of smart contract development and deployment. Furthermore, the distributed nature of the system needs to be considered when thinking about which groups or individuals can influence future developments. Similar to ‘real-world’ societies, blockchains are based on mutual recognition of conventions. Diverse academic disciplines as well as industry can and need to collaborate to advance research in blockchain and to fully understand how the technology might impact our future lives.

Seminar April 8–13, 2018 – <http://www.dagstuhl.de/18152>

2012 ACM Subject Classification Theory of computation → Cryptographic protocols, Networks
→ Peer-to-peer protocols

Keywords and phrases blockchains, consensus algorithms, cryptographic currency, incentive engineering, smart contracts

Digital Object Identifier 10.4230/DagRep.8.4.20

Edited in cooperation with Nicholas Stifter and Philipp Schindler


1 Executive Summary

Edgar Weippl (Secure Business Austria Research, AT)

Foteini Baldimtsi (George Mason University – Fairfax, US)

Stefan Katzenbeisser (TU Darmstadt, DE)

Volkmar Lotz (SAP Labs France – Mougins, FR)

License  Creative Commons BY 3.0 Unported license

© Edgar Weippl, Foteini Baldimtsi, Stefan Katzenbeisser, and Volkmar Lotz

In its beginnings, the technical and socio-economical feasibility of Bitcoin was met with much skepticism; however, this has since changed as both research and practice have outlined the merits of distributed ledger technologies, commonly referred to as “blockchains”. Possible applications of blockchains reach from decentralized settlement layers over complex smart contract systems to tailored authenticated data structures that implement systems for identity or supply chain management. Nevertheless, beyond the immediate opportunities and applications lie many open questions regarding the long-term perspective of both permissionless and permissioned blockchain technologies. For example, while scalability and



Except where otherwise noted, content of this report is licensed
under a Creative Commons BY 3.0 Unported license

Blockchains, Smart Contracts and Future Applications, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 20–31

Editors: Foteini Baldimtsi, Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

sustainability are currently topics of active research and development, other aspects such as usability, interoperability and cryptoeconomics have received considerably less attention. In order to anticipate and address future key topics and questions related to blockchain technologies, this seminar strove to provide an interdisciplinary breeding ground.

The participants focused on future applications and developments of this technology and discussed how such complex systems can thrive over a long period of time. Thereby, we started our seminar by outlining and collecting current and potentially future issues from the diverse viewpoints of the participants. These issues include not only current limitations of the underlying technologies, but also problems encountered in real-world applications.

As an example, we considered the various economic, legal and technological uncertainties and problems that have arisen as a consequence of the recent contentious forks in both the Bitcoin (August 2017) and Ethereum (July 2016) networks. While the possibility of such forks was previously well known, it can be argued that provisionary measures and research on effectively dealing with them was immature and could have been addressed much sooner. In any case, the ramifications of these events have and will continue to influence the discussion and development of blockchain technologies.

Beside establishing the relevant issues through numerous talks, subgroups of participants were formed to discuss a specific set of topics. Over the course of the seminar, participants were encouraged to move between groups and provide input to various topics. We hope to have thus enriched the discussion with different viewpoints and to have facilitated a rewarding range of outcomes; at the point of writing, two papers directly resulting from this Dagstuhl seminar are submitted for review. The goal of the seminar was to develop a shared and open agenda that shapes and directs research and development in the area of distributed ledger technologies to face current and future challenges as well as contribute to the positive development of this field.

The talks and working groups of this first Dagstuhl seminar on Blockchains, Smart Contracts and their future applications focused inter alia on the following topics:

- current and future protocols, including alternative consensus protocols
- governance
- interdisciplinary aspects of Blockchain technology (economy, law)
- cross-chain communication
- scalability and costs
- Goldfinger and other attack vectors

2 Table of Contents

Executive Summary

<i>Edgar Weippl, Foteini Baldimtsi, Stefan Katzenbeisser, and Volkmar Lotz</i>	20
--	----

Overview of Talks

How to Charge Lightning <i>Zohar Aviv</i>	23
Blockchains are from Mars, TEEs are from Intel. An overview of blockchain and Trusted Execution Environment combination <i>Ittay Eyal</i>	23
Perun: virtual payment and state channel networks <i>Sebastian Faust</i>	24
Ouroboros Proof-of-Stake Protocols <i>Peter Gazi</i>	24
Cryptocurrency Analytics – An Agenda for (some more) Interdisciplinary Research <i>Bernhard Haslhofer</i>	25
Goldfinger’s Technical Possibilities – Open Questions for Cross-Chain Interlinking <i>Aljosha Judmayer</i>	25
Anonymity in Cryptocurrencies <i>Sarah Meiklejohn</i>	26
Egalitarian Society or Benevolent Dictatorship: The State of Cryptocurrency Governance <i>Sarah Meiklejohn</i>	26
On the Necessity of a Prescribed Block Validity Consensus: Analyzing the Bitcoin Unlimited Mining Protocol <i>Bart Preneel</i>	26
Randomness for Blockchains <i>Philipp Schindler</i>	27
Selected Legal Aspects of Blockchain Technology Applications <i>Sofie Schock</i>	27
Open Questions in Blockchain Consensus <i>Nicholas Stifter</i>	28
Towards a Generalised Blockchain Fabric <i>Alexei Zamyatin</i>	28

Working groups

Futurology <i>Philipp Schindler and Nicholas Stifter</i>	29
DAG <i>Nicholas Stifter</i>	29
Governance <i>Nicholas Stifter and Philipp Schindler</i>	30

Participants	31
------------------------	----

3 Overview of Talks

3.1 How to Charge Lightning

Zohar Aviv (Hebrew University – Jerusalem, IL)

License © Creative Commons BY 3.0 Unported license
© Zohar Aviv

On-chain transaction channels represent one of the leading techniques to scale the transaction throughput in cryptocurrencies. However, until now the economic effect of transaction channels on the system has not been widely explored. We studied the economics of Bitcoin transaction channels and presented a framework for an economic analysis of the lightning network and its impact on transaction fees in the blockchain. Our framework allows us to reason about different patterns of demand for transactions and different topologies of the lightning network, and to derive the resulting fees for transacting both on and off the blockchain. Our initial results indicate that while the lightning network does allow for a substantially higher number of transactions to pass through the system, it does not necessarily provide higher fees to miners and, as a result, may in fact lead to lower participation in mining within the system.

3.2 Blockchains are from Mars, TEEs are from Intel. An overview of blockchain and Trusted Execution Environment combination

Ittay Eyal (Technion – Haifa, IL)

License © Creative Commons BY 3.0 Unported license
© Ittay Eyal

In this talk we tried to answer two key questions:

- How can TEEs extend blockchains?
- How can blockchains extend TEE-based distributed algorithms?

Thereby, we explored two particular examples:

TEEs for PoW alternatives:

- Proof of Elapsed Time (PoET): CPU waits an exponentially distributed random time rather than wasting work; however, it wastes cupex (in hardware) instead of software – old HW mines as efficient as new HW.
- Proof of useful Works (Zhang et al. 2017): perform useful work for mining
 - Useful work CPU instructions counted as puzzle solution attempts, enforced by TEE
 - Automatic instrumentation for correct instruction counting and reporting
 - Hierarchical attestation with compliance checker

Efficient and asynchronous Blockchain access payment channels by combining blockchain and TEE powers: TEEchain.

$$TEE_A - \mathbf{A-B} - TEE_B \tag{1}$$

Each party's TEE maintains the party's currency, guaranteeing to settle it on the blockchain exactly (at most) once.

The challenges are:

- Form channels between the TEEs
- Enable arbitrary work (if hierarchical attestation)
- TEE crash-fault tolerance
- Multihop payments without synchronous blockchain access

3.3 Perun: virtual payment and state channel networks


Sebastian Faust (TU Darmstadt, DE)

License  Creative Commons BY 3.0 Unported license
© Sebastian Faust

One of the main challenges that hinder further adaption of cryptocurrencies is scalability. Because cryptocurrencies require that all transactions are processed and stored on the blockchain transaction, throughput is inherently limited. An important proposal to significantly improve this are off-chain protocols, where the massive bulk of transactions is executed without requiring the costly interaction with the blockchain. In this talk we introduce Perun – a network of virtual payment and state channels. The main contributions of our work are introducing the concept of virtual channels, and providing the first full specification of arbitrary complex state channel networks. The latter allows users to execute smart contracts in an off-chain way. All our constructions are analysed using the universal composability framework commonly used in cryptography for analysing cryptographic protocols.

3.4 Ouroboros Proof-of-Stake Protocols

Peter Gazi (IOHK – Hong Kong, HK)

License  Creative Commons BY 3.0 Unported license
© Peter Gazi

Bitcoin and most other existing cryptocurrencies use the so-called Proof-of-Work approach to extend the blockchain: If a party wants to create a new block, they have to solve a computation-intensive puzzle and only once they succeed they are allowed to attach a new block to the chain (containing proof that this considerable amount of work has been invested). This leads to an arms race between miners to invest even more computational power (and, hence, electricity) into solving these puzzles, leading to an already worrying level of energy consumption by Bitcoin. Even worse, this energy requirements scale with the size of the system, so the more mainstream Bitcoin becomes, the more energy will be consumed to maintain its security.

Alternatively, Proof-of-Stake protocols use a different approach to decide about the eligibility of parties to create new blocks. Namely, the probability of each party to “win a lottery” and be allowed to create a new block (in a given time interval) is, by the design of the protocol, proportional to the amount of stake (i.e., coins) owned by that party, as recorded by the ledger itself. Evaluating this lottery can be done very easily and without any extensive computation, thus relieving the system from basing its security on a continuous waste of resources. This seemingly simple idea (which is almost as old as Bitcoin itself), however, turns out to be difficult to implement securely, which is why getting a provably secure Proof-of-Stake protocol is so important and has eluded the community for quite some time.

In this talk, I presented the Ouroboros family of provably secure Proof-of-Stake protocols as well as some exciting ongoing work and open questions in the area.

3.5 Cryptocurrency Analytics – An Agenda for (some more) Interdisciplinary Research

Bernhard Haslhofer (AIT Austrian Institute of Technology – Wien, AT)

License © Creative Commons BY 3.0 Unported license
© Bernhard Haslhofer

From an abstract, birds-eye perspective, cryptocurrencies can be perceived as a network in which different types of actors (e.g., exchanges, darknet marketplaces, payment providers) interact with each other through transaction. The goal of cryptocurrency analytics is to investigate and develop scalable quantitative methods, tools and services that contribute to a better understanding of the structure and dynamics of cryptocurrency ecosystems. One can distinguish between two types of analytics tasks: microscopic analysis focusing on the traceability of transaction chains and macroscopic analysis focusing on the investigation of the entire ecosystem after projecting real-world phenomena, such as ransomware attacks, onto the network.

The goal of this talk was to show how cryptocurrency analytics methods can be used in a number of application scenarios, ranging from science to public authorities to the FinTech sector. As a concrete example, this talk presented the results of a recent macroscopic study that analysed and qualified ransomware payments in the Bitcoin ecosystem. Finally, it outlined open application-oriented research questions, structured by the main technical ingredients that enable cryptocurrency analytics tasks: algorithms and heuristics, attribution data, and computation platforms.

3.6 Goldfinger's Technical Possibilities – Open Questions for Cross-Chain Interlinking

Aljosha Judmayer (SBA Research – Wien, AT)


License © Creative Commons BY 3.0 Unported license
© Aljosha Judmayer

Goldfinger attacks, initially described by Kroll et al. in 2013, aim to damage the economy of a cryptocurrency such that the attacker achieves utility outside of the very same cryptocurrency. Until lately, this type of attack has not received much research attention, but the fast-growing number of cryptocurrencies has made such kind of scenarios more plausible, as also outlined by Bonneau (2017; 2018).

This talk surveys the literature on Goldfinger attacks and bribing techniques in the area of cryptocurrencies to extent upon existing methods. Thereby, also new directions for attacks are proposed which utilize merged mining as an attack vector. The goal is to show that smart contracts and cross-chain interlinking of different cryptocurrencies are also enabling technologies to perform more attacks than currently envisioned. This leads to the open question if and how the threat model of permissionless cryptocurrencies needs to be adjusted to better account for such kinds of attacks.

3.7 Anonymity in Cryptocurrencies


Sarah Meiklejohn (University College London, GB)

License  Creative Commons BY 3.0 Unported license
© Sarah Meiklejohn

A long line of recent research has demonstrated that existing cryptocurrencies often do not achieve the level of anonymity that users might expect they do, while at the same time another line of research has worked to increase the level of anonymity by adding new features to existing cryptocurrencies or creating entirely new ones. This talk will explore both de-anonymization attacks and techniques for anonymity that achieve provably secure guarantees.

3.8 Egalitarian Society or Benevolent Dictatorship: The State of Cryptocurrency Governance

Sarah Meiklejohn (University College London, GB)


License  Creative Commons BY 3.0 Unported license
© Sarah Meiklejohn

We initiated a quantitative study of the decentralization of the governance structures of Bitcoin and Ethereum. In particular, we scraped the open-source repositories associated with their respective codebases and improvement proposals to find the number of people contributing to the code itself and to the overall discussion.

We then present different metrics to quantify decentralization, both in each of the cryptocurrencies and, for comparison, in two popular open-source programming languages, Clojure and Rust. We find that for both cryptocurrencies and programming languages, there is usually a handful of people that accounts for most of the discussion. We also look into the effect of forks in Bitcoin and Ethereum, and find that there is little intersection between the communities of the original currencies and those of the forks.

3.9 On the Necessity of a Prescribed Block Validity Consensus: Analyzing the Bitcoin Unlimited Mining Protocol

Bart Preneel (KU Leuven, BE)

License  Creative Commons BY 3.0 Unported license
© Bart Preneel

Bitcoin has attracted many users, but also has been considered as a technical breakthrough by academia. However, the potential of Bitcoin is largely untapped due to its limited throughput. The Bitcoin community is currently facing its biggest crisis in history, since the community disagrees on how to increase the throughput. Among various protocols, Bitcoin Unlimited recently became the most popular candidate, as it allows miners to collectively decide the block size limit according to the rest network capacity. However, the security of Bitcoin Unlimited is heavily debated and no consensus has been reached as the issue is discussed under different mining incentive models. We systematically tested Bitcoin Unlimited's security using three incentive models; we evaluated the two major arguments of

Bitcoin Unlimited’s security: block validity consensus is not necessary for Bitcoin Unlimited’s security as such consensus would emerge in Bitcoin Unlimited based on economic incentives. Our results invalidate both arguments and therefore disprove Bitcoin Unlimited’s security chains. We also discuss whether a prescribed block validity consensus is a necessary feature of a cryptocurrency.

3.10 Randomness for Blockchains

Philipp Schindler (SBA Research – Wien, AT)

License © Creative Commons BY 3.0 Unported license
© Philipp Schindler

S. Nakamoto proposed the first practical solution for the problem of reaching consensus in a dynamic set of potentially anonymous participants without a prior agreement on this set. Bitcoin achieves this advancement at the cost of high computational requirements for Proof-of-Work, leading to vast amounts of electricity being consumed.

Recently, new protocols – using Proof-of-Stake as a fundamental principle – tried to improve upon Nakamoto’s solution. These protocols require a trustworthy source of randomness to maintain desirable security guarantees. However, obtaining trustworthy randomness in a highly decentralized network and under potentially adversarial conditions is by itself a challenging task. Recent academic research as well as projects from the industry try to address this problem by designing random beacon protocols which produce the required random values in regular intervals. We highlight the design challenges of random beacon protocols as well as provide a review and comparison of state-of-the-art protocols.

3.11 Selected Legal Aspects of Blockchain Technology Applications

Sofie Schock (Universität Wien, AT)

License © Creative Commons BY 3.0 Unported license
© Sofie Schock

From a legal perspective, Distributed Ledger Technologies like the Blockchain technology, raise many fundamental questions. Especially in permissionless Blockchain networks, where no “organiser” exists and where the participants act pseudonymously, existing legal tools reach their limits. It turned out that it is not possible to relate liability in such networks in current legal systems. Although proposals for possible solutions of this problem have been made, it is still a long way to a profound and reasonable legal work up. In this context, it is very important for lawyers to understand the technical principles; therefore, a close collaboration between lawyers and technicians is a desired goal.

3.12 Open Questions in Blockchain Consensus

Nicholas Stifter (TU Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Nicholas Stifter

The rise of Bitcoin and its underlying blockchain technology has revitalized the discussion on distributed consensus and connected various scientific disciplines in their quest for reaching a deeper understanding of the characteristics of the Nakamoto consensus. Nevertheless, while research on this topic has led to many valuable insights and advancements in regard to Byzantine consensus protocols, it also raises new fundamental questions:

The sustainability of relying on Proof-of-Work in blockchain consensus is increasingly becoming an issue, while the characteristics and trade-offs of potential alternatives such as Proof-of-Stake or Proof-of-X are still not entirely clear. Apparent miner centralization questions the aspiration of many cryptocurrencies to be decentralized, while incurring scalability difficulties because of this property. Newly proposed consensus protocols for an application in distributed ledgers should strive for simplicity and could benefit from more concise definitions of the requirements and characteristics that they need to satisfy. Introducing game theory to model the behavior of consensus participants, and the formal characteristics and guarantees that can be derived of such assumptions in consensus protocols, is an exciting research direction that has, so far, received relatively little attention. Finally, incentivizing or possibly enforcing consensus participation in decentralized systems, where protocols can be readily modified to create concurrent and competing systems, remains an open question.

3.13 Towards a Generalised Blockchain Fabric

Alexei Zamyatin (Imperial College London, GB)

License  Creative Commons BY 3.0 Unported license
© Alexei Zamyatin

Since the introduction of Bitcoin in 2008, the field of cryptocurrencies has gained popularity in both academic and private sectors. Today, there exist over 1500 cryptocurrencies and new systems are being launched on a daily basis. However, while most blockchain-based digital currencies are of a decentralized nature, secure asset and information exchange between such systems currently requires a trusted third party, e.g., a centrally banked exchange. Throughout the past years, research into facilitating trustless cross-chain communication has resulted in the proposal of numerous concepts and mechanisms. However, to this date, scientific publications are scarce and only a limited number of introduced concepts has been implemented. In this work, we attempt to provide a taxonomy of relevant properties for cross-chain communication, a categorization of existing protocols, and an overview of current challenges hindering the deployment of such schemes.

4 Working groups

4.1 Futurology

Philipp Schindler (SBA Research – Wien, AT) and Nicholas Stifter (TU Wien, AT)

License © Creative Commons BY 3.0 Unported license
© Philipp Schindler and Nicholas Stifter

Predictions about technological advancements and their impact on societies are often largely inaccurate, in particular when made over a larger time span. Nevertheless, anticipating future developments of cryptocurrencies and distributed ledger technologies can help hone in on the core aspects that make up the disruptive potential of these new technologies. The participants of this breakout session engaged in thought experiments on the future impact of blockchain technologies, outlining four different possible scenarios roughly twelve years from now. Hereby, two utopian and two dystopian futures in which the technology prevails or fails respectively, were envisioned. In particular the topic of privacy was a recurring theme, either in the form of an Orwellian nightmare in which total transparency has eroded social cohesion and condensed power among a privileged elite, or in a scenario where the demand and awareness for privacy-enhancing technologies in DLT has led to a broader understanding and ability to effectively and safely use such technologies for the greater good of society. Further, the discussions among participants outlined that the considered utopian and dystopian scenarios lie closely together, whereby the failure of a few key assumptions such as the security of elliptic curve cryptography could quickly turn a positive into a negative outcome.

4.2 DAG


Nicholas Stifter (TU Wien, AT)

License © Creative Commons BY 3.0 Unported license
© Nicholas Stifter

This breakout session was focused on the topic of directed acyclic graphs (DAGs) in blockchain protocol designs. Participants were first able to enjoy an introductory course or refresher on the basic concepts and research challenges of DAG-based blockchain designs that was presented by Aviv Zohar. The speaker then further outlined the Spectre and Phantom DAG protocols and gave insights into their development process, as well as the challenges that were faced when ensuring that the designs could provide the desired security, consistency and liveness properties. The group then compared and explored different characteristics of various DAG-based protocols and design proposals such as Spectre, Phantom, Hashgraph, Fruitchains, GHOST, IOTA or Braids in more detail, focusing on the properties and guarantees these protocols can achieve or intend to provide and how they differ from each other. Participants finally engaged in a vivid discussion on security aspects of already deployed DAG protocols and that new projects emerging from the cryptocurrency community often lack rigorous formal analysis and proofs of their underlying concepts.

4.3 Governance

Nicholas Stifter (TU Wien, AT) and Philipp Schindler (SBA Research – Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Nicholas Stifter and Philipp Schindler

The goal of this breakout session was to identify and discuss relevant issues and open research questions on the topic of governance for cryptocurrencies and Blockchain/Distributed Ledger Technologies (DLT). As an initial step the question of why governance is needed in the first place was addressed by the participants, assuming that present cryptocurrency and DLT implementations may already violate existing norms and fundamental rights. In this context difficulties arise not only because norms are not clearly defined and are subject to change, rendering them hard to formalize, but also because the underlying technological model of many DLT that facilitates open access with weak identities renders the enforcement of such norms hard or impossible. One informal argument the group gave why governance is needed was: (to) *“modify the system in order to: adjust to unpredicted changes in the environment including norms, specifically about the redistribution of wealth/happiness between the users and people affected by (the existence of) the system.”* An important distinction was made between governance questions arising from outside the decentralized system, such as national and international laws and regulations, and governance issues related to the system itself as part of the protocol or operational procedures. Further, different control points and mechanisms for governance were discussed, from which the central observation was made that (software) forks can be employed as an expression of dissent and that the forking mechanism as a governance primitive should receive further study and attention. Finally, the group engaged in the topic of how to effectively approach and study governance questions for DLT and cryptocurrencies. Existing governance models and processes in other open source projects, but also from non-technical systems such as the United Nations or European Union, may provide valuable insights and experience while research on topics such as political communication theories, computational social choice theory or coordination games could help shape a systematic approach.

Participants

- Zohar Aviv
Hebrew University –
Jerusalem, IL
- Foteini Baldimtsi
George Mason University –
Fairfax, US
- Alex Biryukov
University of Luxembourg, LU
- Rainer Böhme
Universität Innsbruck, AT
- Jan Camenisch
IBM Research-Zurich, CH
- Samuel Christie
North Carolina State University –
Raleigh, US
- Ittay Eyal
Technion – Haifa, IL
- Sebastian Faust
TU Darmstadt, DE
- Peter Gazi
IOHK – Hong Kong, HK
- Dieter Gollmann
TU Hamburg, DE
- Raimund Gross
SAP SE – Walldorf, DE
- Bernhard Haslhofer
AIT Austrian Institute of
Technology – Wien, AT
- Aljosha Judmayer
SBA Research – Wien, AT
- Stefan Katzenbeisser
TU Darmstadt, DE
- Kwok-Yan Lam
Nanyang TU – Singapore, SG
- Juho Lindman
University of Gothenburg, SE
- Volkmar Lotz
SAP Labs France – Mougins, FR
- Sarah Meiklejohn
University College London, GB
- Bart Preneel
KU Leuven, BE
- Alessandra Scafuro
North Carolina State University –
Raleigh, US
- Philipp Schindler
SBA Research – Wien, AT
- Sofie Schock
Universität Wien, AT
- Nicholas Stifter
TU Wien, AT
- Thorsten Strufe
TU Dresden, DE
- Edgar Weippl
Secure Business Austria
Research, AT
- Alexei Zamyatin
Imperial College London, GB



Visualization of Biological Data – Crossroads

Edited by

Jan Aerts¹, Nils Gehlenborg², Georgeta Elisabeta Marai³, and
Kay Katja Nieselt⁴

1 KU Leuven, BE, jan.aerts@kuleuven.be

2 Harvard University, US, nils@hms.harvard.edu

3 University of Illinois – Chicago, US, gmarai@uic.edu

4 Universität Tübingen, DE, kay.nieselt@uni-tuebingen.de

Abstract

Our ability to generate and collect biological data has accelerated significantly in the past two decades. In response, many novel computational and statistical analysis techniques have been developed to process and integrate biological data sets. However, in addition to computational and statistical approaches, visualization techniques are needed to enable the interpretation of data as well as the communication of results. The design and implementation of such techniques lies at the intersection of the biology, bioinformatics, and data visualization fields. The purpose of Dagstuhl Seminar 18161 “Visualization of Biological Data – Crossroads” was to bring together researchers from all three fields, to identify opportunities and challenges, and to develop a path forward for biological data visualization research.

Seminar April 15 – 20, 2018 – <http://www.dagstuhl.de/18161>

2012 ACM Subject Classification Applied computing → Bioinformatics, Applied computing → Life and medical sciences, Applied computing → Life and medical sciences, Applied computing → Bioinformatics

Keywords and phrases imaging, omics, sequence analysis, visual analytics, visualisation

Digital Object Identifier 10.4230/DagRep.8.4.32

1 Executive Summary

Jan Aerts (KU Leuven, BE, jan.aerts@kuleuven.be)

Nils Gehlenborg (Harvard University, US, nils@hms.harvard.edu)

Georgeta Elisabeta Marai (University of Illinois – Chicago, US, gmarai@uic.edu)

Kay Katja Nieselt (Universität Tübingen, DE, kay.nieselt@uni-tuebingen.de)

License  Creative Commons BY 3.0 Unported license

© Jan Aerts, Nils Gehlenborg, Georgeta Elisabeta Marai, and Kay Katja Nieselt

The rapidly expanding application of experimental high-throughput and high-resolution methods in biology is creating enormous challenges for the visualization of biological data. To meet these challenges, a large variety of expertise from the visualization, bioinformatics and biology domains is required. These encompass visualization and design knowledge, algorithm design, strong implementation skills for analyzing and visualizing big data, statistical knowledge, and specific domain knowledge for different application problems. In particular, it is of increasing importance to develop powerful and integrative visualization methods combined with computational analytical methods. Furthermore, because of the growing relevance of visualization for bioinformatics, teaching visualization should also become part of the bioinformatics curriculum.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Visualization of Biological Data – Crossroads, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 32–71

Editors: Jan Aerts, Nils Gehlenborg, Georgeta Elisabeta Marai, and Kay Katja Nieselt



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

With this Dagstuhl Seminar we wanted to continue the process of community building across the disciplines of biology, bioinformatics, and visualization. We aim to bring together researchers from the different domains to discuss how to continue the BioVis interdisciplinary dialogue, to foster the development of an international community, to discuss the state-of-the-art and identify areas of research that might benefit from joint efforts of all groups involved.

Based on the topics identified in the seminar proposal, as well as the interest and expertise of the confirmed participants, the following four topics were chosen as focus areas for the seminar, in addition to the overarching topic of collaboration between the data visualization, bioinformatics, and biology communities:

Visualization challenges related to high-dimensional medical data. Patient data is increasingly available in many forms including genomic, transcriptomic, epigenetic, proteomic, histologic, radiologic, and clinical, resulting in large (100s of TBs, 1000s of patients), heterogeneous (dozens of data types per patient) data repositories. Repositories such as The Cancer Genome Atlas (TCGA) contain a multitude of patient records which can be used for patient stratification, for high-risk group and response to treatment discoveries, or for disease subtype/biomarker discoveries. Still, patient records from the clinic are used singularly to diagnose patients in the clinic without including likely insights from other sources. Similarly, molecular expression signatures from the omic sources barely impinge on the clinical observations. There is an urgent need to bridge the divide the precision medicine gap between the laboratory and the clinic, as well as a need to bridge the quantitative sciences with biology. Additionally, many precision medicine studies plan to include sensor data (e.g. physical activity, sleep, and other patient-worn sensors) that will add another dimension of complexity that analysis and visualization tools need to take into account.

This highly relevant topic focused on visual analytic tools and collaborations that will promote and leverage notions of patient similarity across the phenotypical scales. Scalable and robust machine learning methods will need to work synergistically to integrate evidence of similarity while meaningful visual encodings should simultaneously summarize and illuminate patient similitude at the individual and group level. This topic is closely related to some of the topics below.

Visualization of biological networks. Modeling the stochasticity of genetic circuits is an important field of research in systems biology, and can help elucidate the mechanisms of cell behavior, which in turn can be the basis of diseases. These models can further enable predictions of important phenotypic cellular states. However, the analysis of stochastic probability distributions is difficult due to their spatiotemporal and multidimensional nature, and due to the typically large number of simulations run under varying settings. Moreover, stochastic network researchers often emphasize that what is of biological significance is often not of statistical significance – numerical analyses often miss small or rare events of particular biological relevance. A visual approach can help, in contrast, in mining the network dynamics through the landscape defined by these probability distributions.

Another major challenge relates to finding “stable behavior” of networks, including those recruited in signal transduction. Multistability and bistability have been often studied in metabolic chemically reactive networks. Necessary conditions have been formulated to imply the emergence of stable phenotypes. However, these methods have been deployed on small networks. Recently many groups have recognized that scalable methods can be explored using steady state or quasi steady state models that are derived from stoichiometry and rate-action kinetics. These unfortunately suffer from the lack of methods that will examine

the large parametric space. Consider this: N interacting molecules imply N^2 interactions and in turn the same order of the governing “parameters” (activation rates and abundances). For even mid-size portions of salient pathways (EGFR, B-cell Receptor activation, etc.) finding stable states is challenging. It is certainly the case that a complete graph is never realized and sparsity and network mining can be used to glean the necessary structure. Design of experiments followed by visualization of parametric spaces will be required to search for these stable points. Furthermore, the huge size of this space needs possibly new scalable approaches for the visualization.

Visualization for pan-genomics. With the advent of next-generation sequencing we can observe the increase of genome data both in the field of metagenomics (simultaneous assessment of many species) as well as within the field of pan-genomics. In metagenomics, the aim is to understand the composition and operation of complex microbial consortia in environmental samples. On the other hand in pangenomics genomes within a species are studied. While originally a pan-genome has been referred to as the full complement of genes in a clade (mainly a species in bacteria or archaea), this has recently been generalized to considering a pan-genome as any collection of genomic sequences to be analyzed jointly or to be used as a reference rather than a single genome.

In bioinformatics, both topics impose a number of computational challenges. For example, a recent review paper by Marschall et al. on “Computational Pan-Genomics: Status, Promises and Challenges” (DOI: 10.1093/bib/bbw089) addresses current efforts in this sub-area of bioinformatics. This area needs novel, qualitatively different computational methods and paradigms. While the development of new promising computational methods and new data structures both in metagenomics and pangenomics can be observed, a number of open challenges exist. One of them in the area of pangenomics is for example the transition from the representation of reference genomes as strings to representations as graphs. However, the important topic of pangenome visualization has not been addressed in the aforementioned review. Interestingly this has been taken up in a break-out session in a recent Dagstuhl seminar on “Next Generation Sequencing - Algorithms, and Software For Biomedical Applications”, and identified as a topic of urgent interest and demand. One observation for example is that in pan-genomes there are segments of conserved regions interspersed by highly variable regions. Open question here is how to visualize the highly variable regions, or how to interpret its content in the context of its neighborhood. Other open visualization topics involve the visual representation of the graph structure underlying pangenomes.

In the field of metagenomics some common visualization approaches, such as heatmaps or scatter plots in combination with principal component analyses, are used, however, many open challenges exist. In particular those visualization tools that are developed for genomics studies fall short in representing large-scale, high dimensional metagenomics studies. Especially the magnitude of the data presents a challenge to meaningfully represent biologically valuable information from complex analysis results. Thus also in this topic the question of large-scale and heterogeneous data visualization is of central importance.

Curriculum development of biological data visualization. Parallel to the recognized need to teach bioinformatics students about big data in biology, there is a growing need to familiarise students with modern visual analytics methodologies applied to biological data, and to provide hands-on training. While several community members are teaching summer camps, tutorials, and workshops on biological data visualization, many of these educational sessions take the form of an introduction to specific tools. We find ourselves handling similar questions: what is exploratory data visualization, what is visual analytics, which frameworks

to think about visualization exist, how can we explore design space, and how can we visualise biological data to gain insight into them, so that hypotheses can be generated or explored and further targeted analyses can be defined?

Despite the increasing importance of visualization for bioinformatics, there is currently a general lack of integration into the bioinformatics education, and a useful and appropriate curriculum has not yet been developed. In this topic the following questions will be addressed: What should a modern and seminal curriculum for visualization in bioinformatics look like? How far along the introductory visualization courses should this curriculum go, while allowing biological data topics as well? What are the essential topics, and how can comprehensive training be achieved?

The schedule for the seminar was developed by the organizers based on previous successful Dagstuhl seminars. Emphasis was given to a balance between prepared talks and panels and break groups for less structured discussions focused on a selection of highly relevant topics. Three types of plenary presentations were available to participants who had indicated interest in presenting during the seminar: overview talks (20 minutes plus 10 minutes for questions), regular talks (10 minutes plus 5 minutes for questions), and panel presentations (5 minutes per speaker followed by a 20 – 25 minute discussion). The break out groups met multiple times for several hours during the week and reported back to the overall group on several occasions. This format successfully brought bioinformatics and visualization researchers onto the same platform, and enabled researchers to reach a common, deep understanding through their questions and answers. It also stimulated very long, intense, and fruitful discussions that were deeply appreciated by all participants.

This report describes in detail the outcomes of this meeting. Our outcomes include a set of white papers summarizing the breakout sessions, overviews of the talks, and a detailed curriculum for biological data visualization courses.

2 Table of Contents

Executive Summary

Jan Aerts, Nils Gehlenborg, Georgeta Elisabeta Marai, and Kay Katja Nieselt . . . 32

Program and Participants 38

Discussions and Outcomes 39

Conclusion and Next Steps 56

Overview of Talks 56

Spatial Networks in Neuroscience

Katja Bühler 56

Visualizing Public Health Data

Anamaria Crisan 57

Big Mechanism Visualization

Angus Forbes 57

Network Visualization Challenges

Karsten Klein 58

Biological Networks

Alexander Lex 59

Telling Stories With High-D Data in the Clinic

Raghu Machiraju 59

Curriculum Panel: Teaching Visualization

Lennart Martens 59

Visualizing and Interpreting Metabolite-Gene Relationships with RaMP

Ewy Mathé 60

Visualization of Single Cell Cancer Genomes

Cydney Nielsen 60

Strategic Graph Rewriting, Network Analysis, and Visual Analytics: challenges and thoughts

Bruno Pinaud 61

The Bio/Life-Sciences need better visualization of statistical network structures

William Ray 61

Making Sense of Large Scale Image Data

Jens Rittscher 62

High-Dimensional Medical Data Panel: Exploration and Communication in Bio-medical Visualization

Timo Ropinski 62

Metronome – Connecting genotypes and phenotypes

Christian Stolle 63

Collaborations between VIS / Bioinformatics / Bio Communities

Marc Streit 63

Visualization for Pan-genomes and Meta-genomes	
<i>Granger Sutton</i>	64
Democratization of Data Science	
<i>Blaz Zupan</i>	64
Panel discussions	64
Collaboration Panel: From Genomics/Bioinformatics to Visualization – in 5 minutes	
<i>Jan Aerts</i>	64
Collaboration Panel: Mutual Respect	
<i>Sheelagh Carpendale</i>	65
Biological Networks Panel: Matching the User’s Mental Map	
<i>Carsten Görg</i>	65
Biological Networks Panel: Visualising Biological Networks: comparison of trees to graphs	
<i>Jessie Kennedy</i>	65
Collaboration Panel: Visualization and Visual Analysis of Biomolecular Structures	
<i>Barbora Kozlíková</i>	66
Curriculum Panel: Designing a curriculum for teaching visualization in bioinformatics	
<i>Michael Krone</i>	67
Curriculum Panel: Vis is a large number of small problems	
<i>Martin Krzywinski</i>	67
Biological Networks Panel: Scaffolding Bionetwork Visualization with models and theories	
<i>Georgeta Elisabeta Marai</i>	67
Pan-Genomics Panel: Some questions and challenges about comparative genomics and pan-genomics	
<i>Kay Katja Nieselt</i>	68
High-Dimensional Medical Data Panel: High-Dimensional Medical Data	
<i>Jos B.T.M. Roerdink</i>	68
Biological Networks Panel: Visualisation of Biological Networks: Past, Present, and Future	
<i>Falk Schreiber</i>	69
Pan-Genomics Panel: Scaling Sequence Comparison for Pan & Metagenomics	
<i>Danielle Szafir</i>	69
High-Dimensional Medical Data Panel: Living with Algorithms	
<i>Cagatay Turkay</i>	70
Acknowledgements	70
Participants	71

■ **Table 1** Schedule of Dagstuhl Seminar 18161 from April 15th through April 20th, 2018.

Monday	Tuesday	Wednesday	Thursday	Friday
Introductions	High-D Medical Data Talks	Curriculum Panel	Curriculum Discussion	Breakouts
Collaboration Talks & Panel	High-D Medical Data Panel	Breakout Reports	Breakout Reports	Breakout Reports
Biological Networks Talks	Pan-Genomics Talks & Panel	Trip to Villa Borg	Breakouts	
Biological Networks Panel	Breakouts	Cloef Hike	Breakout Reports	

3 Program and Participants

An overview of the schedule for the seminar is provided in Table 1.

During the five days of the seminar, a total of 30 prepared presentations were given across five focus areas:

■ Collaboration

- *Overview Talk* – Marc Streit
- *Panel* – Sheelagh Carpendale, Jan Aerts, Barbora Kozlikova

■ Biological Networks

- *Overview Talk* – Falk Schreiber
- *Regular Talks* – Bruno Pinaud, Katja Bühler, Alexander Lex, Angus Forbes, Karsten Klein
- *Panel* – Will Ray, Jessie Kennedy, Carsten Görg, Liz Marai

■ High-Dimensional Medical Data

- *Overview Talk* – Raghu Machiraju
- *Regular Talks* – Cydney Nielsen, Jens Rittscher, Ewy Mathe, Ana Crisan, Christian Stolte, Blaž Zupan
- *Panel* – Jos Roerdink, Timo Ropinski, Cagatay Turkay, Raghu Machiraju

■ Pan-Genomics

- *Overview Talk* – Granger Sutton
- *Panel* – Danielle Szafr, Kay Nieselt, Granger Sutton

■ Curriculum

- *Panel* – Lennart Martens, Martin Krzywinski, Michael Krone

On the second day, participants joined one or in rare cases two breakout groups that focused on problems in these areas. The break out groups met multiple times for several hours during the week and reported back to the overall group on three occasions.

The breakout groups received detailed instructions to guide their discussions towards tangible outcomes. Specifically, the breakout groups were given the following tasks in addition to the discussion of their focus topic:

■ Day 2

- Identify driving questions for a publication
- Decide what type of publication and venue would be appropriate
- Create a timeline for the remainder of the week
- Identify a speaker for the breakout group

■ Day 3

- Create a rough outline of the manuscript and finalize paper type
- Review closest related work

- **Day 4**
 - Finalize outline
 - Assign manuscript sections to breakout group participants
 - Formulate one paragraph outlining the contributions of the manuscript
- **Day 5**
 - Agree on timeline for deliverables post-seminar

Based on feedback provided at the end of the seminar, this structured approach was well received by the participants and helped them to focus their discussions.

4 Discussions and Outcomes

High-Dimensional Medical Data

The topic on High-Dimensional Medical Data was split into three subtopics: patient similarity, trust, and awareness.

A. Patient similarity

The patient similarity workgroup included the following members: Jan Aerts, James Chen, Arlene Chung, Mirjam Figaschewski, David Gotz, Raghu Machiraju, Jens Rittscher.

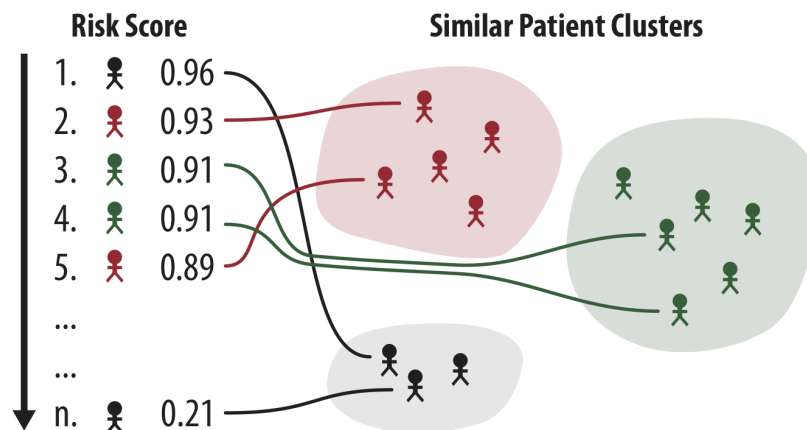
Comparing individuals is a common aspect in different levels of working with patient-related data. First, all-versus-all comparisons are relevant in patient stratification (e.g. to select a patient subgroup which is relevant for inclusion in a clinical trial, or to identify how patient populations behave in a public health context) as well as disease stratification. Second, a single patient can also be compared to larger groups, for example to identify the cases that a new patient resembles so that adequate treatment can be selected. In this workgroup, we discussed the context of calculating these similarities, their different types and constituent parts, and developed some recommendations including how visual analytics can fit into the process.

Patient information is collected in a long list of features (pathology, genotype, EHR-based features, lifestyle, treatment response, prognosis, etc), and different approaches were discussed for combining this information. In traditional stratification methods, such as risk scores, focus often lies on ranking patients in a linear order. However, because there is a mismatch between the linear ordering and the high-dimensional nature of patient data, patients with similar rank may be very different. For example, patients with the same risk for hospital readmission may be at risk for very different reasons (see Figure 1).

Early integration of features, on the one hand, allows for generating a more holistic overview of a patient population which allows the identification of e.g. subgroups and the stratification thereof. A prime example of this is the use of topological data analysis, which aims at discerning the underlying “shape” of a complex dataset (see Figure 2).

Other use cases – such as for point-of-care decisions – require a more hierarchical approach where features are considered one or a few at a time, as in a decision tree. Nevertheless, also for the point-of-care use case the placement of a patient in their broader context can be very beneficial. For example, capturing similarities and building reference libraries can allow for a more systematic approach to clinical grading.

Calculating these similarities however does have its challenges, especially when combining several across different categories and scales. We identified a number of issues with the



■ **Figure 1** In traditional stratification methods, a mismatch often exists between the linearity of e.g. risk scores, and the high-dimensionality and richness of the underlying patient data.

calculation of similarities. First, we may have clearly-defined similarity in specific cases but a family of similarities can give inconsistent partial ordering. Second, what if the corpus of data is dynamic? Third, data may be sparse and there may be individuals that do not have similarities to any other. Finally, there is a significant problem with missing data, as different patients will have data available for different features.

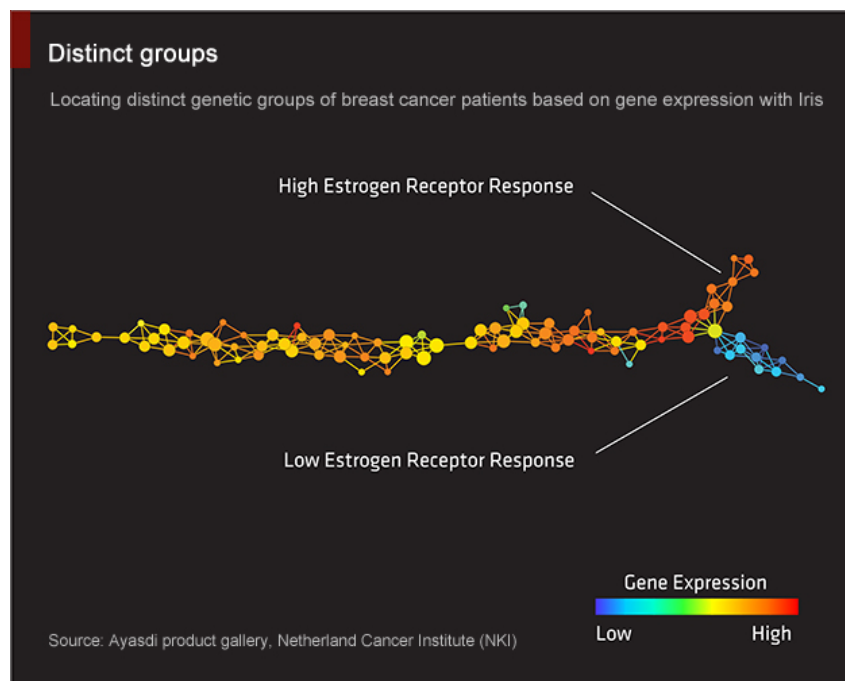
We believe that a visual analytics (VA) approach – i.e. combining interactive data visualization with automated analysis methods (both statistics and machine learning) – can alleviate some of the issues present in this field, and will allow for opportunities for more informed decisions, more trust in these decisions, and greater objectivity. The VA approach is particularly useful for so-called wicked problems such as this. Wicked problems are described as suffering from finitude in resources and/or knowledge, having very complex interactions between components, and partially depending on values and norms of the people involved. In this case, visual analytics can help in making the patient-comparison process more transparent, interpretable and contextualized for users who leverage those insights into their normal workflows. In particular, VA aims to help generate insights and uncover biases and issues with unknown assumptions as they can make these explicit.

B. Revealing biases in the biomedical research process through visualization

Group consisting of Ryo Sakai, Anamaria Crisan, Ewy Mathé, Torsten Möller, Christian Stolte and Jos Roerdink.

Modern biomedical research has become a complex process involving a growing arsenal of technical devices to generate data. It requires collaboration between disciplines to design experiments, manage and process the collected data, and interpret and analyse the results.

Trust is an essential ingredient for successful collaborative projects, and needs to exist on many levels, in each phase of a project: trust in protocols and measurement accuracy for data collection; in algorithms, models, and processes for data processing; in decision-making and result-finding; and, finally, trust in people and their willingness and ability to collaborate to disseminate the results. We believe that visualization could be used to build trust in the research process and confidence in its outcomes by addressing and revealing biases that can enter the process at each stage.



■ **Figure 2** Shape of NKI cancer patient population, coloured by ESR1 expression, and indicating a subpopulation of patients who survived with low ESR1 levels. (taken from Lum et al, 2013; for full details see reference)

Biases can be categorized by their source, along a gradient from machine-centered to human-centered bias (see Figure 3). Over time, the extent and source of a particular bias may change; overlaps can indicate a multitude of factors that need to be taken into account.

At different stages of a research project, the same bias may require different visualizations to reveal its effect in each particular context.

Careful selection of visualizations to highlight each potential bias will help make the analysis transparent, establish a solid basis for quality control and validation, and may also be useful for explaining methods in a publication.

Figure 4 is an example of a structure that can be populated with specific visualization types to address each kind of bias, at each project phase.

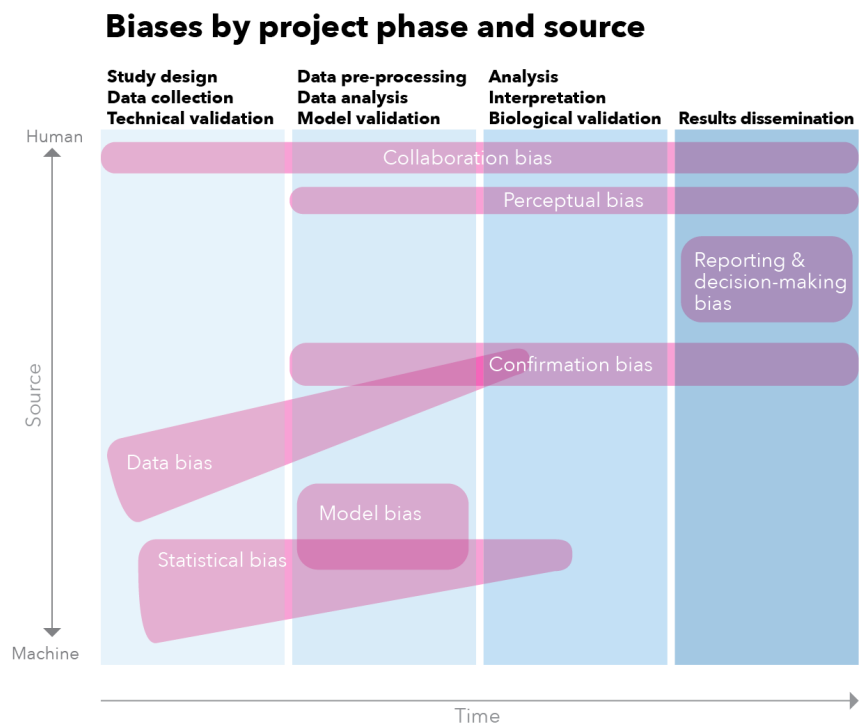
By identifying and quantifying potential biases visually, we believe we can help researchers become vigilant to flaws and pitfalls to mitigate risks in the biomedical research process.

Additional Sources:

- <https://eagereyes.org/basics/encoding-vs-decoding>
- http://decisive-workshop.dbvis.de/?page_id=555#101
- <https://www.computer.org/csdl/trans/tg/2006/04/v0421.html>
- https://en.wikipedia.org/wiki/List_of_cognitive_biases
- <https://betterhumans.coach.me/cognitive-bias-cheat-sheet-55a472476b18>

C. VISard: a card game

Group consisting of Martin Krzywinski, Timo Ropinski, Marc Streit, Cagatay Turkay, Michel A. Westenberg, Blaz Zupan.



■ **Figure 3** Categorization of bias types.

VISard is a card game and playful take on data visualization education and engagement. It teaches players about the vicissitudes of creating visualizations, dealing with data, users and tasks and the fortunes (good and bad) of practical aspects of computing, design and publishing.

Game goal. The goal of this multiplayer game is to be the first to create a visualization that satisfies requirements, while being subject to constraints, benefitting from lucky breaks and suffering setbacks due to unfortunate events. The game may be played cooperatively or competitively – you can hinder other players to avoid being scooped as you race to publish your visualization.

Game process. Each player creates a visualization by playing various cards. The visualization must meet an acceptable level of accuracy, intuitiveness and engagement.

These acceptable levels are defined by a combination of data set, user and task. These levels are the same for each player and generated at the beginning by randomly drawing requirement cards.

Visualization requirements. The requirements for a successful visualization are determined by three cards drawn at random from the requirements pile at the start of the game.

Data requirement cards (Figure 5) describe a dataset or analysis context such as protein interactions or bacterial phylogeny. Each of these data sets is associated with a specific type, such as a network or tree. These types influence the behaviour of other cards.

Each data set contributes uniquely to the accuracy, intuition and engagement requirement. For example, the bacterial phylogeny card (Figure 6) adds +4 to accuracy, +2 to intuition and +1 to engagement.

	Data collection	Algorithm/ Process/ Models	Decision making/ Results	Collaboration
Statistical bias				
Cognitive bias				
Model/ selection bias/ method				
Perceptual bias				
Data bias (sample bias)				
Reporting bias				
Etc. ...				

Figure 4 Categorization of bias types.

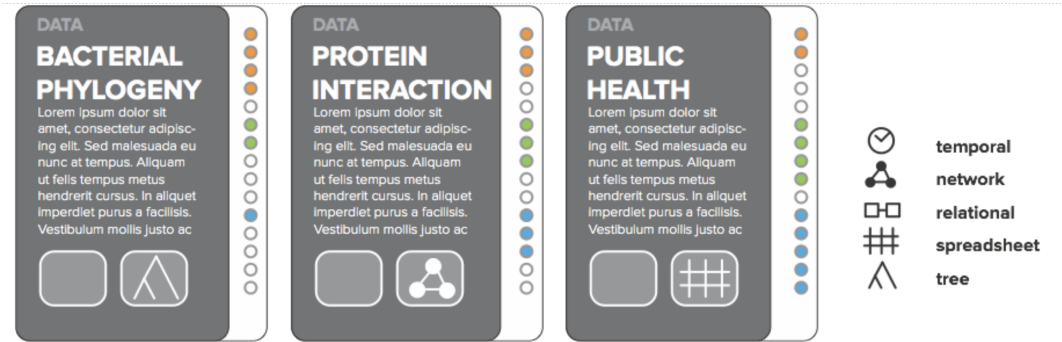


Figure 5 A sample of data requirement cards.

The second requirement is the user. Each of these cards has its own requirements (Figure 7). For example, designing a visualization for a scientist calls for high accuracy (+5) but low engagement (+1). On the other hand, kids require high accuracy (+5) but low engagement (+1).

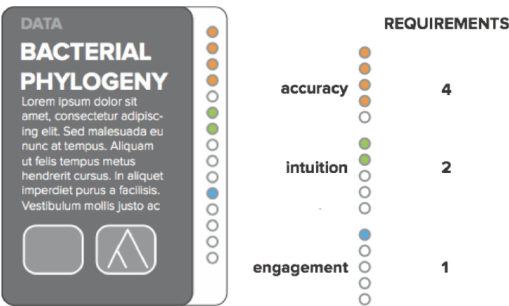
The final requirement card represents the task (Figure 8). Just as for data and user, the task cards contribute to the overall requirement.

The requirement cards are drawn at random – unusual combinations of data, user and tasks are possible! For example, consider the following requirement set: exploring bacterial phylogeny with kids (Figure 9). The total requirements for a visualization for this set of cards is 6 accuracy, 8 intuition and 8 engagement.

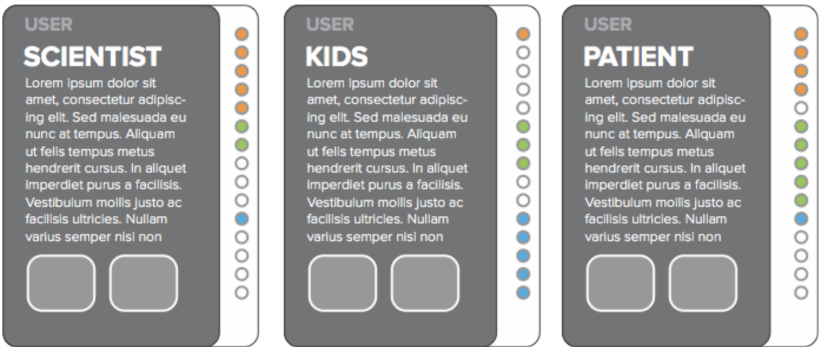
The cards are designed so that they can be stacked to show only the requirement tab to assist in counting the requirements (Figure 10).

Examples of requirement cards with scores. Scores are (accuracy,intuition,engagment,class) where class is an (optional) data type that influences the behaviour of other player cards (e.g. plot type).

- User: scientist (1,5,2), kids (5,1,4), patient (2,3,5), politician/decision maker (3,2,4), student of engineering (4,4,2)



■ **Figure 6** Each requirement card contributes towards a requirement in accuracy, intuition and engagement. A user’s visualization must meet or exceed each of these requirements for a successful visualization.



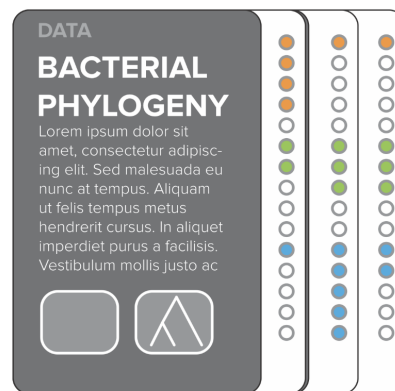
■ **Figure 7** A sample of user requirement cards.



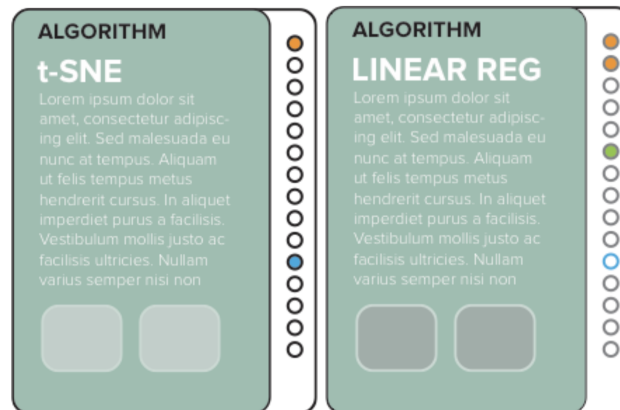
■ **Figure 8** A sample of task requirement cards.



■ **Figure 9** A sample set of data, user and task requirement cards that define the requirements for a successful visualization.



■ **Figure 10** Requirements are shown as circles on the card's tab, allowing cards of a similar type to be stacked.



■ **Figure 11** Example algorithm cards.

- Data: ECG (5,1,4, time series), Country happiness index (5,2,3,geo), Protein interactions (3,2,1,network), Health demographics (5,2,4,table+geo), Bacterial phylogeny (3,4,1,tree), Gene expression (1,5,4,table), Patient collection (1,5,2,text)
- Task: Outlier detection (4,2,2), Trends (2,1,2), Correlation (5,2,5), Cluster analysis (1,2,4), Pedigree analysis (2,3,4)

Building the visualization. Once the requirements have been determined, each player builds a visualization using a combination of analysis, plot, encoding and design cards. Each of these contributes uniquely towards the requirements (Figure 11).

For example, the t-SNE algorithm card contributes +1 to accuracy and +1 to engagement. However, it does not contribute to intuition. On the other hand, linear regression card contributes +2 to accuracy, +1 to intuition but imposes a penalty of -1 to engagement.

The requirement values selected for each card are a combination of our perception of the method, how it might be perceived by users and, importantly, of aspects of the algorithm that we wish to emphasize to the player. For example, the t-SNE card would briefly describe the algorithm and indicate that distance between projected points is not interpretable.

Similarly the plot (visualization) and encoding cards contribute to a user's visualization (Figure 12). Some cards may be incompatible with others – for example, a scatter plot cannot be used on time-course data.



■ **Figure 12** Examples of visualization, encoding and design cards.

Examples of visualization-building cards.

- Design: hot metal colormap (4,3,1), rainbow colormap (5,1,4), log scale (1,3,1), area encoding (4,2,5), length encoding (5,5,5), shape encoding (5,4,4)
- Transformation: k-means, regression analysis, t-SNE, MDS, missing value imputation, PCA
- Visualization type: scatterplot, force-directed layout, treemap, heatmap, matrix layout, mosaic diagram, circos, pie chart, bar chart, parallel coordinates, silhouette plot

Game mechanics – Single player.

1. Draw a data, user and task card randomly. These are your requirements.
2. From a deck of all other cards, player draws 6 cards to make a hand.
3. A round begins by playing a card towards the requirements. Every played card adds or subtracts from the running total of each of the 3 requirement categories. Cards are organized based on type and you can only have one card of each type in play at any given time.
4. Some cards have additional requirements that must be fulfilled. This may prevent playing certain cards or cards of a certain type.
5. Anytime a card played successfully, the user may discard up to 2 cards and draw to complete the hand.

Various end game scenarios are possible. 1. endless play (game ends whenever the player chooses). 2. when a fixed number of cards have been played (e.g. 5, 7, whatever).

Game mechanics – Multiplayer.

1. Draw a data, user and task card for the group and place them in the center. These are the requirements which every player attempts to meet.
2. Each player draws 5 cards from the deck to make a hand.
3. The group chooses who goes first and order proceeds clockwise.
4. A round begins by a player performing one of these tasks (user cannot pass): play a card to build up their solution or perform a task made possible by an action or event card. Cards are organized based on type and you can only have one card of each type in play at any given time. This can be facilitated by stacking the cards of a given type, with the active card placed on top of the stack.

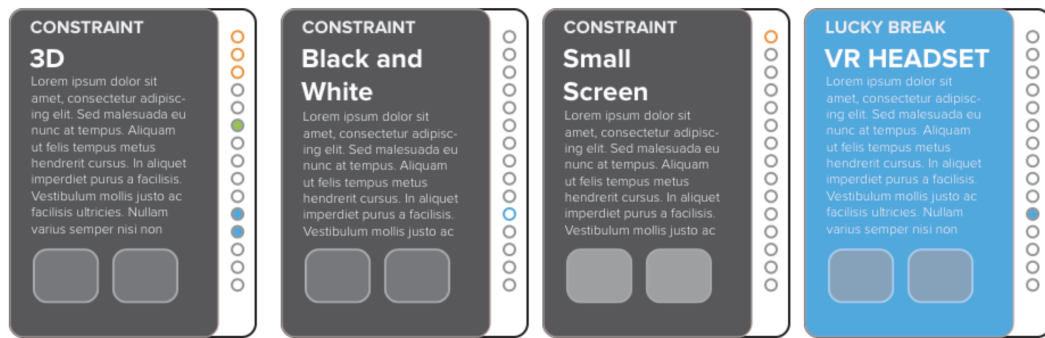
The game ends only when a player chooses to play a publish action card. In order to complete a visualization a player must have at least one card of each type.

Player-specific constraints. To teach users about the challenges of constraints and benefits of new technologies or approaches, mixed with the visualization-building cards are constraint cards (Figure 13).

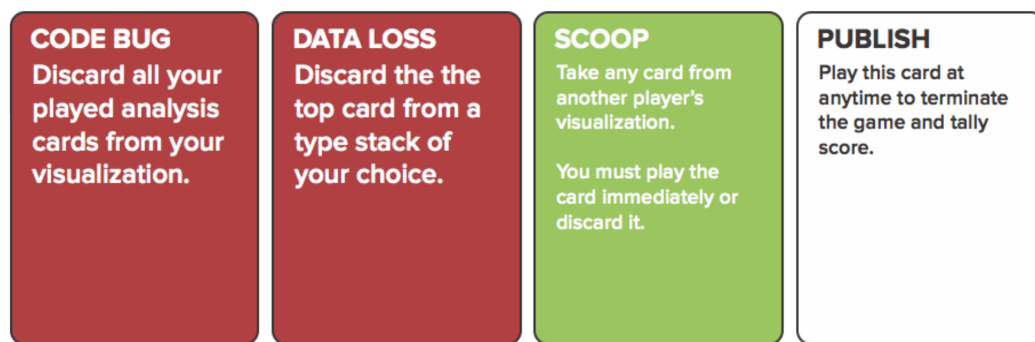
For example, if a user draws a 3D constraint card then they are penalized for accuracy but obtain a bonus for intuition and engagement – the constraint card contributes -3 to accuracy, +1 to intuition and +2 to engagement. Constraint card examples: Excel, PowerPoint, Tableau, Cytoscape, 3D, black and white, small screen, VR headset.

Events. Some cards in the deck act as events (Figure 14). When they're drawn, the player may be forced (or choose) to perform an action.

The event cards represent fortunate or unfortunate events that may occur during the process of building a visualization. Some cards penalize the player (e.g. data loss, which requires that a player discard one of the active cards in their hand) while others benefit the player (e.g., scoop, which allows the player to steal any card from another player).



■ **Figure 13** Examples of constraint cards which impose a bonus and/or penalty to the user. These are specific to a user and modify their requirements for a successful visualization.



■ **Figure 14** Examples of event cards.

The publish card is a special event. It is required for a user to be able to trigger the end of the game, assuming that they have met the requirements of the visualization.

Event cards.

- Publish – ends game at user's discretion
- Scoop – take a cards of a certain type from any other player's visualization. Steal card may allow for more than one type. The stolen card must be immediately played or discarded.
- Swap – exchange a card of a certain type from any other player. Swapped card must be played immediately. There are different kinds of swaps: single card from top of type pile, entire type pile, or entire visualization.
- Data loss – discard the number of cards on the data loss card from the top of type stack
- Reflect disaster – cancel action of another player on your hand or a drawn action card
- Change requirement card – replace one of the requirement cards by a new randomly drawn card
- Modify requirement card – alters the requirements for a given player, place this card face up near your visualization

Game modes. The game may be played with the requirements visible to all players (Figure 15).

The requirement cards are double-sided, with one side printed without the explicit requirements. In this mode, the players must anticipate the requirements of each card. The player who triggers the end of the game takes the chance of meeting the requirements. If



■ **Figure 15** An example game state in which player A and B both see the visualization requirements.

they do not, then the amount that they fall short off is added to the visualizations of other players when tallying the score (Figure 16).

Expansions. The game is scalable through expansion packs. Themes in current news, for example, can be made available as additional data or task cards (Figure 17). Similarly, newly published algorithms or visualizations can be accommodated.

Biological Networks

The biological networks group included the following members: Alexander Lex, Scooter Morris, Jessie Kennedy, Carsten Gorg, Bruno Pinaud, Anne Knudsen, Katja Buhler, Angus Forbes, William Ray, and Liz Marai. In a common meeting, the group brainstormed for open research topics. As a group, we then assessed both individual member interest and expertise in the resulting list of questions and culled the list. After several passes, we converged towards six main topics and the following subgroups; each subgroup produced next a list of keywords to better crystallize the topic, and a list of potential publication venues, along with the publication type (survey versus position versus guidelines etc.):



■ **Figure 16** An example game state in which player A and B do not know the precise requirements. Each player must guess the requirements for each card.



■ **Figure 17** Examples from an ethics expansion pack.

- Topic: Query-networks (details-first, expand on demand) for thousands of nodes
 - Members: A. Lex (Lead), S. Morris, C. Goerg, J. Kennedy, A. Knudsen
 - Keywords: zoom-into-detail vs reorient-for-detail, semantic vs geometric zoom, “too many” nodes vs “too many” edges, “meaning” of thresholds (weight vs relative weight)
 - Venue: Perspective PLoS CompBio
- Topic: Spatiality in neural networks
 - Members: L. Marai (Lead), K. Buhler, A. Forbes
 - Keywords: biological networks, spatial data, 3D coordinates, neurons, atlases, data integration, spatial nonspatial integration, survey, review, design, guidelines, neuroscience, connectome visualization, Hypergraphs, Multilayer networks, constrained layout, multidimensional projections (parallel coords, etc)
 - Venue: Review followed by Taxonomy/Position: TVCG, Nature Methods, Neuroinformatics
- Topic: Visualization for the Rule-Based Modeling of Biological Systems
 - Members: A. Forbes (Lead), B. Pinaud, L. Marai
 - Keywords: rule based model, rule inference, graph rewriting
 - Venue: Review, Bioinformatics/BMC Bioinformatics
- Topic: 10 simple rules to create biological network figures for communication
 - Members: L. Marai (Lead), S. Morris, A. Lex, J. Kennedy, C. Goerg, K. Buhler, B. Pinaud
 - Keywords: Visualization design, Ideas that need keywords/better searches: (data-centric vs user-centric vs task-centric, what to sacrifice for simplicity/informing the user); Is the visualization of both nodes and edges always necessary?
 - Venue: Guideline; PLoS CompBio
- Topic: What Cytoscape needs to do to get vis researchers to work in its web-browser
 - Members: S. Morris (Lead) and everybody else in the group
 - Keywords: Cytoscape, network visualization
 - Venue: (non publishable)
- Topic: Spatial Networks in Bioinformatics
 - Members: W. Ray (Lead), A. Forbes, B. Pinaud, L. Marai
 - Keywords: bioinformatics network visualization
 - Venue: Position/Survey/Guidelines

Additional topics that the group considered were: Comparison, Dynamic Nets, Spatiality in protein networks, Aggregation, Provenance of nets, Multi-attribute nets, and Hypernetwork graphs. These topics of interest could not be tackled due to time constraints, and were slated for discussion at future meetings.

Over the following energetic breakout sessions, the smaller groups converged towards an outline and an abstract for each publication, as well as which group member was in charge of which section. Group leads then contacted editors at potential publication venues. Each group agreed on a timeline for finalizing their target publication, on the platform they were going to use for drafts, and on their preferred means of communication.

Pangenomics

The working group for this topic consisted of the following four members (in alphabetical order):

Kay Nieselt, Jim Procter, Granger Sutton, Danielle Szafrir.

After a first discussion round it was agreed to work on the topic ‘Open Visualisation challenges for Pangenomics’. In the first discussion round also the following tasks, questions and open challenges were identified:

- Which are the standard and most commonly used visualisations for pangenomes?
- Which analytics should be connected with the visualisations, in term of visual analytics (VA) software?
- Which questions do researchers in pangenomics ask?
- What type of data feeds into a Vis or VA tool?
- What commonly used visual encodings exist?
- In terms of scalability (growing pan-genomes), what type of aggregation methods are in place or missing?
- For a given pangenome what is the best computational as well visual approach for an update of a given pan-genome?

In a separate document, many more details for each of these questions and open challenges have been collected.

Topic: ‘The 10 most important visualisations of pangenomes’

for the series ‘Ten simple rules’ in Plos Comp Biol.

The ten rules / most important visualisations are:

1. Central definition of the pangenome: Gene content is the core of pangenome. Thus the central visualisation is a matrix view of the gene content, possibly in conjunction with a synteny viewer (see below no. 4). A dichotomy of approaches exists:
 - a. Start at the whole genome alignment and layer features onto it
 - b. Start at the feature level and zoom into the nucleotide level
2. Overview and details on demand to represent gene organisation (an example tool is PanACEA (T. Clarke et al., BMC Bioinformatics 2018).
3. Visualisation of clustering results for the visual identification of core genome as well as unique genes for individual members. Analytical approaches are for example bi-clustering. Visualisation should allow reordering of rows and columns of matrix.
4. Clustering of species as a dendrogram combined with the gene content matrix, for example as a heatmap. An example tool is ROARY (Page et al., 2015) for a gene-content heatmap or Sequence Surveyor (Albers, Dewey, & Gleicher, 2011) for a heatmap encoding synteny.
5. Allow possibility of tree comparison, for example to highlight leaves that have been rearranged between two trees (example tool is TreeJuxtaposer by Munzner et al., ACM Transactions on Graphics (TOG) 2003)
6. Visually analyse horizontal gene transfer and gene loss (an example tool is panX by Ding, Baumdicker & Neher, NAR 2017)
7. Visually compare intersection and uniqueness of genes between two (sub)sets of a pangenome (given for example by a dendrogram, an example tool is Hierarchical Sets by Pedersen, 2017). If more than two sets (of species’ gene content) are compared, then UpSet (Lex et al., 2014) is a recommended tool.
8. Represent genomic architectures to visually show rearrangements in genomes (example tool is GenomeRing (Herbig et al., Bioinformatics 2012)
9. Curation of genomic annotation: a visualisation of a pangenome together with provided gene annotation in each strain can help to identify poorly and even erroneously annotated features. An example tool is Pan-Tetris (Hennig et al., BMC Bioinformatics 2016).
10. Aggregation: a challenge in future is the issue of scalability. A visual tool for pangenomes should offer the possibility to aggregate species and pangenes.

The group agreed to write such an article and it agreed on a timeline for finalizing the publication. The structure of the article is as follows: we start with (our) central definition of the pangenome with respect to its visual representation and mining possibilities. We will reflect the prokaryotic perspective more than the eukaryotic one and focus on a subset of tasks from prokaryotes. Then the ten rules will follow with example applications.

Collaboration

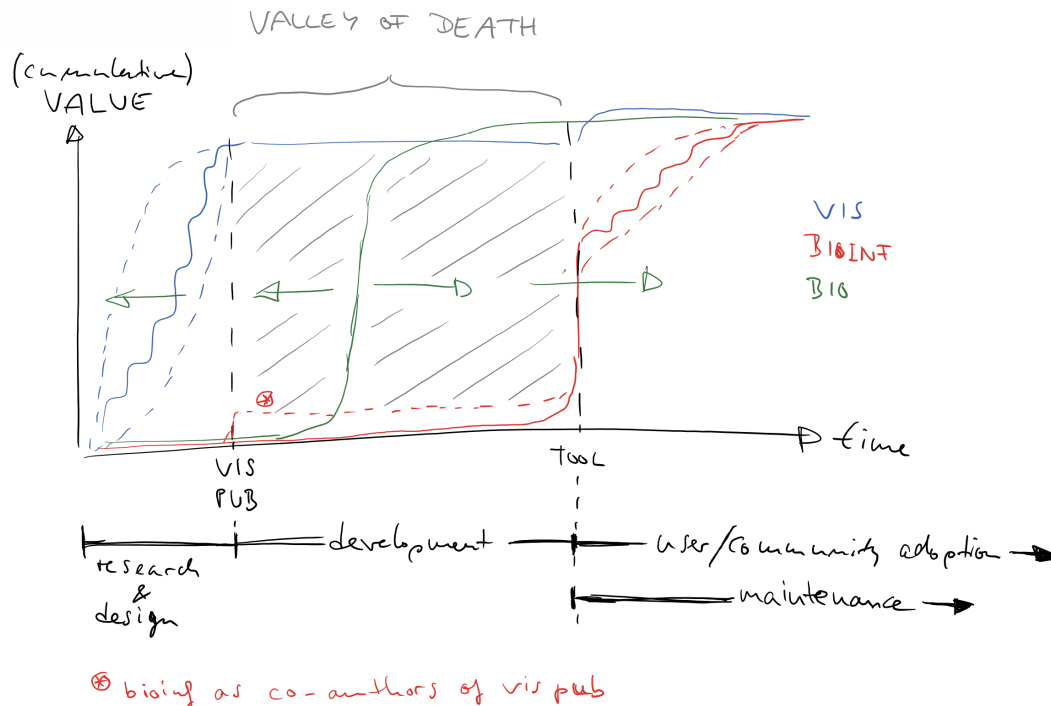
The following seminar participants joined the collaboration working group: Lennart Martens, Cydney Nielsen, Sheelagh Carpendale, Nils Gehlenborg, Michael Krone, Barbora Kozlikova, Helena Jambor, Falk Schreiber, Karsten Klein.

The collaboration breakout group focused on the question of how one can turn the visualization research conducted in a collaboration between visualization, bioinformatics, and biology researchers, into a stable tool that can serve the bioinformatics and biology community beyond the duration of the collaboration.

The group agreed that collaborations between data visualization, bioinformatics, and biology researchers can be productive and result in progress in all three fields. However, one of the major challenges encountered is that collaborations often end too early, when a prototype visualization has been created and potentially published, but not turned into a usable and maintainable tool for the bioinformatics and biology communities. The group identified the vastly different timescales of conducting and publishing research in these three communities as a major driver behind such undesirable outcomes. In essence, there is a period in which the visualization collaborator has already gained close to maximum cumulative value from the collaboration, when the bioinformatician and the biologist have not yet gained much value from the collaboration (see Figure 18). The group termed this period the “valley of death”, because it is the phase during which many collaborations fall apart.

Next, the group identified the reasons for why the valley of death exists, considering the perspectives of visualization, bioinformatics, and biology researchers. For visualization researchers, the main concerns are lack of incentives to move beyond a prototype and create a usable tool that would not result in a visualization venue paper, visualization researchers might lack the software engineering skills to build a production quality tool and the inability to attract and retain professional software developers, as well as a the lack of appreciation for a stable tool that could serve as a basis for future research. From the perspective of bioinformatics researchers, the biggest concerns are the need to publish a usable tool rather than a prototype, the time required to develop a usable tool, as well as the need to move from a feature-laden prototype to a streamlined tool that is focused on core functionality but stable. Biologists are generally less concerned about the valley of death, as the insight needed for their research question might be provided by the prototype – with the data loaded by the visualization researcher – or a a tool built based on the prototype.

Based on these observations, the group discussed what would need to happen in visualization, bioinformatics, and biology research for more collaborations to successfully cross the valley of death. A visualization researcher would be incentivized to continue a collaboration if there was recognition that a finished tool can offer value for future work, the consideration of publishable contributions to the tool (e.g. an evaluation or systems paper), if there was agreement that trans-community research is valuable, if there was more recognition for visualization work published outside the visualization community, and finally that usable tools will likely be helpful in establishing future collaborations. Bioinformatics and biology



■ **Figure 18** A draft diagram produced during the seminar that illustrates the concept of the valley of death.

researchers could support the process by allocating resources for the transition from prototype to tool, which is explicitly supported by some funding agencies. Additionally, biologists and bioinformatics should use the prototype to generate interest in the method in their communities, further incentivizing the development of a tool.

The key take away identified by the group is that everyone in the collaboration needs to have awareness about the valley of death and there needs to be agreement by among all collaborators about how the valley of death will be crossed in this particular context.

Finally, the group identified several representative examples from different biological and data visualization domains, that illustrate characteristics of both successful and unsuccessful collaborations. Based on these examples, the group formulated a structure for the ideal collaboration, which is broken down into three stages. The first stage would result in a prototype and publication in the visualization literature, the second stage would result in joint publications in biology and bioinformatics venues describing a tool based on the prototype and applications of that tool, and the third stage would be adoption of the tool by the community and transition from active development to long-term maintenance.

Curriculum

The seminar discussion of a curriculum took place in three phases. In the first phase, the seminar participants dedicated one breakout session and a plenary session to discussing the contents of a curriculum in biology visualization. For the first breakout session, the seminar participants partitioned spontaneously in five groups. We took advantage of the splendid weather and grouped around the tables in the yard outside the dining room. Each group

discussed the core competencies required in a biology visualization course, and reported back to the plenary.

In the second phase, a small subset of participants, representing each group, dedicated one additional breakout session to summarizing the output of the seminar's work in this direction. Representatives Torsten Möller, Liz Marai, Danielle Albers-Szafir, William Ray, and Bruno Pinaud collected the notes from all working groups and compiled a taxonomy for the contents of such a biology visualization course. One result that emerged from this discussion was that different audiences have different content needs: for example, someone teaching molecular biology visualization will need to cover the rendering pipeline, while someone teaching genomic visualization will not. The result of this intensive work and discussion was a matrix table of contents, with sections mandatory for all such courses, and with optional topics depending on the data type (see table below).

I. Cross-Cutting Processes.

1. Why Visualization?
2. Tasks+Data+Workflows
3. Design Principles + Typography (both process, e.g., prototyping, and visualization design)
4. Evaluation
5. Provenance (optional)
6. Ethics (optional)
7. Rendering Pipeline (optional)

II. Applications. Choose topic(s) of interest, e.g. a subset of Geospatial Data Images, Networks, Populations & Sets, Sequences & Genomes, Tables, Text, and Three-Dimensional Structures. Cover the following topics for each application:

1. Color
2. Perception
3. Visual Encodings
4. Facets
5. Interaction
6. Summarization

III. Additional Characteristics of Data.

1. Temporality
2. Scale / Multi-scale
3. Uncertainty

Each cell in the resulting matrix will be crowd-sourced and moderated by a designated contributor. Each such cell will contain metadata and links to example teaching materials, organized along the following categories:

- Moderator
- Table of Contents
- Instructional Videos (Brief, 5-15 minutes)
- Reading Materials
- Examples (good and bad)
- Exercises & Summative Evaluations
- Tools
- Tutorials
- Example Courses (including durations & schedules)

- Learning Outcomes (Bloom’s Taxonomy)
- Lecture Materials

In the third phase, the results from this consolidating work were reported and discussed in another plenary session. The seminar participants were extremely pleased with the outcome. A lively discussion of the logistics for completing the cells of the curriculum table followed.

5 Conclusion and Next Steps

In the final plenary session all participants of the seminar discussed the possibility of a follow-up meeting or even the possibility to have a regular Dagstuhl seminar about the topic of large-scale biological data visualization. An overwhelming majority of the group voted for a follow-up or even regular meeting. This was also confirmed in the Dagstuhl survey. Finally, the result of the survey showed that the scientific quality of the seminar was rated as ‘outstanding’ (as a median). Thus, the organizers of this seminar would like to discuss possibilities for repeating this seminar with the Dagstuhl directors and staff.

6 Overview of Talks

6.1 Spatial Networks in Neuroscience

Katja Bühler (VRVis – Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Katja Bühler

In my talk, I addressed one of the questions given as a point for discussion in context of the Networks Panel. “What type of spatial data (3D coordinates), if any, show up in biological networks?” Understanding how the brain works is currently subject of large scale brain initiatives worldwide generating huge amounts of data at all scales. The brain is a spatial structure and consequently also the data underlying neurocircuit research is inherently spatial. I started with an overview on different kinds of spatial data and spatial networks being central to neurocircuit research and how standard brains and hierarchical brain parcellations are used to establish a spatial and semantic reference system. These reference systems allow us to integrate data across scales and types, to perform data aggregation to reduce complexity and to provide important anatomical and functional context for neuroscientists. I presented several examples illustrating how these spatial data characteristics can be exploited to create comprehensive network visualizations, to design data structures ensuring interactivity on large scale datasets and to fuse heterogeneous network and non-network data across scales. A discussion of open challenges and requirements on visualizations and interactive visual analytics systems for supporting neuroscientists in their daily research tasks concluded my talk.

6.2 Visualizing Public Health Data

Anamaria Crisan (*University of British Columbia – Vancouver, CA*)

License  Creative Commons BY 3.0 Unported license
© Anamaria Crisan

Background. Genomic epidemiology integrates next-generation sequencing data from surveillance programs and outbreak investigations with administrative datasets, providing a rich pool of data to inform public health decision-making. Bioinformatics pipelines culminating in data visualizations are often used to explore, interrogate, and communicate these complex integrated datasets, but while the bioinformatics tools underlying these platforms are rigorously tested and evaluated, the resulting data visualizations are often created on an ad hoc basis.

Methods. We have conducted a systematic review of the microbial genomic epidemiology literature from the past ten years to survey existing visualizations and to systematically characterize those visualizations by creating a why-what-how annotation code set that describes why the visualization was created (e.g. to show disease transmission in a hospital), what data were used (e.g. genomic data, event data, outcome data), and how the data was visualized (e.g. phylogenetic tree, timeline). To populate the why-what-how code set in a reproducible, transparent, and timely manner we have also created a pipeline that uses text mining and topic modelling to understand why a visualization was created followed by annotation using online open source software borrowed from image analysis research to derive the what and how components of the code set. Together the components of the why-what-how code set form the basis of a typology.

Results. We have developed GEViT (Genomic Epidemiology Visualization Typology), which allows researchers to systematically characterize and analyze visualizations developed specifically for microbial genomic epidemiology applications. Our initial findings show that different visualizations for a common objective (why) incorporated different data types (what) and used a variety of approaches to visualize these data (how), from colour to shapes to textual annotations. The preliminary data visualization corpus and associated code set have been compiled into a searchable gallery with suggestions of best practices that researchers and public health officials can use to guide data visualization efforts to communicate findings, or inform the design of data visualization components within analytic tools.

Conclusions. Through the development of GEViT, we demonstrate how it is possible to reason systematically about data visualization design and analysis. We anticipate that the GEViT resources will provide a comprehensive framework that allows researchers and healthcare stakeholders to design and analyze visualizations that facilitate the exploration and interpretation of complex healthcare datasets.

6.3 Big Mechanism Visualization

Angus Forbes (*University of California, Santa Cruz, US*)

License  Creative Commons BY 3.0 Unported license
© Angus Forbes

My talk presents a series of visualization projects related to the “Big Mechanism” program, supporting a range of tasks broadly relating to the assembly and execution of biological networks extracted from biomedical texts. Understanding the complex processes of life

requires multiple points of view, enabling a wide range of analysis tasks. This understanding, especially when drawn from contemporary heterogeneous big datasets, is often only a working model which is likely to undergo revision, and some of the visualization tools I present explicitly indicate where our knowledge comes from, i.e., which databases, which cell lines, which articles, which sentences. Designing accurate representations of biological data is in itself a interesting research topic, but it is also important to create representations that support useful ways of analyzing this data, and another series of tools I present utilize novel encodings to facilitate reasoning about the dynamics of and casual relationships within complex biological systems.

Overview of BioVis projects:

1. P Murray, F McGee, and AG Forbes. A taxonomy of visualization tasks for the analysis of biological pathway data. BMC Bioinformatics 18(2), 2017. https://creativecommons.soe.ucsc.edu/pdfs/Murray_BioPathTaxonomy_BMCBioinformatics2017.pdf
2. P Boutillier, M Maasha, X Li, HF Medina-Abarca, J Krivine, J Feret, I Cristescu, AG Forbes, and W Fontana. The Kappa platform for rule-based modeling. Bioinformatics 34(15), 2018. https://creativecommons.soe.ucsc.edu/pdfs/Boutillier_KappaPlatform_BioVis2018.pdf
3. AG Forbes, A Burks, K Lee, X Li, P Boutillier, J Krivine, and W Fontana. Dynamic influence networks for rule-based models. IEEE Transactions on Visualization and Computer Graphics 24(1), 2018. https://creativecommons.soe.ucsc.edu/pdfs/Forbes_DIN-Viz_VAST2017.pdf
4. AG Forbes, K Lee, G Hahn-Powell, MA Valenzuela-Escárcega, and M Surdeanu. Text annotation graphs: Annotating complex natural language phenomena. LREC 2018. https://creativecommons.soe.ucsc.edu/pdfs/Forbes_TAG_AnnotatingComplexNLPPhenomena_LREC_2018.pdf
5. TN Dang, P Murray, J Aurisano, and AG Forbes. ReactionFlow: An interactive visualization tool for causality analysis in biological pathways. BMC Proceedings 9(6), 2015. https://creativecommons.soe.ucsc.edu/pdfs/Dang_ReactionFlow_BioVis2015.pdf

6.4 Network Visualization Challenges

Karsten Klein (Universität Konstanz, DE)

License  Creative Commons BY 3.0 Unported license
© Karsten Klein

Network visualization has come a long way and there are many solutions for problems that were posed 10-20 years ago. However, some problems are not really solved, and new issues arise as more and more data is available, integrated, and also tasks become more complex. Notable examples are visual comparison and visualization of dynamic networks. In addition, new technology is available the affordances and requirements of which are often ignored in the visualisation concept design. I give an overview on network visualisation from my point of view, list the most pressing challenges, and give a few examples from my current research.

6.5 Biological Networks

Alexander Lex (University of Utah – Salt Lake City, US)

License © Creative Commons BY 3.0 Unported license
© Alexander Lex

There is a variety of biological data types that can be modeled as networks. Most of these networks are more valuable if they are considered in the context of node and edge attributes. In this talk I present some layout adaption/linearization strategies to visualize such multivariate networks.

I also introduce a distinction between overview and local tasks, those that are concerned with specific nodes, and argue that local network analysis tasks are more common. I present some techniques that are optimized to ensure readability for local network analysis tasks.

6.6 Telling Stories With High-D Data in the Clinic

Raghu Machiraju (The Ohio State University – Columbus, US)

License © Creative Commons BY 3.0 Unported license
© Raghu Machiraju

It is typical for multiple data to be used in the clinic for diagnosis. However, diagnosis in the clinic is stymied by genetic heterogeneity which often results in different outcomes for the same treatment. Patient stratification and biomarker discovery is needed while using multiple data. In this talk, we discuss how “visual stories” can help with gleaning disease etiology and lead to better patient stratifications. Many of these visual stories use interpretable representations. A case is also made for data-driven and often uninterpretable representations especially obtained from deep learning methods.

6.7 Curriculum Panel: Teaching Visualization

Lennart Martens (Ghent University, BE)

License © Creative Commons BY 3.0 Unported license
© Lennart Martens

Teaching visualization can follow an overall scheme that works for most courses (or curricula). This schema focuses (in that order) on the following answers to the questions that a student would like answered. (i) What is the topic of the course (centred on definitions and/or description)? (ii) Why is this topic important? (iii) What is the objective of this course/curriculum (what are the learning outcomes)? (iv) What can be done to reach this course/curriculum objective (transferring the relevant knowledge teaching the actual skills)? (v) Learn how to apply these skills in practice. With regards to the content of such a course, some elements that come to mind are listed next. Start from poor examples of visualizations, and critique these. Teach the basics of human perception. List and describe visual elements, and what each is good for (possibly extending this with what each of these is not good for, similar to the format for software design patterns). List and describe graphical representations and their uses in the same overall way. List and describe existing frameworks for visualization, and teach elementary considerations related to how to make libraries or

plugins in such tools. The practical sessions can be based on improved visualizations for poor examples that the students have found.

6.8 Visualizing and Interpreting Metabolite-Gene Relationships with RaMP

Ewy Mathé (Ohio State University – Columbus, US)

License © Creative Commons BY 3.0 Unported license

© Ewy Mathé

Joint work of Ewy Mathé, Elizabeth Baskin, Senyang Hu, Andrew Patt, Jalal K. Siddiqui, Bofei Zhang
Main reference Bofei Zhang, Senyang Hu, Elizabeth Baskin, Andrew Patt, Jalal K. Siddiqui, Ewy Mathé: “RaMP: A Comprehensive Relational Database of Metabolomics Pathways for Pathway Enrichment Analysis of Genes and Metabolites”, *Metabolites*, 8(1). pii: E16, 2018.
URL <https://doi.org/10.3390/metabo8010016>

The value of metabolomics in translational research is undeniable and metabolomics data is increasingly being generated in large cohorts, alongside other omics data such as gene expression. Analysis of these integrated datasets and functional interpretation of disease-associated metabolites is difficult, and is often hampered by the lack of user-friendly computational tools. With this in mind, we developed RaMP (Relational database of Metabolomics Pathways), which integrates biological pathways from KEGG, Reactome, WikiPathways, and HMDB. The database is accessible directly (mysql dump) or through an R package that is publicly available via GitHub (<https://github.com/Mathelab/RaMP-DB>) and includes detailed documentation on installation and usage. The next steps are to visualize the contents of the database to evaluate metabolite annotations between different databases and to create visual approaches to enable toggling between different types of information (e.g. biological pathways, chemical information, etc.). During this process of developing tools, it is important to balance out generalized/simple tools, that are easier to implement and arguably more user-friendly, and domain-specific/tailored tools, that are powerful and flexible but require in-depth understanding. In all cases though, robustness and reproducibility should be integrated.

6.9 Visualization of Single Cell Cancer Genomes

Cydney Nielsen (BC Cancer Agency – Vancouver, CA)

License © Creative Commons BY 3.0 Unported license

© Cydney Nielsen

Joint work of Cydney Nielsen, Maia Smith, Samantha Leung, Viktoria Bojilova, Oleg Golovko, Daniel Machev, Sohrab Shah

Cancer development is an evolutionary process driven by mutation. Single cell genomics is changing our ability to quantify tumour heterogeneity and observe the dynamics of genetically distinct cells over time and anatomical space. This rich research domain offers many visualization challenges and I will highlight four pressing issues that potentially generalize to other areas of biomedical research: (1) designing new visual representations; (2) creating interfaces that serve a broad spectrum of users; (3) achieving responsive interactivity with large and varied data; and (4) integrating with the analytical process. In conclusion, I would encourage us as a community to further integrate our visualizations into the relevant analysis

workflows, such that interactive visualization is increasingly embraced by the bioinformatics and biology communities as a central analysis methodology, rather than niche.

6.10 Strategic Graph Rewriting, Network Analysis, and Visual Analytics: challenges and thoughts

Bruno Pinaud (University of Bordeaux, FR)

License © Creative Commons BY 3.0 Unported license
© Bruno Pinaud

Joint work of Maribel Fernández, Hélène Kirchner, Bruno Pinaud, Jason Vallet

Main reference Maribel Fernández, Hélène Kirchner, Bruno Pinaud, Jason Vallet: “Labelled graph strategic rewriting for social networks”, *J. Log. Algebr. Meth. Program.*, Vol. 96, pp. 12–40, 2018.

URL <http://dx.doi.org/10.1016/j.jlamp.2017.12.005>

In my 10-minute talk I have presented some challenges and thoughts about rule-based modelling and the usage of visualisation to steer every step of the workflow: model design, simulation, then analysis. This talk is based on my work on Porgy (<http://porgy.labri.fr>) and the collaboration with Maribel Fernandez (King’s College London), Hélène Kirchner (Inria, France) and Guy Melançon (U. Bordeaux, France).

I started with a quick reminder about Graph Rewriting which is all about designing executable specifications of complex systems and in the end trying to understand how the behaviour of the system at a global scale emerges from rules specifying how local modifications operate. To create such a software, one big challenge is if there is a data-structure universal enough to handle efficiently all operations of the system and moreover, a data-structure powerful enough to support different type of networks (e.g., bio, social net, capital markets, relational database design).

To give my answer to this question, I have presented in a few slides our visual graph programming environment called Porgy and our data-structure called “labelled port graph”. However, this graph model has to be used along with a graph hierarchy mechanism to avoid duplicating nodes/edges/attributes like the one implemented in Tulip (Porgy is built upon Tulip). To conclude, I left some open questions about the usage of higher order rules (i.e., a node of the rule replaces a sub-graph) and from a more technical point of view the usage of graph database to improve the rule matching phase.

6.11 The Bio/Life-Sciences need better visualization of statistical network structures

William Ray (Ohio State University – Columbus, US)

License © Creative Commons BY 3.0 Unported license
© William Ray

Many biological systems possess properties such that there are natural elements that are thought of as nodes, with weighted connections between them that can be thought of as edges, but that do not fit into traditional graph-theoretic frameworks, and that therefore are difficult to represent using traditional graph-layout tools.

For example, if one looks at a protein family – a collection of proteins from different organisms that all perform the same function – one can learn quite a lot about why proteins that perform that function work, or don’t work.

However, that inspection requires looking not-only at the choices that evolution has made at each position in the protein, but also how these choices are interrelated.

Unfortunately the intuitive graph-theoretic representation for a protein treats each sequential residue as a node in a graph, and treats dependencies, distances, or other biophysical relationships between residues that can be determined for the family, as weighted edges between nodes. This representation can show, for example, the mutual information between different positions in the family alignment, but can't show amino-acid-specific relationships between position.

As a result, this position-centric view disguises many important dependencies, such as when the large majority of choices are independent, but some small subset have a strong dependence requirement.

Conditional Random Fields provide an interesting formalism for approaching this data. Structurally CRFs (and similar probabilistic networks) describe node-link networks where each node contains a set of categorical sub-nodes, and each edge is composed of conditionally-weighted sub-edges between the sub-nodes. This formalism maps conveniently to these biological networks, and it appears to be a natural mapping to many biological phenomenon with conditionally-interrelated features. As a result, visualizing and interacting with the structure of Conditional Random Fields can provide important insight into fundamental biology.

6.12 Making Sense of Large Scale Image Data

Jens Rittscher (University of Oxford, GB)

License  Creative Commons BY 3.0 Unported license
© Jens Rittscher

Three concrete examples for generating high-dimensional data from image data sets are being presented. The first illustrates visualisation tasks in high-throughput screening. The main challenge here to discover new cellular phenotypes. A human organotypic cell culture system that models epithelial interactions in vitro serves as a second example. Finally, I will use digital pathology to demonstrate how various different technology can be nitrated to visualize genomic and molecular information in the tissue architecture context. In summary, the talk will motivate three questions and challenges: (1) How can we integrate dynamic information over time, (2) How can we analyse and visualise expression of molecular markers in the tissue architecture context and (3) The need for developing metrics that capture patient similarity.

6.13 High-Dimensional Medical Data Panel: Exploration and Communication in Biomedical Visualization

Timo Ropinski (Universität Ulm, DE)

License  Creative Commons BY 3.0 Unported license
© Timo Ropinski

When designing visualizations, typically user, data and task need to be considered in order to obtain an effective visualization. Whereas in the area of biomedical visualization at least three different types of users need to be taken into account: medical doctors, medical researchers, and patients. Furthermore, with respect to data, often the high dimensionality in this context is challenging. Unfortunately, for many scenarios in this field, state-of-the-art high-dimensional visualization techniques are not appropriate. When for instance a medical

doctor analyses blood work, often the main task is to compare the data at hand with given reference values. Thus, no embedding of several data sets is required, but rather a comparative visualization of relatively few ones. Similar requirements must be met when a patient reviews his/her tracked health data. On the other hand, when a medical doctor communicates made findings, storytelling techniques seem to be the relevant technique of choice. Accordingly, biomedical visualization researchers need to look into these different requirements, when developing or selecting appropriate visualizations.

6.14 Metronome – Connecting genotypes and phenotypes

Christian Stolte (New York Genome Center, US)

License © Creative Commons BY 3.0 Unported license
© Christian Stolte

Joint work of Christian Stolte, Kevin Shi, Nathaniel Novod, Nina Lapchuk, Fred Criscuolo, Toby Bloom

URL <https://metronome.nygenome.org>

MetroNome is a web-based genotype/phenotype exploration platform with a data visualization interface. It is focused on enabling data sharing, data integration, data exploration, and identification of cohorts via complex combinations of genotypic and phenotypic traits, across diseases.

Metronome is intended to allow researchers to:

- explore data with minimal effort to generate and test hypotheses
- identify cohorts of interest by filtering across multiple types of data, including genotype and clinical data
- use data visualization to find relationships among genomic variants and subject or sample attributes
- share data among groups of collaborators, privately and securely
- combine private data with large public cohorts while retaining full control over that data

Metronome is intended to hold all types of genomic variants, gene expression data, as well as de-identified subject data from medical records.

6.15 Collaborations between VIS / Bioinformatics / Bio Communities


Marc Streit (Johannes Kepler Universität Linz, AT)

License © Creative Commons BY 3.0 Unported license
© Marc Streit

In the first part of my talk, I summarize what advice researchers and practitioners can get from a theory of visualization. We – as a community – currently provide advice by publishing models and theories, by collecting techniques and methods, and by describing best practices. While this is very useful, it is often not actionable. A less explored possibility is to provide cheat sheets in the form of decision trees that can help practitioners to create effective visualizations. These decision trees could be created as a community effort, underpinned with our models, and carefully annotated. In the second part, I talk about why generalizing design studies is hard, why data and task abstraction is key to create impact in visualization through collaboration with domain experts, and what lessons I've learned from previous collaborations.

6.16 Visualization for Pan-genomes and Meta-genomes

Granger Sutton (J. Craig Venter Institute – Rockville, US)

License  Creative Commons BY 3.0 Unported license
© Granger Sutton

Pan-genomes share many characteristics with meta-genomes and can use the same visualization approaches in part but also have distinctive needs. At the highest level a pan-genome is a universe of genes distributed across a set of genomes where each genome contains a set of genes which is a subset of the universe. This is also true for meta-genomes but with genomes being replaced by samples or environments. The top level data representation is then a two dimensional matrix of genes across genomes or samples. A typical visualization is a heat map which has been bi-clustered to provided cladograms in both dimensions. Both can also be represented by metabolic networks of what functional capabilities are contained. In many cases a meta-genomic sample will in fact contain one or more pan-genomes. Meta-genomes tend to have much less complete or fractured genome representations than pan-genomes. Deconvoluting assembled contigs into species specific bins is unique to meta-genomes and often read based approaches rather than assembled contigs are used for meta-genomes.

6.17 Democratization of Data Science

Blaz Zupan (University of Ljubljana, SI)

License  Creative Commons BY 3.0 Unported license
© Blaz Zupan

I will talk about how visual programming, interactive visualization, and explorative data analysis can help us in making visual analytics and machine learning accessible to everyone. I will demo these concepts in the case of single cell data analytics in Orange (visit <http://orange.biolab.si> or <http://youtube.com/orangedatamining> for short videos).

7 Panel discussions

7.1 Collaboration Panel: From Genomics/Bioinformatics to Visualization – in 5 minutes

Jan Aerts (KU Leuven, BE)

License  Creative Commons BY 3.0 Unported license
© Jan Aerts

In this short talk in preparation on a panel discussion regarding collaboration between visualization experts and biology/bioinformatics researchers, I start with a quick overview of my own journey from genomics to visualization research. In addition, and more importantly, I indicate some challenges that we encounter in bridging the gap between these two domains. These include reusability of solutions, composability of bioinformatics tools versus often monolithic approach for visualization tools, and the (incorrectly) perceived nice-to-have view on visualization in omics projects.

7.2 Collaboration Panel: Mutual Respect

Sheelagh Carpendale (University of Calgary, CA)

License © Creative Commons BY 3.0 Unported license
© Sheelagh Carpendale

This talk is about mutual respect – just one of the many important factors in a good collaboration. One aspect of mutual respect is developing an understanding of how the different research communities think about the way they would like to make contributions to their discipline. While biologists’ goals usually center around developing a better understanding of their data, ideally leading to new biological insights, visualization researchers’ goals usually center around contributing to advancing visualization through creating new visual representations, new layout approaches and/or new exploration techniques. There can be a tendency to favour the more easily understand the global importance of new biological insights. However, it is important in a collaboration that one disciplines goals not over shadow the other. We should remember that there are visual representations that have fundamentally empowered society, for example, the alphabet is a ‘visualization’ of spoken language. It may be difficult to learn but is a very powerful visual representation.

7.3 Biological Networks Panel: Matching the User’s Mental Map

Carsten Görg (University of Colorado – Aurora, US)

License © Creative Commons BY 3.0 Unported license
© Carsten Görg

Biologists tend to think about relationships between biological entities they study in a specific way. Often, they have a detailed mental map or even use an actual sketch or drawing that represents relationships between biological entities or biological processes. Computationally generated representations of networks typically don’t match the biologists’ mental or drawn representation. We propose a network layout approach that arranges the nodes not only based on the network topology but takes the underlying biological semantics into account to create a high-level blueprint of the network. Biologists can interactively rearrange the elements in the blueprint so that the representation matches their mental model. The detailed layout is then generated based on the constraints and structure defined in the blueprint.

7.4 Biological Networks Panel: Visualising Biological Networks: comparison of trees to graphs

Jessie Kennedy (Edinburgh Napier University, GB)

License © Creative Commons BY 3.0 Unported license
© Jessie Kennedy

Comparison in and between biological networks is a common problem in biological visualisation. In biological taxonomy visualisation the underlying data structure is a graph consisting of many overlapping trees, where one of the user tasks is to compare their taxonomy with pre-existing taxonomies. In 2000, we developed two visualisations to support comparison of multiple taxonomies, one was a force directed graph layout, where the user could add

as many of the taxonomies as desired. Taxonomies in the graph are identified by having different coloured edges and nodes contain coloured marks to show which taxonomies they belong to. The graph layout suffered from hairball issues therefore we introduced search and filter mechanisms to assist in understanding for the user. However, the users still found the graph layout difficult to comprehend due to the inability to easily identify and separate individual taxonomies and the difficulty in seeing the top down layout of the taxonomies. We also developed a small multiples visualisation with icicle plots representing each taxonomy with linking and focus & context techniques for exploring and comparing the taxonomies. The tool supported removal of ranks in the taxonomies to ease comparison of trees by forcing similar tree structures across the taxonomies. This approach was much preferred by the taxonomists. We then developed a combined approach based on a directed acyclic graph, which maintained the top down layout of the taxonomies, where the user could highlight one taxonomy in the context of the other taxonomies to easily show the differences.

More recently we have been addressing the problem of comparison of multiple networks faced by computational biologists trying to determine best network models for a range of biological networks such as gene interaction networks to ecological networks. In determining the biological network the computational biologists make use of Bayesian network inference algorithms to generate 100s-100s of candidate networks which are given a score based on their fit to the underlying model. The biologists need to examine and compare these hundreds of networks to understand the scores assigned to the networks, e.g. to determine if networks with similar scores are similar or different. If similar then it is likely that the highest scoring network will be the best, however if different then the user might want to generate a consensus network from a range of networks selected by the user. This problem concerns comparison of many small to medium networks rather than one or a few large networks. We have created Bayespiles which is adapted from small multiples, a matrix based visualisation of many networks which piles and summarises networks. BayesPiles enables the exploration, organisation and comparison of hundreds of scored directed networks from multiple heuristic search runs. It features two matrix-based representation modes for directed networks (top-down and diamond), a normalised histogram that shows the distribution of scores in the solution space, flexible network ordering based on run ID, iteration or score, node reordering, interactive comparison of networks across groups, support for the manual construction of a consensus network, interactive graph filtering mechanisms and a summary view of all outgoing edges for selected nodes.

There remain many challenges in comparing networks including scalability to many large networks and comparing multilayer and multidimensional networks.

7.5 Collaboration Panel: Visualization and Visual Analysis of Biomolecular Structures

Barbora Kozlíková (Masaryk University – Brno, CZ)

License  Creative Commons BY 3.0 Unported license
© Barbora Kozlíková

In my five-minute talk I am introducing my experience in collaboration with protein engineering research group. With its members, we are focusing on analysis and visualization of protein structures, namely searching for tunnels connecting the protein outer environment with its active site. Such tunnels can be subsequently used for transportation of ligands to

the active site. In my talk I am stressing the importance of finding the common language with the domain experts and developing mutual trust. I also discuss different publication possibilities and options, as well as issues related to transferring the research results into practice, which requires more engineering and management skills than the research ones.

7.6 Curriculum Panel: Designing a curriculum for teaching visualization in bioinformatics


Michael Krone (Universität Stuttgart, DE)

License  Creative Commons BY 3.0 Unported license
© Michael Krone

My talk focusses on the requirements and contents of a curriculum (or course) for teaching visualization in bioinformatics. I will present my personal recommendation for core visualization principles that should be taught in the context of biological visualization. Key aspects that have to be considered are the background of the intended audience (computer science, bioinformatics, biology) and the level of their studies (bachelor, master, PhD). Students also need to learn certain technical skills in order to be able to create their own visualizations (e.g., programming). Libraries like D3 for non-spatial data or Three.js for spatial data can be powerful tools that lessen the programming burden. This leads to the question of how to teach students to use these tools efficiently in a reasonably short time. In summary, I think that teaching basic visualization concepts is more important than teaching using tools.

7.7 Curriculum Panel: Vis is a large number of small problems


Martin Krzywinski (BC Cancer Research Centre – Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Martin Krzywinski

An effective way to teach visualization is to break a visualization task or challenge into a large number of small problems. Many of these small problems recur and for each there is a relatively small number of ways in which well-meaning users get it wrong. This talk demonstrates redesign examples of typical visualizations from biology and demonstrates the similarity across users' missteps in the context of this kind of divide-and-conquer strategy.

7.8 Biological Networks Panel: Scaffolding Bionetwork Visualization with models and theories

Georgeta Elisabeta Marai (University of Illinois – Chicago, US)

License  Creative Commons BY 3.0 Unported license
© Georgeta Elisabeta Marai

Four years ago, I joined a fearless research group at the Electronic Visualization Laboratory, who have the wisdom to question all existing paradigms. In that spirit, I question three definitions and paradigms for biological networks. First, a biological network is any network that applies to biological systems – for example, a functional network in the mouse audio

cortex is still a biological network. Second, some biological networks have spatial components – even when those components are not anchored in the physical (e.g., image) space, they bear meaning to the biologist. Third, in terms of principles that should guide the selection of a visualization technique for biological networks, “overview-first” is not the only possible design approach. There is also “Search-first” (van Ham and Perer 2009), “Details-first”, and so on.

7.9 Pan-Genomics Panel: Some questions and challenges about comparative genomics and pan-genomics

Kay Katja Nieselt (Universität Tübingen, DE)

License  Creative Commons BY 3.0 Unported license
© Kay Katja Nieselt

In my short panel talk I briefly outline some of the questions and challenges in pan-genomics. I start first with a generalized definition of a pangenome. Based on that I point out that pangenomics has and will influence a number of both traditional viewpoints in biology in the future, such as the definition of a species. One main point is the data structure that represents a pangenome, which depends on its definition as well as context that it is studied in. Depending on the data structure, different visualisations are needed that biologists would want to see when studying pangenomes. There are many algorithmic as well as visualisation challenges in this field, such as scalability and update, which hopefully will be addressed during this seminar.

7.10 High-Dimensional Medical Data Panel: High-Dimensional Medical Data

Jos B.T.M. Roerdink (University of Groningen, NL)

License  Creative Commons BY 3.0 Unported license
© Jos B.T.M. Roerdink

In my presentation I will focus on a number of aspects when dealing with High-Dimensional Medical Data, such as:

How to collaborate? Important issues are spending time with collaborators, using simple visualizations with explanations, avoiding sophistication and information overload, and the need to build trust.

How to integrate into existing, complex data ecosystems? This is generally not possible in medicine, because of certification issues. But integrating tools in systems of medical researchers is. It is important to find a liaison person.

Who are the key influencers in this field? This concerns first of all people with a genuine interest who want to invest time and energy, funding agencies, societal drives.

Cross-cutting interests shared by other communities. High on my list are comparison of visualizations, workflows, and provenance.

7.11 Biological Networks Panel: Visualisation of Biological Networks: Past, Present, and Future

Falk Schreiber (Universität Konstanz, DE)

License © Creative Commons BY 3.0 Unported license
© Falk Schreiber

Networks play an important role in the life sciences. Networks can represent data and processes from chemistry (e.g. chemical structure graphs) to molecular biology (e.g. metabolic networks, co-expression networks) to ecology (e.g. food webs, animal behaviour networks) to medicine (e.g. infection networks) to other related areas. In the first part, this talk will present the history and state-of-the-art of network visualisation (layout) with a focus on metabolic networks. Here we will discuss benefits and disadvantages of common layout algorithms often used to visualise biological networks and look at some specific layout methods. The second part of the talk will investigate visualisation-related topics such as graphical standards for biological networks (e.g. SBGN) and visual analytics for biological network exploration and investigation. This presentation will finish with an outlook to future developments such as immersive analytics for biological networks.

References

- 1 O. Kohlbacher, F. Schreiber and M. O. Ward: ‘Multivariate networks in the life sciences’ in A. Kerren, H. C. Purchase and M. O. Ward (eds.), *Multivariate network visualization*, Springer LNCS 8380, 61–73, 2014.
- 2 C. Bachmaier, U. Brandes and F. Schreiber: ‘Biological networks’ in R. Tamassia (ed.), *Handbook of graph drawing and visualization*, Chapman & Hall/CRC Press, 621–652, 2013.
- 3 N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, P. Ghazal, K. Hideya, L. Li, Y. Matsuoka, A. Villéger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn and H. Kitano: ‘The Systems Biology Graphical Notation’ *Nature Biotechnology* 27: 735–741, 2009.

7.12 Pan-Genomics Panel: Scaling Sequence Comparison for Pan & Metagenomics

Danielle Szafrir (University of Colorado – Boulder, US)

License © Creative Commons BY 3.0 Unported license
© Danielle Szafrir

Pangenome and metagenome comparisons require biologists to identify and interpret meaningful similarities and differences between organisms. This comparison problem requires analysis tools to support comparisons as the number and complexity of sequences increase and also introduce new questions unsupported by different tools. Existing sequence comparison tools enable scalability along at most two of these different dimensions. By understanding the needs and computational and visual challenges associated with comparison tasks in pan- and metagenomes, we can begin to create visualizations that support the needs of these analyses along all three dimensions.

7.13 High-Dimensional Medical Data Panel: Living with Algorithms

Cagatay Turkay (City – University of London, GB)

License  Creative Commons BY 3.0 Unported license
© Cagatay Turkay

The analysis and modelling of high-dimensional medical data is relevant for a wide spectrum of users (researchers/clinicians/patients) in different capacities and complexities. Wherever a user stands on this spectrum, due to the complexities that high dimensional data introduces (heterogeneity / sparsity / uncertainty), interacting with algorithms is unavoidable, be it in terms of getting a recommendation or in terms of building explanatory models. The pursuit for interpretable, comprehensible and explainable algorithms is getting interest in several domains currently including machine learning, knowledge discovery and data visualisation focusing on several different application domains. Visualisation has already shown great potential as an expressive and insightful medium with integrated and linked representations of several components of algorithms and engaging different users in various capacities. This talk investigates the different users in the context of high-dimensional medical data, touches upon a number of visual analytics techniques, and discusses a number of challenges that cuts across these different use cases at the intersection of algorithms and users.

References

- 1 Endert, A., W. Ribarsky, C. Turkay, B. L. Wong, Ian Nabney, I. Díaz Blanco, and F. Rossi. “The state of the art in integrating machine learning into visual analytics.” In *Computer Graphics Forum*, vol. 36, no. 8, pp. 458 – 486. 2017.
- 2 Hohman, F. M., Kahng, M., Pienta, R., and Chau, D. H. (2018). Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics*.
- 3 Krause, J., Perer, A., and Ng, K. (2016, May). Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 5686 – 5697). ACM.

8 Acknowledgements

We would like to thank all participants of the seminar for their contributions and lively discussions; we also would like to thank the scientific directorate of Dagstuhl for providing us with the opportunity to organize this seminar. Finally, the seminar would not have been possible without the untiring help of the (scientific) staff of Dagstuhl, including Ms. Annette Beyer, Ms. Heike Clemens and Mr. Michael Gerke.

Participants

- Jan Aerts
KU Leuven, BE
- Katja Bühler
VRVis – Wien, AT
- Sheelagh Carpendale
University of Calgary, CA
- James L. Chen
Ohio State University –
Columbus, US
- Arlene Chung
University of North Carolina –
Chapel Hill, US
- Anamaria Crisan
University of British Columbia –
Vancouver, CA
- Mirjam Figaschewski
Universität Tübingen, DE
- Angus Forbes
University of California at
Santa Cruz, US
- Nils Gehlenborg
Harvard University, US
- Carsten Görg
University of Colorado –
Aurora, US
- David H. Gotz
University of North Carolina –
Chapel Hill, US
- Helena Jambor
TU Dresden, DE
- Jessie Kennedy
Edinburgh Napier University, GB
- Karsten Klein
Universität Konstanz, DE
- Anne Knudsen
University of Calgary, CA
- Barbora Kozlíková
Masaryk University – Brno, CZ
- Michael Krone
Universität Stuttgart, DE
- Martin Krzywinski
BC Cancer Research Centre –
Vancouver, CA
- Alexander Lex
University of Utah –
Salt Lake City, US
- Raghu Machiraju
The Ohio State University –
Columbus, US
- Georgeta Elisabeta Marai
University of Illinois –
Chicago, US
- Lennart Martens
Ghent University, BE
- Ewy Mathé
Ohio State University –
Columbus, US
- Torsten Möller
Universität Wien, AT
- Scooter Morris
University of California –
San Francisco, US
- Cydney Nielsen
BC Cancer Agency –
Vancouver, CA
- Kay Katja Nieselt
Universität Tübingen, DE
- Bruno Pinaud
University of Bordeaux, FR
- James Procter
University of Dundee, GB
- William Ray
Ohio State University –
Columbus, US
- Jens Rittscher
University of Oxford, GB
- Jos B.T.M. Roerdink
University of Groningen, NL
- Timo Ropinski
Universität Ulm, DE
- Ryo Sakai
PharmiWeb Solutions –
Bracknell, GB
- Falk Schreiber
Universität Konstanz, DE
- Christian Stolte
New York Genome Center, US
- Marc Streit
Johannes Kepler Universität
Linz, AT
- Granger Sutton
J. Craig Venter Institute –
Rockville, US
- Danielle Szafir
University of Colorado –
Boulder, US
- Cagatay Turkay
City – University of London, GB
- Michel A. Westenberg
TU Eindhoven, NL
- Blaz Zupan
University of Ljubljana, SI



Normative Multi-Agent Systems

Edited by

Mehdi Dastani¹, Jürgen Dix², Harko Verhagen³, and
Serena Villata⁴

1 Utrecht University, NL, m.m.dastani@uu.nl

2 TU Clausthal, DE, dix@tu-clausthal.de

3 Stockholm University, SE, verhagen@dsv.su.se

4 Laboratoire I3S – Sophia Antipolis, FR, villata@i3s.unice.fr

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18171 “Normative Multi-Agent Systems”. Normative multi-agent systems combine models for multi-agent systems with normative concepts, like obligations, permissions, and prohibitions. As such, they promise to be a suitable model, for example for (regulated) multiagent societies, organizations, electronic institutions, autonomous agent cooperation (with humans-in-the-loop) and much more. The aim of this seminar was to bring together researchers from various scientific disciplines, such as computer science, artificial intelligence, philosophy, law, cognitive science and social sciences to discuss the emerging topic concerning the *responsibility* of autonomous systems. Autonomous software systems and multi-agent systems in open environments require methodologies, models and tools to analyse and develop flexible control and coordination mechanisms. Without them, it is not possible to steer the behaviour and interaction of such systems and to ensure important overall properties. *Normative multi-agent systems* is an established area focussing on how norms can be used to control and coordinate autonomous systems and multi-agents systems without restricting the autonomy of the involved systems. Such control and coordination systems allow autonomous systems to violate norms, but respond to norm violations by means of various sanctioning mechanisms. Therefore it is crucial to determine which agents or agent groups are accountable for norm violations. The focus of this seminar laid on how the responsibility of autonomous systems can be defined, modelled, analysed and computed.

Seminar April 22–27, 2018 – <http://www.dagstuhl.de/18171>

2012 ACM Subject Classification Computing methodologies → Multi-agent systems

Keywords and phrases autonomous systems, control and coordination, norm-based systems, responsibility

Digital Object Identifier 10.4230/DagRep.8.4.72

Edited in cooperation with Tobias Ahlbrecht

1 Executive Summary

Mehdi Dastani (Utrecht University, NL)

Jürgen Dix (TU Clausthal, DE)

Harko Verhagen (Stockholm University, SE)

License © Creative Commons BY 3.0 Unported license
© Mehdi Dastani, Jürgen Dix, and Harko Verhagen

The multi-disciplinary workshop on Normative Multi-Agent Systems attracted leading international scholars from different research fields (e.g. theoretical computer science, programming languages, cognitive sciences, law, and social sciences).



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Normative Multi-Agent Systems, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 72–103

Editors: Mehdi Dastani, Jürgen Dix, Harko Verhagen, and Serena Villata



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The seminar was a blend of talks, discussions and group work. It began on the first day with short “teaser talks” (10 + 5 minutes) related to the main topic of *norms and responsibility*, one given by almost each participant. The talks were meant to be inspiring and thought-provoking, channeling ideas for the following days. While some missed the established procedure with longer talks, the new format was overall very well received and allowed for many different thoughts and concepts to be presented and discussed in relatively short time.

Four working groups formed at the end of the first day for the norm-related topics responsibility, new logics, ethics/values and (machine) learning.

The aim of the group sessions, on the second and fourth day, was to get a shared understanding of the specific topics and to identify future research possibilities. Each group reported back in a plenary session at the end of each group work day, where the groups also tried to establish interconnections between them.

Responsibility. This group discussed how to grasp the very abstract concept of responsibility. A big chunk was dedicated to the formalization of responsibility. Many (vastly different) assumptions were laid out. The problem of “delegating responsibility” was discussed with special intensity. The group (being by far the largest one) split later to discuss different notions of responsibility on the basis of selected examples. A working paper was produced, included in this report under Section 4.1.

New logics. The aim of this group was to find out how to tackle norms and responsibility in terms of logics, especially how new logics for this task could be devised.

Ethics/values. This group discussed the more ethics-oriented aspects of normative systems. Values provide an additional layer for normative reasoning: e.g. “how acceptable is it to violate a given norm?” The group produced a draft of a paper on “The Value(s) of Water” connecting NorMAS to the AI for Good initiative. Work is planned to continue during 2018 resulting in a paper for publication, e.g. in ACM communications or a similar outlet.

(Machine) Learning. The learning group discussed the opportunity of integrating norms and responsibility into machine learning procedures. As those are usually opaque, this presents as a notable challenge. For example, the learning’s input data has to be pre-processed to get a normatively acting system. Also, the learned sub-symbolic system should be enhanced with “regular” symbolic reasoning, which can be better regulated by norms and analysed for responsibility.

The fourth day was further enriched by a brainstorming session to identify possible applications. The subsequent clustering revealed the topics

- **transport**, e.g. smart grid/home, intelligent cars,
- **tools**, e.g. for autonomous service composition, legal reasoning, or supporting software/requirements engineering,
- **climate & agriculture**, e.g. agents negotiating fertilizer and water use, or an app that helps monitoring personal climate-affecting activities,
- **societies**, e.g. norms improving sustainability, monitoring of online forums for bad behavior or hate speech detection,
- **security**, e.g. protecting personal freedom by dynamically analysing normative consequences of law proposals, monitoring a company’s compliance with EU regulations, improving access to restricted access datasets, or making societies resilient for data surveillance by means of contract negotiations,

- **health**, e.g. ethical decision-making, norms for improving personal health and fitness, defining wellbeing by norms, handling of patient/health data, and a big interest in healthcare robots,
- **energy**, e.g. modelling energy security with norms, managing air quality, observing long-term consequences, agents monitoring (personal) energy use to identify bad behavior, or regulating industrial relations or the energy and material footprint.

The application areas were discussed in a plenary session and formed the input to the discussion on future plans for the NorMAS community. Several conferences were identified to target proposals for a NorMAS-related workshop as part of the event. The community sees many relevant application areas not in the least in autonomous internet services and physical agents such as robots, vehicles and drones, where social reasoning will be of the utmost importance. Bringing the work from NorMAS to these areas will be highly beneficial to the involved communities.

2 Contents

Executive Summary

Mehdi Dastani, Jürgen Dix, and Harko Verhagen 72

Overview of Talks

Norms in the Multi-Agent Programming Contest
Tobias Ahlbrecht 76

Causality, Responsibility and Blame in Team Plans
Natasha Alechina, Joseph Halpern, and Brian Logan 76

Overview of Legal Liability of Autonomous Systems and Implications for Norms-based Systems
Kevin D. Ashley 76

On the role of accountability in programming MAS
Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio 77

A Formalisation of Moral Responsibility and the Problem of Many Hands
Tiago de Lima 79

Supervising Autonomous Systems
Davide Dell’Anna 80

Isabelle/HOL: a Computational Framework for Normative Reasoning
Ali Farjami, Christoph Benz Müller, and Xavier Parent 80

Natural Strategic Ability
Wojtek Jamroga 81

Programming Responsibility in Norm-Aware Agents
Brian Logan 81

Simulating the hermeneutics of irresponsibility
Martin Neumann 81

Anchoring Electronic Institutions
Pablo Noriega, Julian Padget, and Harko Verhagen 82

Rule Based SLAs for Water (RBSLA4Water)
Adrian Paschke 83

Goal-based Argumentation for Intelligent Deliberation
Douglas Walton 84

Trust, Responsibility, and Explanation
Michael Winikoff 84

Group Responsibility Under Imperfect Information
Vahid Yazdanpanah 85

Working groups

Formal definitions of responsibility
Natasha Alechina, Tiago de Lima, Brian Logan, Ken Satoh, and Douglas Walton 85

Participants 103

3 Overview of Talks

3.1 Norms in the Multi-Agent Programming Contest

Tobias Ahlbrecht (TU Clausthal, DE)

License © Creative Commons BY 3.0 Unported license
© Tobias Ahlbrecht

I briefly present the Multi-Agent Programming Contest, a competition attempting to stimulate research in the area of multi-agent system development and programming. I will touch on its potential for norm usage and evaluation and vice versa, with regard to the opportunity of incorporating norms in the next scenario.

3.2 Causality, Responsibility and Blame in Team Plans

Natasha Alechina (University of Nottingham, GB), Joseph Halpern, and Brian Logan (University of Nottingham, GB)

License © Creative Commons BY 3.0 Unported license
© Natasha Alechina, Joseph Halpern, and Brian Logan
Joint work of Natasha Alechina, Joseph Halpern, Brian Logan
Main reference Natasha Alechina, Joseph Halpern, Brian Logan: “Causality, Responsibility and Blame in Team Plans”, in Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017, pp. 1091–1099, ACM, 2017.
URL <http://dl.acm.org/citation.cfm?id=3091279>

Many objectives can be achieved (or may be achieved more effectively) only by a group of agents executing a team plan. If a team plan fails, it is often of interest to determine what caused the failure, the degree of responsibility of each agent for the failure, and the degree of blame attached to each agent. In the talk, I will show how team plans can be represented in terms of structural equations, and how the definitions of causality introduced by Halpern (2015) and degree of responsibility and blame introduced by Chockler and Halpern (2004) can be applied to determine the agent(s) who caused the failure and what their degree of responsibility/blame is. I will present results on the complexity of computing causality and degree of responsibility and blame, which show that they can be determined in polynomial time for many team plans of interest. The talk is based on joint work with Joseph Halpern and Brian Logan.

3.3 Overview of Legal Liability of Autonomous Systems and Implications for Norms-based Systems

Kevin D. Ashley (University of Pittsburgh, US)


License © Creative Commons BY 3.0 Unported license
© Kevin D. Ashley

Autonomous systems present novel circumstances for assessing legal liability. Autonomous vehicles, for instance, promise to increase traffic safety overall. Inevitably, however, such vehicles will also cause accidents injuring people and property, and the providers of such vehicles and their component software systems will be subject to law suits on behalf of victims. This talk briefly surveys how the American law of product liability and negligence

would address such scenarios and highlights some potentially interesting practical and legal differences between a machine learning versus a norms-based architecture when autonomous vehicles cause accidents. The legal framework could lead to a discussion to elicit more details about the norms-based and machine learning architectures in order to explore in greater depth these potential practical and legal differences where the ML-based perceptual system and the norms-based reasoner meet.

3.4 On the role of accountability in programming MAS

Matteo Baldoni (University of Turin, IT), Cristina Baroglio, and Roberto Micalizio

License  Creative Commons BY 3.0 Unported license
© Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio

Multiagent Systems (MAS) represents a viable programming paradigm for the development of complex systems characterized by multiple threads of execution that run in parallel. Most of the design methodologies and programming platforms that have been proposed in the literature (e.g., OperA [8], OMNI [9], OCEAN [12], 2OPL [7], JaCaMo [3], and [11]) are grounded on the metaphor of the *organization*: The system under development is seen as a human-like organization where organizational goals, possibly decomposed into subgoals, are distributed to agents playing organizational roles. A set of norms rule the admissible interactions among agents within the organization. Such a normative system issues obligations, permissions, and prohibitions as a consequence of what agents do within the organization. Notably, obligations and the like do not require any acceptance by the agents. Indeed, obligations are the means through which an organization stimulates the agents to perform some tasks. Of course, agents, inasmuch autonomous entities, can decide whether to satisfy an obligation or violate a prohibition. Thus, in order to enforce the norm-specified, desired behavior some sanctioning mechanism is often introduced. The idea is that a rational agent will satisfy obligations to avoid sanction.

We deem that the organizational metaphor is a very effective way to approach the design and development of complex systems, but the current formalizations are still incomplete in properly capturing the notion of organization from a software engineering point of view. Current approaches, in fact, strongly depend on *obligations* for getting tasks done, but this imposes some, often unspoken, assumptions. First, since an obligation towards an agent is satisfied when the agent activates a proper behavior, it is assumed that the agent has necessarily a proper behavior for each obligation it will ever receive. This assumption is easily satisfied only when the set of goals that can be assigned to an agent are known in advance and do not change over time. But goals are dynamic by nature, and hence it may be possible that when the normative system issues an obligation towards an agent, that agent does not have a proper behavior for satisfying that obligation. We have demonstrated this in the context of JaCaMo platform (see [2]). Second, it is assumed that the sanctions associated with the violation of an obligations are a sufficient tool for conditioning agents' behaviors. Agents, however, can deliberately decide to violate an obligation despite the sanction, and will do so in those cases when the obligation does not match the agent's goals and the sanction is acceptable. Thus, obligations may fall short in stimulating agents doing tasks, either because they can be directed to agents that do not possess the proper capabilities, or because the sanction is not an absolute criteria for an agent to decide how to act.

It is interesting to note that such shortcomings of obligations are well-known and widely accepted in sociology (see, for instance, [10, 15, 13]). In social terms, an agent *voluntarily* triggers an act only if that act is *desirable* for the agent. Therefore, normative sanctions often have little consequence on the agent, and no consequence at the society level. It was also observed in the requirements engineering field [6] that agents' obedience to the system norms cannot be taken for granted. Agent autonomy demands a different way of conceptualizing software modularity: not in terms of subgoals that are assigned to the agents, but rather in terms of responsibilities that are explicitly taken on by the agents. This last observation concerns also approaches that, instead of relying on norms/obligations, rely on social commitments [5, 16]. On the one side, the creation of a commitment is a deliberate act of the agent that takes on a duty. On the other side, however, a detached commitment is a directed obligation from the debtor to the creditor of the commitment. As such, an agent can violate its commitments when it deems advantageous to do so. We deem that a commitment is still inadequate for modeling "responsibilities" in a way that can be exploited from a software engineering perspective. In fact, agents could create commitments to bring about conditions that are not completely under their control. In these cases, sanctioning an agent that has not satisfied a commitment is of little help.

In this paper we argue that the current models for supporting agent interaction and coordination – norm/obligation-oriented, as well as commitment-oriented – should be complemented in some way. We found support to our intuition in the literature from the areas of sociology (and in particular ethnomethodology) and from political sciences, identifying in *accountability* the key missing concept. Starting from sociology, citing [4]: "Garfinkel developed the idea of the accountable character of action to emphasize that social action is organized so that it can be reported and described. In other words, *people design social actions so that others can see and say what those actions are*. For ethnomethodologists, accountability is a pervasive feature of how people co-ordinate their actions." Instead, from political sciences [1]: if an individual is accountable, that accountability will act as a constraint on their decision process. Holding people accountable means asking them to explain their actions, especially when they fail to bring about expected goals. Accountability is therefore the underlying force that influence actions in human organizations, and more generally, in human relationships.

On the software side, accountability can be a powerful tool for motivating better practices, and consequently more reliable and trustworthy systems [14]. Our intuition is that accountability can be understood as a software engineering element that helps a designer devise a complex system and, at the same time, can be the base for handling exceptions at run time in a more effective way than of an obligation/sanction mechanism. In this ongoing work we explore the possibility to found the realization of distributed systems on the two basic notions of responsibility and accountability, tracing connecting points with more traditional approaches, and tracing also directions of research that we deem significant.

References

- 1 Paul A. Anderson. Justifications and precedents as constraints in foreign policy decision-making. *American Journal of Political Science*, 25(4), 1981.
- 2 Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. ADOPT JaCaMo: Accountability-Driven Organization Programming Technique for JaCaMo. In A. Bo, A. Bazzan, J. Leite, L. van der Torre, and S. Villata, editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems, 20th International Conference*, number 10621 in Lecture Notes in Computer Science, pages 295–312. Springer, 2017.
- 3 Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Sci. Comput. Program.*, 78(6):747–761, 2013.

- 4 Graham Button and Wes Sharrock. The organizational accountability of technological work. *Social Studies of Science*, 28(1):73–102, 1998.
- 5 C. Castelfranchi. Principles of Individual Social Action. In *Contemporary action theory: Social action*, volume 2, pages 163–192, Dordrecht, 1997. Kluwer.
- 6 Amit K Chopra and Munindar P Singh. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *Proc. of the 25th Int. Conf. on WWW*, 2016.
- 7 Mehdi Dastani, Nick AM Tinnemeier, and John-Jules Ch Meyer. A programming language for normative multi-agent systems. In *Handbook of Research on Multi-Agent Systems: semantics and dynamics of organizational models*, pages 397–417. IGI Global, 2009.
- 8 Virginia Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, Utrecht University, 2004. Published by SIKS.
- 9 Virginia Dignum, Javier Vázquez-Salceda, and Frank Dignum. OMNI: introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems, Second International Workshop ProMAS, Selected Revised and Invited Papers*, volume 3346 of *Lecture Notes in Computer Science*, pages 181–198. Springer, 2004.
- 10 Emile Durkheim. *De la division du travail social*. 1893.
- 11 Marc Esteva, Juan-Antonio Rodríguez-Aguilar, Carles Sierra, Pere Garcia, and Josep L. Arcos. On the formal specification of electronic institutions. In Frank Dignum and Carles Sierra, editors, *Agent Mediated Electronic Commerce: The European AgentLink Perspective*, pages 126–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- 12 Nicoletta Fornara, Francesco Viganò, Mario Verdicchio, and Marco Colombetti. Artificial institutions: a model of institutional reality for open multiagent systems. *Artificial Intelligence and Law*, 16(1):89–105, 2008.
- 13 Harold Garfinkel. *Studies in ethnomethodology*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1967.
- 14 Helen Nissenbaum. Accountability in a computerized society. *Science and engineering ethics*, 2(1):25–42, 1996.
- 15 Talcott Parsons. *The Structure of Social Action*. Collier-Macmillan, London, 1968.
- 16 Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

3.5 A Formalisation of Moral Responsibility and the Problem of Many Hands

Tiago de Lima (CNRS - Lens, FR)

License © Creative Commons BY 3.0 Unported license
© Tiago de Lima

In this talk, I present a formalization of the so called problem of many hands. Using the basic concepts upon which the meanings of responsibility are defined, we construct a logic which enables us to express sentences like ‘individual i is accountable for ϕ ’, ‘individual i is blameworthy for ϕ ’ and ‘individual i has the obligation to see to it that ϕ ’. Such effort contributes to the discussion about responsibility in at least two ways. First, it clarifies the definitions and also their differences and similarities. Second, it assesses the consistency of the formalization of responsibility, not only by showing that definitions are not inconsistent, but also by providing a formal demonstration of the relation between three different meanings of the word responsibility. Moreover, the formal account can be used to derive new properties of the concepts, thus, giving new insights that can be used to advance the discussion. And

finally, the formalism proposed here provides a framework wherein criteria for ascribing responsibilities can be stated and, if individuals are to be held responsible for outcomes, then, at least, justifications can be made clear.

3.6 Supervising Autonomous Systems

Davide Dell’Anna (Utrecht University, NL)

License © Creative Commons BY 3.0 Unported license
© Davide Dell’Anna

Joint work of Davide Dell’Anna, Mehdi Dastani, Fabiano Dalpiaz

Norms with sanctions have been widely employed as a mechanism for controlling and coordinating the behavior of agents without limiting their autonomy. The norms enforced in a multi-agent system (MAS) can be revised in order to increase the likelihood that desirable system properties (such as company’s core values or ethical principles) are fulfilled or that system performance is sufficiently high. We provide a description of a supervision system that monitors the execution of a MAS, identifies deviations from the overall system objectives, and with the help of a probabilistic model (Bayesian Network) automatically proposes norm revisions that are expected to increase system objectives achievement. A preliminary experimental evaluation of the effectiveness of the framework on an urban smart transportation simulator is proposed. The experimental results are promising: data retrieved from system execution can be successfully employed to suggest and apply appropriate revisions of norms at runtime, allowing the MAS to reach an adequate satisfaction of the desired overall system objectives.

3.7 Isabelle/HOL: a Computational Framework for Normative Reasoning

Ali Farjami (University of Luxembourg, LU), Christoph Benzmüller (FU Berlin, DE), and Xavier Parent

License © Creative Commons BY 3.0 Unported license
© Ali Farjami, Christoph Benzmüller, and Xavier Parent

Joint work of Christoph Benzmüller, Ali Farjami, Xavier Parent

Main reference Christoph Benzmüller, Ali Farjami, Xavier Parent: “Faithful Semantical Embedding of a Dyadic Deontic Logic in HOL”, CoRR, Vol. abs/1802.08454, 2018.

URL <http://arxiv.org/abs/1802.08454>

We have provided the theoretical foundation for the implementation and automation of dyadic deontic logic within off-the-shelf higher-order theorem provers and proof assistants. We have devised (shallow) semantical embedding of some dyadic deontic logics in classical higher-order logic. The embedding has been encoded in Isabelle/HOL, which turns this system into a proof assistant for deontic logic reasoning. The experiments with this environment provide evidence that these logic implementations fruitfully enables interactive and automated reasoning at the meta-level and the object-level. We built a computational framework, based Isabelle/HOL, for normative reasoning.

3.8 Natural Strategic Ability

Wojtek Jamroga (Polish Academy of Sciences - Warsaw, PL)

License © Creative Commons BY 3.0 Unported license
© Wojtek Jamroga

Joint work of Wojtek Jamroga, Vadim Malvone, Aniello Murano

Main reference Wojtek Jamroga, Vadim Malvone, Aniello Murano: "Reasoning about Natural Strategic Ability", in Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017, pp. 714–722, ACM, 2017.

URL <http://dl.acm.org/citation.cfm?id=3091227>

In game theory, as well as in the semantics of game logics, a strategy can be represented by any function from states of the game to the agent's actions. That makes sense from the mathematical point of view, but not necessarily in the context of human behavior. This is because humans are quite bad at executing complex plans, and also rather unlikely to come up with such plans in the first place. In this work, we adopt the view of bounded rationality, and look only at "simple" strategies in specifications of agents' abilities. I will formally define what "simple" means, and present a variant of alternating-time temporal logic that takes only such strategies into account. I will also briefly point out where it possibly connects with the notion of responsibility.

3.9 Programming Responsibility in Norm-Aware Agents

Brian Logan (University of Nottingham, GB)

License © Creative Commons BY 3.0 Unported license
© Brian Logan

In this talk I will consider the problem of programming group norms specifying that a group of agents are responsible for bringing about some state, i.e., a group obligation. The group norm specifies what should be achieved, by when, and the sanction for the group if the norm is violated, but not the responsibilities of each agent to bringing about the desired state or individual sanctions in the event of a violation. As such they provide a degree of abstraction that is critical for the implementation of many large normative MAS. However group norms introduce several new programming challenges, in particular the delegation of responsibility for norm enforcement from the MAS to an agent or agents within the group. I will present an approach to implementing group-norm-aware agents that are able to deliberate on their individual goals, group norms and sanctions when deciding whether to participate in a team plan.

3.10 Simulating the hermeneutics of irresponsibility

Martin Neumann (Jacobs University Bremen, DE)

License © Creative Commons BY 3.0 Unported license
© Martin Neumann

In the talk I will approach the issue of responsibility from the reverse angle by investigating corruption as a manifestation of irresponsibility. Corruption is a phenomenon of misuse of a position of trust. As case the Ukraine is selected, which is characterized by a current high level of corruption. The project addresses the question of how civil society can be organized in the

interplay of political and legislative institutions and cultural dimensions of civil engagement. Addressing the perception of (ir)responsible fulfillment of social roles during interactions need to take a cultural dimension into account. This requires socio-cognitive coupling of how participants make sense of the phenomenology of a situation from the perspective of their worldview. For this purpose a methodology will be applied that has been developed in the previous project GLODERS integrating qualitative content analysis, agent-based simulation, and narrative analysis of simulation results. Central feature is preserving traceability to the empirical evidence throughout the research process. Traceability enables interpretation of simulations by generating a narrative storyline of the simulation. Thereby simulation enables a qualitative exploration of textual data. The whole process generates a thick description of the subject of study. Simulation results generate virtual narratives by decomposing and rearranging the empirical in-vivo codes. This can be described as an exploration of the horizon of the space of cultural possibilities. The talk will outline work in progress and I hope for stimulating feedback at an early stage of research.

3.11 Anchoring Electronic Institutions

Pablo Noriega (IIIA - CSIC - Barcelona, ES), Julian Padget (University of Bath, GB), and Harko Verhagen (Stockholm University, SE)

License © Creative Commons BY 3.0 Unported license

© Pablo Noriega, Julian Padget, and Harko Verhagen

Joint work of Pablo Noriega, Harko Verhagen, Mark d’Inverno, Julian Padget

Main reference Pablo Noriega, Harko Verhagen, Mark d’Inverno, Julian Padget: “A Manifesto for Conscientious Design of Hybrid Online Social Systems”, in Proc. of the Coordination, Organizations, Institutions, and Norms in Agent Systems XII - COIN 2016 International Workshops, COIN@AAMAS, Singapore, Singapore, May 9, 2016, COIN@ECAI, The Hague, The Netherlands, August 30, 2016, Revised Selected Papers, Lecture Notes in Computer Science, Vol. 10315, pp. 60–78, Springer, 2016.

URL http://dx.doi.org/10.1007/978-3-319-66595-5_4

Online Institutions capture the three main features that characterise classical institutions: (i) they are a set of artificial constraints that articulate human interactions (North); (ii) they are a regulated social space where institutional actions and facts take place (Searle); and (iii) they are coordination artefacts that constitute an interface between the individual decision-making models of agents and a collective activity they pursue (Simon). They can be understood as socio-cognitive technical systems in as much as all interactions happen online, and the agents that participate in them may be natural or artificial entities endowed with some form of social rationality. Moreover they are normative multiagent systems because, actually, only those interactions that comply with –enforced– institutional norms may have an institutional effect.

In this paper we are concerned with a very practical problem: what one has to take into account so that an online institution works effectively in the real world (in the sense that attempted actions, only when deemed institutionally admissible, produce the actual intended effects). We approach this question in two steps: first we discuss how an abstract isolated institution may be anchored and then we extend the discussion to institutions that are situated in a wider and changing socio-technical environment.

For our discussion we build on the “WIT framework” that represents an institution as three interconnected views (working, institutional and technological), and the requirements for “conscientious” design (thoroughness, mindfulness and responsibility) [1].

The use of the WIT framework allows for a separation of concerns implicit in the design and implementation of a given electronic institution. Thus we inspect the pragmatical

requirements of the three views and their pair-wise relationships, and elucidate what needs to be satisfied in order to guarantee that the online institution functions properly.

References

- 1 Pablo Noriega, Harko Verhagen, Mark d’Inverno, and Julian Padget. A manifesto for conscientious design of hybrid online social systems. In Stephen Cranefield, Samhar Mahmoud, Julian Padget, and Ana Paula Rocha, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems XII - COIN 2016 International Workshops, COIN@AAMAS, Singapore, Singapore, May 9, 2016, COIN@ECAI, The Hague, The Netherlands, August 30, 2016, Revised Selected Papers*, volume 10315 of *Lecture Notes in Computer Science*, pages 60–78. Springer, 2016.

3.12 Rule Based SLAs for Water (RBSLA4Water)

Adrian Paschke (FU Berlin, DE)

License © Creative Commons BY 3.0 Unported license
© Adrian Paschke

Joint work of George Iordache, Adrian Paschke, Mariana Mocanu, Catalin Negru
Main reference George Iordache, Adrian Paschke, Mariana Mocanu, Catalin Negru: “Service Level Agreement Characteristics of Monitoring Wireless Sensor Networks for Water Resource Management (SLAs4Water)”, in *SIC Journal (Studies in Informatics and Control)*, Special Issue Advanced Services in Heterogeneous Distributed Systems, vol. 26(4), pp. 379–386, 2017
URL <https://doi.org/10.24846/v26i4y201701>

Water monitoring infrastructures use various components such as supervisory, control and data acquisition systems, wireless sensors or smart meters producing data in different formats and scales. [1] One of the most important characteristics of a Service Level Agreement (SLA) or executable Smart Contract when discussing about Monitoring Wireless Sensor Networks (MWSNs) is its effectiveness in assuring business success, a high provider profit, an increased level of client satisfaction and trust. In order to ensure that these goals will be achieved the provider of the MWSN must define several parameters [2] that characterize the Service Level Agreement between the MWSN provider and the MWSN customer. The characteristics of the SLA in place between the MWSN provider and the MWSN customer must be defined by taking into consideration various parameters that are particular to the MWSN such as routing algorithms, recovery from failure, monitoring and reporting aspects. This talk addresses a solution for an efficient and effective Service Level Agreement (SLA) design [3] and an implementation that applies a Rule-based SLA (RBSLA) solution [4], implemented by distributed Provalet agents [5], for the automated monitoring and enforcement of the service level objectives in the case of water resources management. The underlying logic applies the ContractLog knowledge representation [4, 6] and the Rule Based Service Level Agreement RuleML language [7].

References

- 1 EU H2020 Data4Water project - D1.1 Technology survey: Prospective and challenges. <http://data4water.pub.ro/mod/book/tool/print/index.php?id=104>, accessed Feb. 2018.
- 2 Paschke, A., Schnappinger-Gerull, E. A Categorization Scheme for SLA Metrics. Multi-Conference Information Systems (MKWI06), Passau, Germany, 2006.
- 3 George Iordache, Adrian Paschke, Mariana Mocanu and Catalin Negru. Service Level Agreement Characteristics of Monitoring Wireless Sensor Networks for Water Resource Management (SLAs4Water). In *SIC Journal (Studies in Informatics and Control)*, Special Issue Advanced Services in Heterogeneous Distributed Systems, 11/2017.

- 4 Adrian Paschke, Martin Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1): 187-205 (2008)
- 5 Adrian Paschke. Provalets – Component-based Mobile Agents for Rule-based Data Access, Processing and Analytics. In Special Issue on Linked Data in Business in *Journal of Business & Information Systems Engineering (BISE)*, 5/2016.
- 6 Paschke, A., Bichler, M., Dietrich, J. ContractLog: An Approach to Rule Based Monitoring and Execution of Service Level Agreements. *International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML 2005)*, Galway, Ireland, 2005.
- 7 Adrian Paschke. RBSLA – Rule based Service Agreements. *Rule-based Contract Representation and Management for electronic Contracts, Policies and Service Level Agreements*, <http://rbsla.ruleml.org/>, accessed Feb. 2018.

3.13 Goal-based Argumentation for Intelligent Deliberation

Douglas Walton (University of Windsor, CA)

License © Creative Commons BY 3.0 Unported license
© Douglas Walton

Joint work of Douglas Walton, Alice Toniolo, Timothy J. Norman

Main reference Douglas Walton, Alice Toniolo, Timothy J. Norman: “Towards a richer model of deliberation dialogue: Closure problem and change of circumstances”, *Argument & Computation*, Vol. 7(2-3), pp. 155–173, 2016.

URL <http://dx.doi.org/10.3233/AAC-160009>

This paper surveys some recent work in argumentation theory on the problem how a group of autonomous intelligent agents can use goal-based defeasible reasoning in a normative dialogue setting to arrive rationally at a conclusion on what is the best thing to do in a changing set of circumstances requiring action. Resources from argumentation studies (an interdisciplinary field) are shown to be useful for current research of how to model arguments about responsibility in multiagent systems. Argumentation-based models of intelligent deliberation dialogue are shown to be useful for developing autonomous systems that support human practical (goal-based) reasoning. Having an open knowledge base enabling new evidence to be taken in as the deliberation proceeds is shown to be an important feature if the system is to model realistic deliberation.

3.14 Trust, Responsibility, and Explanation

Michael Winikoff (University of Otago, NZ)

License © Creative Commons BY 3.0 Unported license
© Michael Winikoff

Joint work of Michael Winikoff, Virginia Dignum, Frank Dignum

My talk considered the overarching issue of trusting autonomous systems, and the factors that lead to appropriate levels of trust in autonomous systems. I particularly focussed on the role of explanation, and described an explanation mechanism and its evaluation.

3.15 Group Responsibility Under Imperfect Information

Vahid Yazdanpanah (*University of Twente, NL*)

License © Creative Commons BY 3.0 Unported license
© Vahid Yazdanpanah

Joint work of Vahid Yazdanpanah, Mehdi Dastani, Wojciech Jamroga

A major issue in autonomous multi-agent systems is to determine who bears the responsibility for avoiding the occurrence of undesirable events. In this work, we take a forward-looking approach and model responsibility based on agents' preclusive power with respect to a given state of affairs. While some recent contributions tackled the issue under the perfect information assumption, we look at the broader picture, and provide operational semantics for reasoning about responsibility under imperfect information.

4 Working groups

4.1 Formal definitions of responsibility

Natasha Alechina (*University of Nottingham, GB*), Tiago de Lima (*CNRS - Lens, FR*), Brian Logan (*University of Nottingham, GB*), Ken Satoh (*National Institute of Informatics - Tokyo, JP*), and Douglas Walton (*University of Windsor, CA*)

License © Creative Commons BY 3.0 Unported license
© Natasha Alechina, Tiago de Lima, Brian Logan, Ken Satoh, and Douglas Walton

4.1.1 Introduction

The aim of this working paper is to investigate how responsibility may be formalised.¹ We consider four formalisms, modal logic, a logic of strategic ability, causal models and formal arguments, and for each formalism, we show how responsibility for an event or state of affairs can be formalised in two simple settings. We focus on responsibility for violations of a norm, specifically responsibility for failure to discharge an obligation.

4.1.2 The simplest case

We begin by considering the simplest case, where an agent is obliged to perform an action² and only that agent acts.

► **Example 1.** A plant must be watered in order to prevent it dying. Agent 1 has an obligation to water the plant. Agent 1 does not water the plant. The plant dies. Who is responsible for the death of the plant? Who is responsible for the violation of the obligation?

In this simple setting, responsibility for the state of affairs, and responsibility for violation of the norm coincide. In the remainder of this section, we show how responsibility can be modelled in each of the four formalisms we consider.

¹ This working paper can be seen as the report of a working group on formalising responsibility in normative multi-agent systems that formed part of Dagstuhl Seminar 18171 *Normative Multi-Agent systems* held at Schloss Dagstuhl in April 2018.

² Or bring about a state of affairs; in this simple example, where there is a single action that brings about a state of affairs, the two notions coincide.

4.1.2.1 Modal logic with three modalities [12]

In this section, we formalise responsibility in a modal logic with three modalities: knowledge K , obligation O and possibility (executability of an action) \Diamond . Essentially we need to be able to say that the agent knows that it has an obligation to keep the plant alive, it knows causal dependency between watering and the plant being alive, and it knows that it is able to water the plant. Knowledge is veridical, so the statements the agent knows indeed hold. The following statements describe legal requirements for the intensional responsibility of agent 1 for the death of the plant, and the violation of the obligation (w is watering, d is plant is dead):

facts $w \rightarrow \neg d, \neg w, d$ (objective causality and objective facts)³

knowledge of obligation $K(O\neg d)$

knowledge of capability $K\Diamond w$

understanding of what the agent is doing $K\neg w$

knowledge of causality $K(w \rightarrow \neg d)$

Unintentional violation (error/negligence): when instead of knowledge of causality $K(w \rightarrow \neg d)$ we have $O(K(w \rightarrow \neg d)) \wedge \neg K(w \rightarrow \neg d)$.

4.1.2.2 Logic of strategic ability [3]

In this section, we consider the formalism Coalition Epistemic Dynamic Logic (CEDL) as proposed in [4, 3]. It is a propositional multi-modal logic whose language is built using a countable set \mathbb{P} of propositional variables, a finite set \mathbb{N} of agent names and a finite set \mathbb{A} of action names. A joint action is defined as a total function $\delta : \mathbb{N} \rightarrow \mathbb{A}$. A partial joint action $\delta|_G$ is defined as the set $\{(i, a) \mid i \in G \text{ and } (i, a) \in \delta\}$. In addition to the usual connectives \neg and \wedge , the logic also has a modal operator for knowledge and another one for actions. A formula of the form $K_G\varphi$ means ‘the group of agents G knows that φ ’ (distributive knowledge). A formula of the form $[\delta]\varphi$ means ‘after all possible executions of δ , it is the case that φ ’. In this case, the idea is that each agent in \mathbb{N} execute its corresponding action in δ simultaneously. The language also permits the use of partial joint actions $a|_G$. Thus, a formula of the form $[a|_G]\varphi$ is also possible. In this case the idea is that each agent in G execute its corresponding action in δ simultaneously and we do not consider what the other agents in $\mathbb{N} \setminus G$ are doing. We have as its meaning ‘after all possible executions of $a|_G$ by the group of agents G and whatever the agents in $\mathbb{N} \setminus G$ do, it is the case that φ ’.

The models of this logic are structures of the form $M = \langle W, \{R_i \mid i \in \mathbb{N}\}, \{T_\delta \mid \delta : \mathbb{N} \rightarrow \mathbb{A}\}, \{V_p \mid p \in \mathbb{P}\} \rangle$, where W is a non-empty set of possible worlds; Each $R_i \subseteq W \times W$ is the indistinguishability relation of the agent i ; Each $T_\delta \subseteq W \times W$ is the transition relation of the joint action δ ; Each $V_p \subseteq W$ is a valuation function for p . In addition, we require that these models satisfy some constraints, for instance to ensure that it grasps correctly the concepts of knowledge and actions.

³ Expressing causality as ‘watering causes the plant to be alive’ versus ‘no watering causes plant’s death’ is more in line with the legal reasoning. In Japanese criminal law, at least in negligence cases, the relevant question is what is the duty of care to avoid damage, and the decision is whether the person violates the duty or not. In this sense, causality which mentions how to avoid damage would be more appropriate.

The satisfaction relation is the usual one for the classical connectors plus:

$$M, w \models K_G \varphi \quad \text{iff} \quad \text{for all } w' \in \bigcap_{i \in \mathbb{N}} R_i(w) \text{ we have } M, w' \models \varphi$$

$$M, w \models [\delta|_G] \varphi \quad \text{iff} \quad \text{for all } \delta' \text{ and all } w' \in T_{\delta|_G \cup \delta'|_{\mathbb{N} \setminus G}}(w) \text{ we have } M, w' \models \varphi$$

To be able to defined responsibility, we need some operators which are defined via abbreviations.

Ensuring

The formula $E_{\delta|_G} \varphi$ means ‘by executing $\delta|_G$, the group G ensures that φ ’. This operator is defined as an abbreviation:

$$E_{\delta|_G} \varphi \stackrel{\text{def}}{=} \neg[\delta|_G] \perp \wedge [\delta|_G] \varphi$$

In other words, the action δ is executable and every possible execution of it by G leads to a state where φ is true.

Ability

The formula $\langle\langle G \rangle\rangle \varphi$ means ‘group G is able to ensure φ ’. This is defined as:

$$\langle\langle G \rangle\rangle \varphi \stackrel{\text{def}}{=} \bigvee_{\delta} E_{\delta|_G} \varphi$$

In other words, there is an executable action δ such that its execution by G leads to a state where φ is true. (Note that the set of all joint actions δ is finite.)

Knowing how ability

The formula $H_G \varphi$ means ‘the group G knows how to ensure φ ’. This defined as follows:

$$H_G \varphi \stackrel{\text{def}}{=} \bigvee_{\delta} K_G E_{\delta|_G} \varphi$$

Obligations

To be able to express obligations, we add a set \mathbb{V} of violations to the logic. This set contains variables vio_G meaning ‘violation for the group G ’. Then, the formula $O_G \varphi$ means ‘it is obligatory for the group G that φ is true’, which is defined as:

$$O_G \varphi \stackrel{\text{def}}{=} \neg \varphi \rightarrow vio_G$$

Knowing causality

The formula $C_{\delta|_G} \varphi$ means ‘the group G knows that the execution of $\delta|_G$ causes φ ’. It is defined by:

$$C_{\delta|_G} \varphi \stackrel{\text{def}}{=} K_G E_{\delta|_G} \varphi \wedge \neg \langle\langle \emptyset \rangle\rangle \varphi$$

If we follow all the definitions above, we find that knowing causality amounts to group G knows that action δ is executable and its execution always lead to a state where φ is true and, in addition, it is not the case that φ is inexorably true in the next state.

Responsibility

Forward-looking responsibility is also defined as an abbreviation:

$$R_G\varphi \stackrel{\text{def}}{=} O_G H_G \varphi \wedge \langle\langle\emptyset\rangle\rangle O_G \varphi$$

Forward-looking responsibility is then defined as the obligation to have the ability to ensure φ plus the obligation that φ is true in the next state.

The definition of backward-looking responsibility, also called blame, given in [3] is based on the operators C and R. That definition can be considered a “prudent” one. Indeed, agents are blamed for φ when they knowingly cause φ . This is not enough if one wants to blame agents for outcomes that result from negligence (such as in Example 5, page 94). In this case, the following definition may be more appropriate:

$$B_{\delta|_G}\varphi \stackrel{\text{def}}{=} \neg C_{\delta|_G} \neg \varphi \wedge R_G \neg \varphi$$

In this definition, group G is blamed for φ if and only if G does not avoid the undesired outcome φ but had the forward-looking responsibility to avoid it.

Now, let us finally model Example 1 in this logic. We need one propositional variable, one agent and two actions. Let $\mathbb{P} = \{d\}$, where d means “the plant is dead”. In addition, let $\mathbb{N} = \{1\}$ and let $\mathbb{A} = \{nop, water\}$. The set of joint actions contains:

$$\alpha = \{(1, nop)\}$$

$$\beta = \{(1, water)\}$$

Now, assume a model satisfying the following formulas:

$$K_1(\neg[\alpha]\perp \wedge [\alpha]d)$$

$$K_1(\neg[\beta]\perp \wedge [\beta]\neg d)$$

$$R_1 \neg d$$

The first formula means ‘agent 1 knows that α is executable and its execution leads to a state where the plant is dead’. The meaning of second one is similar. The third formula means ‘1 is forward-looking responsible for the plant is not dead’.

Because there is an action after which the plant is not dead, the model satisfies $\neg\langle\langle\emptyset\rangle\rangle d$. Then, the model also satisfies $C_{\alpha|_1} d$, which implies $\neg C_{\alpha|_1} \neg d$. This means that the model satisfies $B_{\alpha|_1} d$. In other words, agent 1 is blamed for d .

4.1.2.3 Causal models [6, 2, 1]

In this section, we consider the approach to formalising responsibility proposed by Chockler and Halpern [2]. We first briefly review Halpern’s definition of causality [6] and Chockler and Halpern’s definition of responsibility and blame [2]. Much of the description below is taken from [6]. The Halpern and Pearl approach (hereafter HP) assumes that the world is described in terms of variables and their values. Some variables may have a causal influence on others. This influence is modelled by a set of *modifiable structural equations*. Variables are split into two sets: the *exogenous* variables, whose values are determined by factors outside the model, and the *endogenous* variables, whose values are ultimately determined by the exogenous variables. The structural equations describe how the outcome is determined.

Formally, a *causal model* M is a pair $(\mathcal{S}, \mathcal{F})$, where \mathcal{S} is a *signature* and \mathcal{F} is a function that associates a structural equation with each variable. A signature \mathcal{S} is a tuple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$,

where \mathcal{U} is a set of exogenous variables, \mathcal{V} is a set of endogenous variables, and \mathcal{R} associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y (i.e., the set of values over which Y ranges). \mathcal{F} associates with each endogenous variable $X \in \mathcal{V}$ a function denoted F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} - \{X\}} \mathcal{R}(Y)) \rightarrow \mathcal{R}(X)$. Thus, F_X defines a structural equation that determines the value of X given the values of other variables. Setting the value of some variable X to x in a causal model $M = (\mathcal{S}, \mathcal{F})$ results in a new causal model, denoted $M_{X \leftarrow x}$, which is identical to M , except that the equation for X in \mathcal{F} is replaced by $X = x$.

Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a *primitive event* is a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$. A *causal formula* (over \mathcal{S}) is one of the form $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$, where φ is a Boolean combination of primitive events. Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}] \varphi$. The special case where $k = 0$ is abbreviated as φ . Intuitively, $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$ says that φ would hold if Y_i were set to y_i , for $i = 1, \dots, k$.

Following [6, 8], we only consider *acyclic* models. In acyclic models, there is a total ordering \prec of the endogenous variables such that if $X \prec Y$, then X is independent of Y , that is, $F_X(\vec{z}, y, \vec{v}) = F_X(\vec{z}, y', \vec{v})$ for all $y, y' \in \mathcal{R}(Y)$. If $X \prec Y$, then the value of X may affect the value of Y , but the value of Y cannot affect the value of X . If M is an acyclic causal model, then given a *context*, that is, a setting \vec{u} for the exogenous variables in \mathcal{U} , there is a unique solution for all the equations: it is possible to solve the equations for the variables in the order given by \prec . A causal formula ψ is true or false in a causal model, given a context. We write $(M, \vec{u}) \models \psi$ if the causal formula ψ is true in causal model M given context \vec{u} . The \models relation is defined inductively. $(M, \vec{u}) \models X = x$ if the variable X has value x in the unique (since we are dealing with acyclic models) solution to the equations in M in context \vec{u} . The truth of conjunctions and negations is defined in the standard way. Finally, $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}] \varphi$ if $(M_{\vec{Y}=\vec{y}}, \vec{u}) \models \varphi$. Thus, $[\vec{Y} \leftarrow \vec{y}] \varphi$ is true in (M, \vec{u}) if φ is true in the model that results after setting the variables in \vec{Y} to \vec{y} .

The causal model M_1 for Example 1 is as follows (note that we introduce the variable for a normative fact of an obligation in addition to the plain facts):

- $\mathcal{U}_1 = \{A_1\}$ is the set of exogenous variables; A_1 corresponds to Agent 1's intention of watering the plant;
- $\mathcal{V}_1 = \{ObF, D, W\}$ is the set of endogenous variables; ObF stands for obligation fulfilled, D for the plant is dead, W for the agent waters the plant
- \mathcal{R} is given by the following structural equations:
 - $W = A_1$;
 - $ObF = W$;
 - $D = \neg W$;

The context is $\{\neg A_1\}$.

Next we define causality. Causality is relative to a model and a context. Only conjunctions of primitive events, abbreviated as $\vec{X} = \vec{x}$, can be causes. What can be caused are arbitrary Boolean combinations of primitive events. Roughly speaking, $\vec{X} = \vec{x}$ is a cause of φ if, had $\vec{X} = \vec{x}$ not been the case, φ would not have happened. To deal with many well-known examples (see [6]), the actual definition is more complicated.

► **Definition 2.** $\vec{X} = \vec{x}$ is an *actual cause* of φ in (M, \vec{u}) if the following three conditions hold:

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x})$ and $(M, \vec{u}) \models \varphi$.

AC2^m. There is a set \vec{W} of variables in \mathcal{V} and settings \vec{x}' of the variables in \vec{X} and \vec{w} of the variables in \vec{W} such that $(M, \vec{u}) \models \vec{W} = \vec{w}$ and

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}] \neg \varphi.$$

AC3. \vec{X} is minimal; no subset of \vec{X} satisfies conditions AC1 and AC2^m.

AC1 states that for $\vec{X} = \vec{x}$ to be a cause of φ , both $\vec{X} = \vec{x}$ and φ have to be true. AC3 is a minimality condition, which ensures that only the conjuncts of $\vec{X} = \vec{x}$ that are essential are parts of a cause. AC2^m (the “m” is for modified; the notation is taken from [6]) captures the counterfactual. It says that if we change the value of \vec{X} from \vec{x} to \vec{x}' , while possibly holding the values of the variables in some (possibly empty) set \vec{W} fixed at their values in the current context, then φ becomes false. We say that (\vec{W}, \vec{x}') is a *witness* to $\vec{X} = \vec{x}$ being a cause of φ in (M, \vec{u}) . If $\vec{X} = \vec{x}$ is a cause of φ in (M, \vec{u}) and $X = x$ is a conjunct of $\vec{X} = \vec{x}$, then $X = x$ is *part of a cause* of φ in (M, \vec{u}) .

In Example 1, the cause of $\neg ObF$ is $\neg W$, and the cause of D is also $\neg W$. The witness is empty in both cases.

The notion of *degree of responsibility* was introduced by Chockler and Halpern in [2]. Roughly speaking, the degree of responsibility $X = x$ for φ measures the minimal number of changes and number of variables that have to be held fixed in order to make φ counterfactually depend on $X = x$. We use the formal definition in [7], which is appropriate for the modified definition of causality used here.

► **Definition 3.** The *degree of responsibility* of $X = x$ for φ in (M, \vec{u}) , denoted $dr((M, \vec{u}), (X = x), \varphi)$, is 0 if $X = x$ is not part of a cause of φ in (M, \vec{u}) ; it is $1/k$ if there exists a cause $\vec{X} = \vec{x}$ of φ and a witness (\vec{W}, \vec{x}') to $\vec{X} = \vec{x}$ being a cause of φ in (M, \vec{u}) such that (a) $X = x$ is a conjunct of $\vec{X} = \vec{x}$, (b) $|\vec{W}| + |\vec{X}| = k$, and (c) k is minimal, in that there is no cause $\vec{X}_1 = \vec{x}_1$ for φ in (M, \vec{u}) and witness (\vec{W}', \vec{x}'_1) to $\vec{X}_1 = \vec{x}_1$ being a cause of φ in (M, \vec{u}) that includes $X = x$ as a conjunct with $|\vec{W}'| + |\vec{X}_1| < k$.

In Example 1, the degree of responsibility of $\neg W$ (essentially, agent 1’s (in) action), for both $\neg ObF$ and D is 1: $dr((M_1, \{\neg A_1\}), (\neg W), \neg ObF) = 1$.

This definition of responsibility assumes that everything relevant about the facts of the world and how the world works is known. In general, there may be uncertainty about both. The notion of *blame* takes this into account. We model an agent’s uncertainty by a pair (\mathcal{K}, Pr) , where \mathcal{K} is a set of causal settings, that is, pairs of the form (M, \vec{u}) , and Pr is a probability distribution over \mathcal{K} . We call such a pair an *epistemic state*. Note that once we have such a distribution, we can talk about the probability that $\vec{X} = \vec{x}$ is a cause of φ relative to (\mathcal{K}, Pr) : it is just the probability of the set of pairs (M, \vec{u}) such that $\vec{X} = \vec{x}$ is a cause of φ in (M, \vec{u}) . We also define the *degree of blame* of $X = x$ for φ to be the expected degree of responsibility:

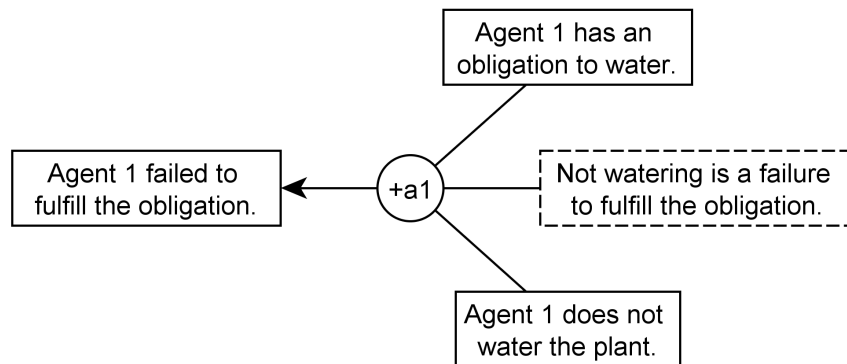
► **Definition 4.** The *degree of blame* of $X = x$ for φ relative to the epistemic state (\mathcal{K}, Pr) is

$$\sum_{(M, \vec{u}) \in \mathcal{K}} dr((M, \vec{u}), X = x, \varphi) \text{Pr}((M, \vec{u})).$$

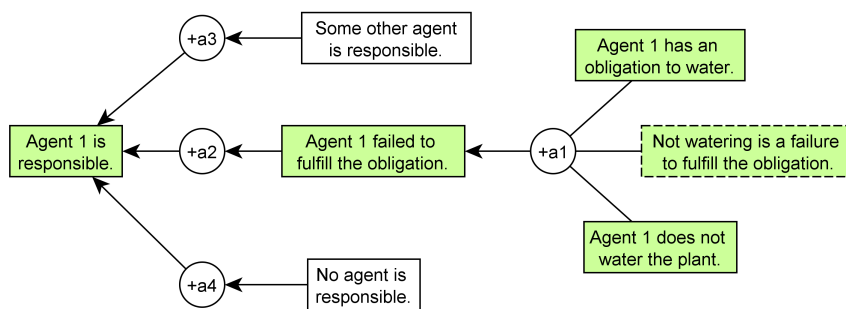
In Example 1, the degree of blame of $\neg W$ may be quite low if $\text{Pr}((M_1, \{\neg A_1\}))$ is low; for example, the agent may not know the structural equation for ObF and assign probability 0 to M_1 .

4.1.2.4 Argumentation theory

The Carneades Argumentation System, named after the Greek skeptical philosopher, is open source software, available at <http://carneades.github.io/>. It is a computational system, because the model consists of mathematical structure whose operations are all computable.



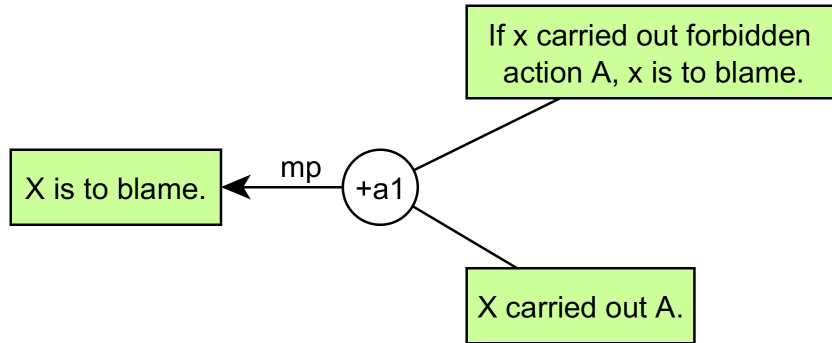
■ **Figure 1** Example 1 propositions and arguments.



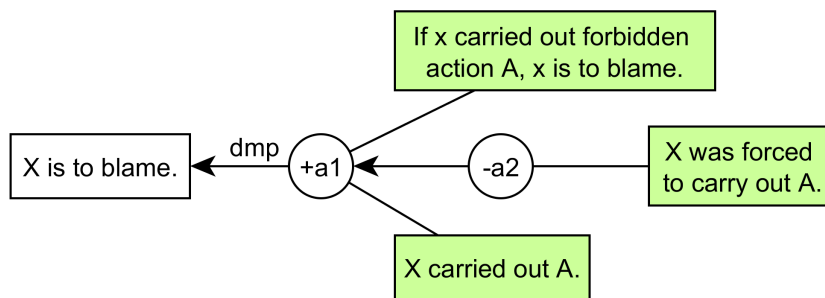
■ **Figure 2** Example 1 with responsibility accepted.

It is also a formal system. Carneades formalises argument graphs, as bipartite, directed graphs, consisting of argument nodes linked to statement nodes. Argument graphs model inferential relationships among arguments and statements. An argument graph is a bipartite, directed, labeled graph, consisting of statement nodes and argument nodes connected by premise and conclusion edges. Formally, an argument graph is a 4-tuple $\langle S, A, P, C \rangle$, where S is a set of statement nodes, A is a set of argument nodes, P is a set of premises, and C is a set of conclusions. To see examples, look ahead to Figures 1-5.

The argument diagrams shown below are graph structures drawn in the style of the Carneades Argumentation System, see, for example, [15]. The sentences in the rectangular nodes denote propositions. The circular nodes represent arguments, which can be pro or con a proposition, or an argument. The propositions are premises or conclusions of arguments. Several premises can support the conclusion together in what is called the linked argument configuration. Or two or more propositions can independently support a conclusion in what is called convergent argumentation structure in informal logic. Implicit premises or conclusions are indicated by the dashed perimeter of the rectangular node. The general idea is that arguments have a graph structure, and in the most typical instances there is an ultimate conclusion to be proved or disproved that is represented as a root of an argumentation tree. By this means the sequence of argumentation on both sides of a disputed issue can be visually represented, and in the end the pro-arguments can be weighed against con arguments so that it can be charged which side had the stronger argument using standards and burdens of proof. An example is shown in Figure 1.



■ **Figure 3** Deductive Modus Ponens Argument.



■ **Figure 4** Pollock-Style Undercutter.

When a rectangular node has a green background, it means that this proposition has been accepted by the audience (in many instances, the user). Based on the user's input, Carneades can use argumentation schemes to calculate whether a conclusion is accepted (labelled 'in') based on a set of premises. An example is shown in Figure 2.

One might initially tend to think that the problem of assigning blame to an agent in a normative multiagent system can simply be dealt with by applying the following rule (R1): if agent *x* carries out action *A*, and action *A* is forbidden in the normative system, then *x* is to blame for carrying out *A*. And in general R1 might work as a base principle for assigning blame in a normative system, but there are two problems with applying it to real cases.

The first problem, the defeasibility of this principle, was extensively discussed long ago by [9], and his contemporaries. Suppose, for example, that *x* did carry out action *A*, but this action was not voluntary, because it fell under the category of one of a list of defeating conditions. For example, suppose *x* was forced to carry out action *A* by another agent *y*. In such an instance it may be true that agent *x* carried out action *A*, and that action *A* is forbidden in the normative system, but it might not be true that *x* is to blame for carrying out action *A*. As Hart pointed out, there might be a long, even open-ended list of such defeating conditions.

How this problem is modeled in formal argumentation systems can be seen by considering the kind of structure pictured in Figure 3. The plus sign represents a pro argument, meaning that the premises are put forward to support acceptance of the conclusion.

The argument in this instance is based on the rule of *modus ponens* as standardly defined in classical deductive logic. If both premises are true, it follows deductively that the conclusion

has to be true. In Figure 3 both premises are colored in green, indicating that both have been accepted by the audience. This means that in a formal argumentation system, such as Carneades, the system will automatically color the conclusion in green.

However, let's go on to consider what happens if we model the inference not by using the deductive version of *modus ponens*, but by defeasible *modus ponens*. When a defeasible rule of inference is used, the acceptance of both premises shifts a weight of presumption towards acceptance of the conclusion, but does not require that the conclusion has to be true. To see how this kind of inference rule works, we need to consider some different ways of attacking and defeating an argument characteristic of argumentation theory.

It is widely recognized in formal argumentation systems of the kind studied in artificial intelligence that there are three ways of attacking an argument: you can attack one or more of the premises, you can attack the conclusion, or you can attack the inferential link joining the premises to the conclusion [11]. The third way is associated with the form of argument attack called a Pollock-style undercutter, which can be illustrated by Pollock's [10] classic example. Suppose I am looking at a light, and it looks red to me, but I also know that it is illuminated by a red light and that red lights can make an object look red even when they are not. Note that the new evidence does not rebut the claim that the object is red, because it might be red for all I know. It merely undercuts the original argument, meaning that it casts the original argument into doubt by undermining the rule that anything that looks red has to be red [14].

This same kind of reasoning can be applied to reasoning from a forbidden action to blame.

By looking at Figure 4, we can see how the defeasible argumentation applies to drawing a conclusion that an agent is to blame for a particular action based on the premises that this action was forbidden and that the agent carried out. Both premises are accepted, and hence in Figure 4 are shown in green, just as they were in Figure 3. But in Figure 4 the inference to the conclusion is based on defeasible *modus ponens* (dmp), which leaves the inference to the conclusion open to being undercut by new information that has come into a particular case [14]. In this instance, the new information is that the agent was forced to carry out the action in question. This finding acts as a con argument, shown as argument a2 in Figure 4, where the minus sign indicates a con argument, an argument that has been put forward to attack a prior argument.

The problem posed by these considerations can be addressed by an argumentation system which allows some arguments to attack other arguments, and in particular which allows for the use of defeasible forms of argument such as defeasible *modus ponens*. However, portraying an inference from premises about an agent's action, and whether these actions are forbidden, to a conclusion that the agent was to blame as a deductive form of argument, does not take defeasibility into account. This is a severe limitation in studying how ethical and legal reasoning are accounted for when studying how to reason properly about responsibility.

The second problem is that the action A that x carried out might have set a chain of consequences into motion, and one of these consequences might constitute an outcome that is forbidden to bring about in the normative system. In such a case, x might correctly have been seen to be properly blamed for carrying out action A, even though A in itself was not forbidden in the normative system. This takes us to the task of modeling the indirect consequences of an agent's actions through causal sequences in order to show how to reason properly about responsibility in multiagent systems.

4.1.3 Distinguishing between causality and responsibility

In this section, we consider a more complex case, in which an agent is obliged to perform an action, and more than one agent may act. In this setting, responsibility for a state of affairs, and responsibility for violation of the norm do not necessarily coincide.

► **Example 5.** As before, a plant must be watered in order to prevent it dying. Agent 1 has an obligation to water the plant. Agent 1 does not water the plant. Agent 2 could have watered the plant (is able to see that the plant is not watered, and is able to water it) but didn't. The plant dies. Who is responsible for the death of the plant? Who is responsible for the violation of the obligation?

4.1.3.1 Modal logic with three modalities

The existence of Agent 2 does not change the analysis for this example. Agent 1 is still responsible for violating the obligation and the dead plant. Agent 2 did not have an obligation to water the plant, and hence is not responsible.

4.1.3.2 Logic of strategic ability

Similarly as before, we can model Example 5 with $\mathbb{P} = \{d\}$, $\mathbb{N} = \{1, 2\}$ and $\mathbb{A} = \{nop, water\}$, and also:

$$\begin{aligned}\alpha &= \{(1, nop), (2, nop)\} \\ \beta &= \{(1, nop), (2, water)\} \\ \gamma &= \{(1, water), (2, nop)\} \\ \delta &= \{(1, water), (2, water)\}\end{aligned}$$

Assume a model satisfying:

$$\begin{aligned}&K_N(\neg[\alpha] \perp \wedge [\alpha]d) \\ &K_N(\neg[\beta] \perp \wedge [\beta]\neg d) \\ &K_N(\neg[\gamma] \perp \wedge [\gamma]\neg d) \\ &K_N(\neg[\delta] \perp \wedge [\delta]\neg d) \\ &R_1\neg d \\ &\neg R_2\neg d\end{aligned}$$

We have that the model satisfies $\neg C_{\alpha|_1}\neg d$. (Also note that $\alpha|_1$ is the same as $\beta|_1$.) This means that it also satisfies $B_{\alpha|_1}d$. In other words, agent 1 is blamed for the death of the plant. However, since agent 2 did not have forward-looking responsibility for the plant, 2 is not blamed for the undesirable outcome.

4.1.3.3 Causal models

The model M_2 for Example 5 is as follows:

- $\mathcal{U}_2 = \{A_1, A_2\}$ is the set of exogenous variables; A_i corresponds to Agent i 's intention of watering the plant;
- $\mathcal{V}_2 = \{ObF, D, W_1, W_2\}$ is the set of endogenous variables; ObF stands for obligation fulfilled, D for the plant is dead, W_i for agent i waters the plant;
- \mathcal{R} is given by the following structural equations:
 - $W_1 = A_1$;

- $W_2 = A_2$;
- $ObF = W_1$;
- $D = \neg W_1 \wedge \neg W_2$;

The context is $\{\neg A_1, \neg A_2\}$.

Now the cause of $\neg ObF$ is still $\neg W_1$, but the cause of D is $\neg W_1, \neg W_2$. That is, as in the Strategic Ability model, we can distinguish between the cause of the plant's death, and responsibility for the failure to fulfil the obligation.

4.1.3.4 Argumentation theory

In this section, we consider how argumentation theory can be used to give an explanation of some aspects of the reasoning in Example 5. We begin by changing the question asked in the example according to the following formulation, with the aim of trying to understand what general ethical principle could be applied to the specific circumstances of the case.

In Example 5, Agent 1 has an obligation to water the plant. Agent 1 did not water the plant. But agent 1 knew that if he did not water the plant the plant will die. The plant dies. Agent 2 has no obligation to water the plant. But agent 2 also knew that if she did not water the plant the plant will die. Who is to blame for the death of the plant?

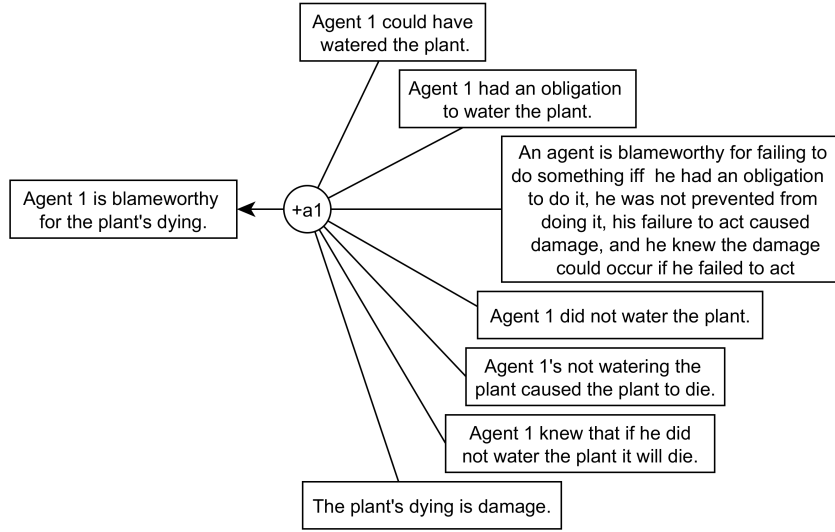
From the text of this example, the argumentation expressed in it can be represented as an instance of case-based reasoning based on an implicit ethical generalization stating a set of conditions that are necessary and sufficient for drawing the conclusion that agent 1 is blameworthy for the plant's dying. The generalization is the statement that an agent is blameworthy for failing to do something if and only if the agent had an obligation to do it, he was not prevented from doing it, his failure to act caused damage, and he knew the damage could occur if he failed to act. Each one of the four conditions is taken to be necessary in the generalization to support the inference to the conclusion that agent one is blameworthy for the plant's dying, and the conjunction of the four conditions is taken to be sufficient to support the conclusion that agent one is blameworthy for the plant's dying. The argumentation structure representing this reasoning is shown in Figure 5.

What this means in the Carneades Argumentation System is that if all seven premises of argument a1 are accepted, the conclusion of argument a1 should be accepted as following from them.

But what about the case of agent 2? The argument diagram for the case of agent 2 is the same as the case of agent 1, as shown in Figure 5, except that the second premise from the top, containing the proposition that agent 1 had an obligation to water the plant, does not hold. This means that even although the other six premises do hold, and are therefore colored green in the diagram, the conclusion now fails to hold. So the conclusion is colored with a white background, showing that it is no longer accepted, based on the argument.

One conclusion that can be drawn from this way of structuring the argumentation in the case is that the generalization shown in the large rectangular node in figure 5, once combined in an argument structure with the other premises specifying the factual circumstances taken to hold in the case, provides sufficient support for us to draw the conclusion that agent 1 is blameworthy. Another conclusion that can be drawn is that once the premise that agent 2 had an obligation to water the plant is no longer accepted as holding in this variant of the case, the conclusion that agent 2 is blameworthy is no longer supported as acceptable.

The question now raised is whether the ethical generalization used to draw the conclusions about blameworthiness based on the differing circumstances of the two agents is the correct basis for drawing this conclusion. In other words, is this generalization the correct ethical principle that should generally be used for deciding whether or not an agent is blameworthy



■ **Figure 5** Argument for the Responsibility of Agent 1 in the Plant Example.

when an agent has carried out actions fitting the requirements of the circumstances specified in the two examples? So far, it can stand as a hypothesis that this ethical principle can provide a provisional basis for drawing conclusions of this sort in specific cases. It can be put in place as a starting point for investigating further more complex cases where the circumstances are varied to fit problematic cases of assigning blame and responsibility. If or when counter-examples are found in the new cases, the generalization may have to be modified or even given up.

4.1.4 Unintentional violation

► **Example 6.** Agent 1 has an obligation to water the plant. Agent 1 does not water the plant because it is raining and agent 1 assumes the plant will get watered by the rain. However the plant is under cover and does not get watered by the rain. Agent 2 could have watered the plant (is able to see that the plant is not watered, and is able to water it) but didn't. The plant dies. Who is responsible for the death of the plant? Who is responsible for the violation of the obligation?

4.1.4.1 Modal logic with three modalities

This is the case of error/negligence on the part of agent 1: $\neg K_1(w_1 \rightarrow \neg d)$ (or, $\neg K_1(\neg w_1 \rightarrow d)$).

4.1.4.2 Logic of strategic ability

We cannot model this example correctly in CEDL. The reason is that, in the example, the agent “believes” (instead of “knows”) that the rain will water the plant. The notion of belief is different than that of knowledge. If an agent believes φ then φ is true in all situations that are considered possible by the agent but (as in Example 6) the agent may be wrong. In terms of Kripke semantics, this means that the actual situation may not be one of the

situations that the agent considers possible. Even more technically, this means that the axiom T ($K\varphi \rightarrow \varphi$) is not valid in a logic modelling beliefs. However, it is valid for knowledge, and thus, valid in CEDL.

This is why CEDL, as it stands, cannot model the problem. To better understand it, let the formula $K_i[\alpha]\neg d$ stand for ‘the agent knows that, after the rain (α), the plant is not dead’. By axiom T, we must have $[\alpha]\neg d$, which means ‘after the rain, the plant is not dead’. The latter cannot be the case in Example 6.

To avoid the latter problem, one may propose to just replace axiom T by axiom D and thus work with a logic where operator K means belief instead of knowledge. The operator K in this case would be the common operator for beliefs, for example, studied in [5]. But there is another axiom in this logic that may cause problems. The so-called ‘no-forgetting’ principle, which is as follows:

$$K_G[\alpha|_G]\varphi \rightarrow [\alpha|_G]K_G\varphi$$

This principle implies that the knowledge (or in this case the beliefs) of agents either increase or stay the same after the execution of any action. The presence of ‘no-forgetting’ prevents situations where agents come to know (or believe) something that contradicts what was known (or believed) before. If we get back to our example, we have that, at first, the agent thinks that the plant will be alive after the rain, but once the plant dies, the agent still thinks that it is alive. For instance, let the formula $K_i[\alpha|_N]\neg d$ stand for ‘the agent believes that, after the rain (α) the plant is not dead’. By ‘no-forgetting’, we must have $[\alpha|_N]K_i \neg d$, which means ‘after the rain, the agent believes that the plant is not dead’. The result is a logic where, if the agent believes something that is not correct, the agent will keep believing it, no matter what.

We may, nonetheless, try to model Example 6 in this new formalism. First, we have to add a third agent that represents the environment. (This can be seen as “the rain” in the example.) The set of agents is thus $N = \{1, 2, 3\}$. Every agent, including the environment agent 3, may water the plant or not. Hence, the sets of actions for each agent are $A_i = \{water, skip\}$, for $i \in N$. The joint actions are thus all the combinations of these two actions: $\{(1, skip), (2, skip), (3, skip)\}, \{(1, skip), (2, skip), (3, water)\}, \dots$. Now, since agent 1 thinks that the rain will water the plant, we could try to assume a model satisfying the following formula:

$$\begin{aligned} &K_1[(3, skip)]\perp \\ &K_1(\neg[(3, water)]\perp \wedge [(3, water)]\neg d) \end{aligned}$$

These formulas mean that agent 1 believes that agent 3 cannot skip and hence agent 3 necessarily waters the plant. However, by ‘no-forgetting’, we must have:

$$[(3, skip)]K_1\perp$$

The latter is inconsistent with axiom D. To try to find a way around this problem, we may consider that the agent believes that action skip also waters the plant. In this case we have, instead, a model satisfying:

$$\begin{aligned} &K_1([(3, skip)]\neg\perp \wedge [(3, skip)]\neg d) \\ &[(1, skip), (2, skip), (3, skip)]\neg\perp \wedge [(1, skip), (2, skip), (3, skip)]d \end{aligned}$$

These two formulas mean that agent 1 believes that, after agent 3 skips, the plant is alive, but actually this is not the case. For the other actions, we have the usual. Thus assume that

the model also satisfies:

$$\begin{aligned}
& K_N(\neg[(1, skip), (2, skip), (3, water)] \perp \wedge [(1, skip), (2, skip), (3, water)] \neg d) \\
& K_N(\neg[(1, skip), (2, water), (3, skip)] \perp \wedge [(1, skip), (2, water), (3, skip)] \neg d) \\
& \vdots \\
& R_1 \neg d \\
& \neg R_2 \neg d \\
& \neg R_3 \neg d
\end{aligned}$$

Because there is an action after which the plant is dead, the model satisfies $\neg\langle\langle\emptyset\rangle\rangle\neg d$. We also have that agent 1 believes that skipping ensures that the plant will be alive: $K_1 E_{(1, skip)} \neg d$. Then, by the definitions given, the agent “believably” causes that the plant is alive after skipping: $C_{\alpha_1} \neg d$ (even though it may actually be dead). This means that the model satisfies $\neg B_{(1, skip)} d$. In other words, agent 1 is not blamed for the eventual death of the plant. The reason that agent 1 is excused is that the agent believes that the death of the plant is prevented. One may of course wonder whether this is enough to excuse the agent.

4.1.4.3 Causal models

Responsibility stays the same (agents are both causally responsible). However under the reasonable probability distribution over possible models (where the chance of the plant not being watered by the rain when it is raining is very small) the degree of blame attached to agent 1 is small.

4.1.4.4 Argumentation theory

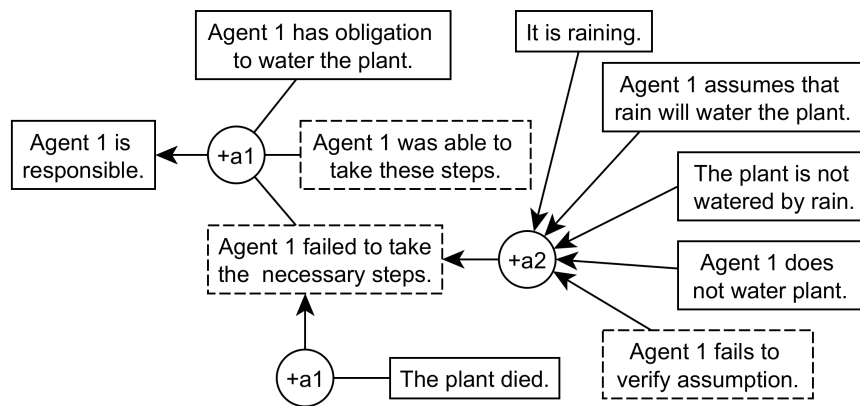
In example 6 two agents are involved. One of them wrongly assumes that the plant will get watered by rain, but this does not turn out to be true. The other agent could have also watered the plant but didn’t, and so the plant dies. The arguments in this case are composed from seven propositions stated in the key list below.

Key List for Responsibility of Agent 1:

- (1) Agent 1 has an obligation to water the plant.
- (2) Agent 1 does not water the plant.
- (3) It is raining.
- (4) Agent 1 assumes that the plant will get watered by the rain.
- (5) The plant does not get watered by the rain.
- (6) The plant died.
- (7) Agent 1 is responsible for the death of the plant.

It is known that the plant did not get watered by the rain because it was under cover. This is an explanation of why the plant did not get watered (as opposed to an argument), so it was not included in the argument diagram in Figure 6. However, three implicit premises need to be inserted in order for us to make sense of the argumentation in the example. The following three implicit premises are shown in rectangles with broken-line perimeters.

- (8) Agent 1 failed to verify his assumption that the plant will get watered by the rain.
- (9) Agent 1 failed to take the necessary steps to see to it that the plant was watered.
- (10) Agent 1 was able to take these necessary steps.



■ **Figure 6** Argument Diagram for Agent 1 in Example 6.

Here is the general ethical principle behind drawing inferences about responsibility and blame in cases concerning unintentional violations of an obligation. An agent can be held responsible for failing to fulfill an obligation even if he thought it would be fulfilled in the normal course of events without his taking any further steps to see that this happens. This can occur where the agent had some reason to assume that in the circumstances he does not need to intervene to see to it that the obligation is fulfilled. If it was not, he can be held responsible for failing to fulfill his obligation on the grounds that he failed to take steps that he could have and should have taken to see to it that the bad outcome he was obliged to prevent from occurring did not occur. In law this sort of principle applies to judging cases of responsibility relating to failures such as ‘taking due care’ or taking precautions. Next we need to consider the responsibility of agent 2. Here we have a key list of five propositions and we need to add one implicit premise.

Key List for Responsibility of Agent 2

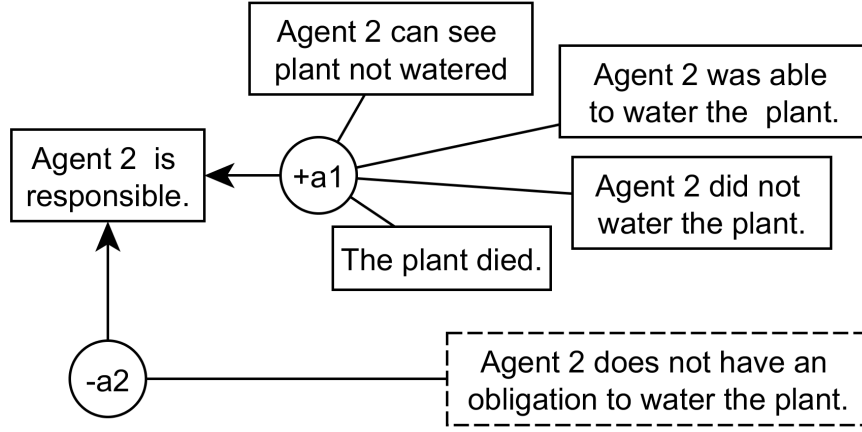
- (1) Agent 2 is able to see that the plant is not watered.
- (2) Agent 2 is able to water the plant.
- (3) Agent 2 did not water the plant.
- (4) The plant died.
- (5) Agent 2 is responsible for the death of the plant.

Implicit Premise

- (6) Agent 2 does not have an obligation to water the plant.

Based on this interpretation of the argument in Example 6 concerning the responsibility of agent 2, the argument diagram shown in Figure 8 shows that there is a pro argument +a1 supporting the conclusion that agent 2 is responsible, but there is also a con argument –a2 attacking the conclusion that agent 2 is responsible.

The con argument a2 shows that the pro argument is not strong enough to prove its conclusion because, as shown in the ethical principle formulated in connection with Example 5 (see Figure 5), having an obligation to carry out action is a necessary requirement to draw the conclusion that an agent can be held responsible for failure to carry out the action. Hence in this instance, the con argument defeats the pro argument.



■ **Figure 7** Argument Diagram for Agent 2 in Example 6.

4.1.5 Causality revisited (and knowledge of strategy)

► **Example 7.** Agents 1 and 2 have an obligation that exactly one of them waters the plant (if both of them do, the plant also dies). None of them waters the plant. The plant dies. Who is responsible for the death of the plant? Who is responsible for the violation of the obligation?

4.1.5.1 Modal logic with three modalities

Suppose both agents have an obligation of ‘not too much watering’ since otherwise the plant dies. Suppose that both agents watered too much and the plant died. Using *conditio sine qua non*, the conclusion is derived:

- even if Agent 1 had not watered too much, the plant would have died anyway so Agent 1 is not responsible and
- even if Agent 2 had not watered too much, the plant would have died anyway so Agent 2 is not responsible.

For analysis, see [13].

4.1.5.2 Logic of strategic ability

This is similar to a previous example. The only difference on the modelisation is that exactly one of the agents is responsible: Let the group of agents be $G = \{1, 2\}$, and assume a model satisfying:

$$R_G \neg d \\ (R_1 \neg d \vee R_2 \neg d) \wedge (\neg R_1 \neg d \vee \neg R_2 \neg d)$$

As before, we have that the model satisfies $C_{\alpha|G} d$. This means that $B_{\alpha|G} d$ is satisfied and thus, group G is blamed for the death of the plant.

The difference here is that exactly one of the agents is individually blamed for the death of the plant. That is, we have:

$$(B_{\alpha|1} d \vee B_{\alpha|2} d) \wedge (\neg B_{\alpha|1} d \vee \neg B_{\alpha|2} d)$$

4.1.5.3 Causal models

Each of the agents is individually responsible (if everything else stayed the same and agent 1 watered the plant, the plant would have been alive; similarly for agent 2). The degree of blame is not 1 however since there is a non-zero probability that agent 1 watering the plant (in the context where agent 2 also waters the plant) would have caused it to die.

4.1.6 Group responsibility

► **Example 8.** Agents 1 and 2 have an obligation to water the plant. Neither of them does. The plant dies. Who is responsible for the death of the plant? Who is responsible for the violation of the obligation?

In this case we consider only logic of strategic ability and causal models.

4.1.6.1 Logic of strategic ability

This example can be modeled as follows. Similarly as before we have:

$$\begin{aligned}\alpha &= \{(1, \text{nop}), (2, \text{nop})\} \\ \beta &= \{(1, \text{nop}), (2, \text{water})\} \\ \gamma &= \{(1, \text{water}), (2, \text{nop})\} \\ \delta &= \{(1, \text{water}), (2, \text{water})\}\end{aligned}$$

Let the group of agents be $G = \{1, 2\}$, and assume a model satisfying:

$$\begin{aligned}&K_N(\neg[\alpha] \perp \wedge [\alpha]d) \\ &K_N(\neg[\beta] \perp \wedge [\beta] \neg d) \\ &K_N(\neg[\gamma] \perp \wedge [\gamma] \neg d) \\ &K_N(\neg[\delta] \perp \wedge [\delta] \neg d) \\ &R_G \neg d\end{aligned}$$

We have that the model satisfies $C_{\alpha|G}d$. This means that it also satisfies $B_{\alpha|G}d$. In other words, group G is blamed for the death of the plant.

Note that no agent is individually held forward-looking responsible for the plant is not dead. This is why agents 1 and 2 are not blamed individually. However, if one of them, e.g. 1, is held individually responsible, i.e. $R_1 \neg d$, then it would be held responsible, exactly as in the previous example. The same for agent 2.

4.1.6.2 Causal models

The cause of the plant dying is that neither agent watered the plant; so both agents' actions are part of a single cause. The degree of responsibility is therefore 1/2 for each agent. The degree of blame depends on the probability distribution; it is reasonable to assume that it is the same as the degree of responsibility (that is, the given context has probability 1).

References

- 1 Natasha Alechina, Joseph Y. Halpern, and Brian Logan. Causality, responsibility and blame in team plans. In S. Das, E. Durfee, K. Larson, and M. Winikoff, editors, *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, Sao Paulo, Brazil, May 2017. IFAAMAS, IFAAMAS.

- 2 H. Chockler and J. Y. Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 20:93–115, 2004.
- 3 Tiago de Lima and Lambèr Royakkers. A formalisation of moral responsibility and the problem of many hands. In Ibo van de Poel, Lambèr Royakkers, and Sjoerd D. Swart, editors, *Moral Responsibility and the Problem of Many Hands*, Routledge Studies in Ethics and Moral Theory, chapter 3, pages 93–130. Taylor & Francis, 2015.
- 4 Tiago de Lima, Lambèr Royakkers, and Frank Dignum. Modeling the problem of many hands in organisations. In H. Coelho, Rudi Studer, and Michael Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 79–84. IOS Press, 2010.
- 5 Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
- 6 J. Y. Halpern. A modification of the Halpern-Pearl definition of causality. In *Proc. 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 3022–3033, 2015.
- 7 J. Y. Halpern. *Actual Causality*. MIT Press, Cambridge, MA, 2016.
- 8 J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *British Journal for Philosophy of Science*, 56(4):843–887, 2005.
- 9 H. L. A. Hart. The ascription of responsibility and rights. In Gilbert Ryle and Antony Flew, editors, *Proceedings of the Aristotelian Society*, pages 171–194. Blackwell, 1951.
- 10 J.L. Pollock. *Cognitive Carpentry*. The MIT Press, 1995.
- 11 H. Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.
- 12 Ken Satoh. Private communication, 2018.
- 13 Ken Satoh and Satoshi Tojo. Disjunction of causes and disjunctive cause: a solution to the paradox of *conditio sine qua non* using minimal abduction. In Tom M. van Engers, editor, *Legal Knowledge and Information Systems - JURIX 2006: The Nineteenth Annual Conference on Legal Knowledge and Information Systems, Paris, France, 7-9 December 2006*, volume 152 of *Frontiers in Artificial Intelligence and Applications*, pages 163–168. IOS Press, 2006.
- 14 D. Walton. Evaluating expert opinion evidence. In *Argument Evaluation and Evidence*, volume 23 of *Law, Governance and Technology Series*, pages 117–144. Springer, 2016.
- 15 D. Walton and T. F. Gordon. Formalizing informal logic. *Informal Logic*, 35(4):508–538, 2015.

Participants

- Tobias Ahlbrecht
TU Clausthal, DE
- Natasha Alechina
University of Nottingham, GB
- Kevin D. Ashley
University of Pittsburgh, US
- Matteo Baldoni
University of Turin, IT
- Christoph Benzmüller
FU Berlin, DE
- Célia da Costa Pereira
Laboratoire I3S –
Sophia Antipolis, FR
- Mehdi Dastani
Utrecht University, NL
- Tiago de Lima
CNRS – Lens, FR
- Davide Dell’Anna
Utrecht University, NL
- Jürgen Dix
TU Clausthal, DE
- Ali Farjami
University of Luxembourg, LU
- Dov M. Gabbay
King’s College London, GB
- Aditya K. Ghose
University of Wollongong, AU
- Matthias Grabmair
Carnegie Mellon University –
Pittsburgh, US
- Joris Hulstijn
Tilburg University, NL
- Wojtek Jamroga
Polish Academy of Sciences –
Warsaw, PL
- Özgür Kafali
University of Kent –
Canterbury, GB
- Sabrina Kirrane
Wirtschaftsuniversität Wien, AT
- Brian Logan
University of Nottingham, GB
- Emiliano Lorini
University of Toulouse, FR
- Martin Neumann
Jacobs University Bremen, DE
- Pablo Noriega
IIIA – CSIC – Barcelona, ES
- Julian Padget
University of Bath, GB
- Adrian Paschke
FU Berlin, DE
- Nicolas Payette
LABSS – ISTC – CNR –
Rome, IT
- Ken Satoh
National Institute of Informatics –
Tokyo, JP
- Matthias Scheutz
Tufts University – Medford, US
- Viviane Torres da Silva
IBM Research –
Rio de Janeiro, BR
- Leon van der Torre
University of Luxembourg, LU
- Harko Verhagen
Stockholm University, SE
- Douglas Walton
University of Windsor, CA
- Michael Winikoff
University of Otago, NZ
- Vahid Yazdanpanah
University of Twente, NL



Algebraic Effect Handlers go Mainstream

Edited by

Sivaramakrishnan Krishnamoorthy Chandrasekaran¹, Daan Leijen²,
Matija Pretnar³, and Tom Schrijvers⁴

1 University of Cambridge, GB, sk826@cl.cam.ac.uk

2 Microsoft Research – Redmond, US, daan@microsoft.com

3 University of Ljubljana, SI, matija.pretnar@fmf.uni-lj.si

4 KU Leuven, BE, tom.schrijvers@cs.kuleuven.be

Abstract

Languages like C#, C++, or JavaScript support complex control flow statements like exception handling, iterators (yield), and even asynchrony (async/await) through special extensions. For exceptions, the runtime needs to be extended with exception handling stack frames. For iterators and asynchrony, the situation is more involved, as the compiler needs to turn regular code into stack restoring state machines. Furthermore, these features need to interact as expected, e.g. finally blocks must not be forgotten in the state machines for iterators. And all of this work needs to be done again for the next control flow abstraction that comes along.

Or we can use algebraic effect handlers! This single mechanism generalizes all the control flow abstractions listed above and more, composes freely, has simple operational semantics, and can be efficiently compiled, since there is just one mechanism that needs to be supported well. Handlers allow programmers to keep the code in direct-style, which is easy to reason about, and empower library writers to implement various high-level abstractions without special extensions.

The idea of algebraic effects handlers has already been experimented with in the form of small research languages and libraries in several mainstream languages, including OCaml, Haskell, Clojure, and Scala. The next step, and the aim of this seminar, is to seriously consider adoption by mainstream languages including both functional languages such as OCaml or Haskell, as well as languages like JavaScript and the JVM and .NET ecosystems.

Seminar April 22–27, 2018 – <http://www.dagstuhl.de/18172>

2012 ACM Subject Classification Software and its engineering → Control structures, Software and its engineering → Formal methods, Software and its engineering → Semantics

Keywords and phrases algebraic effect handlers, implementation techniques, programming abstractions, programming languages

Digital Object Identifier 10.4230/DagRep.8.4.104

Edited in cooperation with Jonathan Immanuel Brachthäuser



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Algebraic Effect Handlers go Mainstream, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 104–125

Editors: Sivaramakrishnan Krishnamoorthy Chandrasekaran, Daan Leijen, Matija Pretnar, and Tom Schrijvers



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Executive Summary

Sivaramakrishnan Krishnamoorthy Chandrasekaran (University of Cambridge, GB)

Daan Leijen (Microsoft Research – Redmond, US)

Matija Pretnar (University of Ljubljana, SI)

Tom Schrijvers (KU Leuven, BE)

License © Creative Commons BY 3.0 Unported license

© Sivaramakrishnan Krishnamoorthy Chandrasekaran, Daan Leijen, Matija Pretnar, and Tom Schrijvers

Algebraic effects and their handlers have been steadily gaining attention as a programming language feature for composably expressing user-defined computational effects. Algebraic effect handlers generalise many control-flow abstractions such as exception handling, iterators, `async/await`, or backtracking, and in turn allow them to be expressed as libraries rather than implementing them as primitives as many language implementations do. While several prototype languages that incorporate effect handlers exist, they have not yet been adopted into mainstream languages. This Dagstuhl Seminar 18172 “Algebraic Effect Handlers Go Mainstream” touched upon various topics that hinder adoption into mainstream languages. To this end, the participants in this seminar included a healthy mix of academics who study algebraic effects and handlers, and developers of mainstream languages such as Haskell, OCaml, Scala, WebAssembly, and Hack.

This seminar follows the earlier, wildly successful Dagstuhl Seminar 16112 “From Theory to Practice of Algebraic Effects and Handlers” which was dedicated to addressing fundamental issues in the theory and practice of algebraic effect handlers. We adopted a similar structure for this seminar. We had talks each day in the morning, scheduled a few days ahead. The folks from the industry were invited to present their perspectives on some of the challenges that could potentially be addressed with the help of effect handlers. The afternoons were left free for working in self-organised groups and show-and-tell sessions with results from the previous days. We also had impromptu lectures on the origins of algebraic effects and handlers, which were quite well received and one of the highlights of the seminar.

Between the lectures and working-in-groups, the afternoons were rather full. Hence, a few participants offered after-dinner “cheesy talks” just after the cheese was served in the evening. The participants were treated to entertaining talks over delightful cheese and fine wine. We encourage the organisers to leave part of the day unplanned and go with what the participants feel like doing on that day. The serendipitous success is what makes Dagstuhl Seminars special.

We are delighted with the outcome of the seminar. There were interesting discussions around the problem of *encapsulation* and leaking of effects in certain higher order use cases, with several promising solutions discussed. It was identified that the problem of encapsulation and leaking effect names is analogous to the name binding in lambda calculus. Another group made significant progress in extending WebAssembly with support for effect handlers. The proposal builds on top of support for exceptions in WebAssembly. During the seminar week, the syntax extensions and operational semantics were worked out, with work begun on the reference implementation. During the seminar, Andrej Bauer pointed out that several prototype implementations that incorporate effect handlers exist, each with their own syntax and semantics. This makes it difficult to translate ideas across different research groups. Hence, Andrej proposed and initiated effects and handlers rosetta stone – a repository of examples demonstrating programming with effects and handlers in various programming languages. This repository is hosted on GitHub and has had several contributions during and after the seminar.

In conclusion, the seminar inspired discussions and brought to light the challenges in incorporating effect handlers in mainstream languages. During the previous seminar (16112), the discussions were centered around whether it was even possible to incorporate effect handlers into mainstream languages. During this seminar, the discussions were mainly on the ergonomics of effect handlers in mainstream languages. This is a testament to the success of the Dagstuhl Seminars in fostering cutting edge research.

2 Contents

Executive Summary

Sivaramakrishnan Krishnamoorthy Chandrasekaran, Daan Leijen, Matija Pretnar, and Tom Schrijvers 105

Overview of Talks

Linking Types for Multi-Language Software <i>Amal Ahmed</i>	109
Idealised Algol <i>Robert Atkey</i>	109
What is algebraic about algebraic effects and handlers? <i>Andrej Bauer</i>	110
Event Correlation with Algebraic Effects <i>Oliver Bracevac</i>	110
Effect Handlers for the Masses <i>Jonathan Immanuel Brachthäuser</i>	111
Combining Algebraic Theories <i>Jeremy Gibbons</i>	111
Handlers.Js: A Comparative Study of Implementation Strategies for Effect Handlers on the Web <i>Daniel Hillerström</i>	113
First Class Dynamic Effect Handlers and Deep Finally Handling <i>Daan Leijen</i>	113
Encapsulating effects <i>Sam Lindley</i>	114
Experiences with structuring effectful code in Haskell <i>Andres Löh</i>	119
Make Equations Great Again! <i>Matija Pretnar</i>	119
Quirky handlers <i>Matija Pretnar and Žiga Lukšič</i>	119
What is coalgebraic about algebraic effects and handlers? <i>Matija Pretnar</i>	120
Effect Handlers for WebAssembly (Show and Tell) <i>Andreas Rossberg</i>	120
Neither Web Nor Assembly <i>Andreas Rossberg</i>	121
Efficient Compilation of Algebraic Effects and Handlers <i>Tom Schrijvers</i>	121
Algebraic effects – specification and refinement <i>Wouter Swierstra</i>	122

Adding an effect system to OCaml	
<i>Leo White</i>	122
Multi-Stage Programming with Algebraic Effects	
<i>Jeremy Yallop</i>	122
Working groups	
Denotational Semantics for Dynamically Generated Effects	
<i>Robert Atkey</i>	123
Reasoning with Effects	
<i>Jeremy Gibbons</i>	124
Participants	125

3 Overview of Talks

3.1 Linking Types for Multi-Language Software

Amal Ahmed (Northeastern University – Boston, US)

License © Creative Commons BY 3.0 Unported license
© Amal Ahmed

Main reference Daniel Patterson, Amal Ahmed: “Linking Types for Multi-Language Software: Have Your Cake and Eat It Too”, in Proc. of the 2nd Summit on Advances in Programming Languages, SNAPL 2017, May 7-10, 2017, Asilomar, CA, USA, LIPIcs, Vol. 71, pp. 12:1–12:15, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017.

URL <https://doi.org/10.4230/LIPIcs.SNAPL.2017.12>

In the last few years, my group at Northeastern has focused on verifying compositional compiler correctness for today’s world of multi-language software. Such compilers should allow compiled components to be linked with target components potentially compiled from other, very different, languages. At the same time, compilers should also be fully abstract: that is, they should ensure that if two components are equivalent in all source contexts then their compiled versions are equivalent in all target contexts. Fully abstract compilation allows programmers to reason about their code (e.g., about correctness of refactoring) by only considering interactions with other code from the same language. While this is obviously an extremely valuable property for compilers, it rules out linking with target code that has features or restrictions that can not be represented in the source language that is being compiled.

While traditionally fully abstract compilation and flexible linking have been thought to be at odds, I’ll present a novel idea called Linking Types [1] that allows us to bring them together by enabling a programmer to opt into local violations of full abstraction only when she needs to link with particular code without giving up the property globally. This fine-grained mechanism enables flexible interoperation with low-level features while preserving the high-level reasoning principles that fully abstract compilation offers.

An open question is whether algebraic effects and effect handlers might be an effective way of designing linking-types extensions for existing languages.

References

- 1 Daniel Patterson and Amal Ahmed. *Linking Types for Multi-Language Software: Have Your Cake and Eat It Too*. In Summit on Advances in Programming Languages (SNAPL), 2017.

3.2 Idealised Algol

Robert Atkey (University of Strathclyde – Glasgow, GB)

License © Creative Commons BY 3.0 Unported license
© Robert Atkey

Joint work of Robert Atkey, Michel Steuwer, Sam Lindley, Christophe Dubach

Main reference Robert Atkey, Michel Steuwer, Sam Lindley, Christophe Dubach: “Strategy Preserving Compilation for Parallel Functional Code”, CoRR, Vol. abs/1710.08332, 2017.

URL <https://arxiv.org/abs/1710.08332>

Idealised Algol was introduced by John Reynolds in the late 1970s as the orthogonal combination of λ -calculus and imperative programming. The resulting language is an elegant combination of imperative programming (variables, while loops, etc.) and procedures (provided by λ -abstraction). A variant of Idealised Algol, called Syntactic Control of


Interference, provides a way to banish aliasing within the language, and hence to provide a way to express race free parallelism.

In this talk, I discussed the similarities between Idealised Algol’s expressive handling of variables and mutable state, in particular Reynolds’ conception of variables as “objects” consisting of a getter and a setter, and handlers for algebraic effects. Idealised Algol appears to naturally include a particularly efficient subset of handlers: linear handlers (the continuation must always be used), that are tail recursive. By sticking to this subset, it is possible to generate efficient code without expensive stack manipulation.

Some of this work is joint with Sam Lindley, Michel Steuwer and Christophe Dubach, where we have applied Idealised Algol with interference control to the problem of generating code from functional specifications for execution on parallel computing hardware, such as multicore processors and GPUs.

3.3 What is algebraic about algebraic effects and handlers?

Andrej Bauer (University of Ljubljana, SI)

License  Creative Commons BY 3.0 Unported license
© Andrej Bauer

In this tutorial we reviewed the classic treatment of algebraic theories and their models in the category of sets. We then drew an uninterrupted line of thought from the classical theory to algebraic effects and handlers. First we generalized operations with integral arities to parameterized operations with arbitrary arities, as these are needed for modeling computational effects. The free models of theories with generalized operations can be used as denotations of effectful programs. The universal property of a free models can be used to derive the notion of handlers. At the level of types, the value types correspond to sets of generators and the computation types to the free models. The naive set-theoretic treatment presented in the tutorial should be replaced with a domain-theoretic one if we wanted adequate denotational semantics of a realistic programming language with general recursion. The contents of the tutorial has been written up an extended in [1].

References

- 1 Andrej Bauer. What is algebraic about algebraic effects and handlers? *ArXiv e-print 1807.05923*, 2018. <https://arxiv.org/abs/1807.05923>.

3.4 Event Correlation with Algebraic Effects

Oliver Bracevac (TU Darmstadt, DE)

License  Creative Commons BY 3.0 Unported license
© Oliver Bracevac

Joint work of Oliver Bracevac, Nada Amin, Guido Salvaneschi, Sebastian Erdweg, Patrick Eugster, Mira Mezini
Main reference Oliver Bracevac, Nada Amin, Guido Salvaneschi, Sebastian Erdweg, Patrick Eugster, Mira Mezini, “Versatile Event Correlation with Algebraic Effects”, *Proc. ACM Program. Lang.*, Vol. 2, pp. 67:1–67:31, ACM, 2018.
URL <http://dx.doi.org/10.1145/3236762>

This talk presents a language design on top of algebraic effects and handlers for defining n -way joins over asynchronous event sequences. The design enables mix-and-match-style compositions of join variants from different domains, e.g., stream-relational algebra, event

processing, reactive and concurrent programming, where joins are defined in direct style pattern notation. Their matching behavior is programmable via dedicated control abstractions for coordinating and aligning asynchronous streams. Our insight is that semantic variants of joins are definable as cartesian product computations with side effects influencing how the computation unravels. Based on this insight, we can afford working with a naive enumeration procedure of the cartesian product and turn it into efficient variants, by injection of appropriate effect handlers.

3.5 Effect Handlers for the Masses

Jonathan Immanuel Brachthäuser (*Universität Tübingen, DE*)

License © Creative Commons BY 3.0 Unported license

© Jonathan Immanuel Brachthäuser

Joint work of Jonathan Immanuel Brachthäuser, Philipp Schuster, Klaus Ostermann

Main reference Jonathan Immanuel Brachthäuser, Philipp Schuster, Klaus Ostermann, “Effect Handlers for the Masses”, under submission at the Conference on Object Oriented Programming Systems Languages & Applications, Boston, USA, 2018.

Algebraic effect handlers are a program structuring paradigm with rising popularity in the functional programming language community. Effect handlers are less wide-spread in the context of imperative, object oriented languages. We present library implementations of algebraic effects in Scala and Java. Both libraries are centered around the concept of handler/capability passing and a shallow embedding of effect handlers. While the Scala library is based on a monad for multi-prompt delimited continuations, the Java library performs a CPS translation as bytecode instrumentation. We discuss design decisions and implications on extensibility and performance.

References

- 1 Jonathan Immanuel Brachthäuser, Philipp Schuster. *Effekt: Extensible Algebraic Effects in Scala*. Proceedings of the 8th ACM SIGPLAN International Symposium on Scala, Vancouver, Canada, 2017
- 2 Jonathan Immanuel Brachthäuser, Philipp Schuster. *Effect Handlers for the Masses*. Under submission at the Conference on Object Oriented Programming Systems Languages & Applications, Boston, USA, 2018

3.6 Combining Algebraic Theories

Jeremy Gibbons (*University of Oxford, GB*)

License © Creative Commons BY 3.0 Unported license

© Jeremy Gibbons

Joint work of Kwok Cheung, Jeremy Gibbons

I summarized results due to Hyland, Plotkin, and Power [5, 6] on combining algebraic theories, as explored in the doctoral thesis [2] of my student Kwok Cheung. Specifically, the *sum* $S + T$ of algebraic theories S and T has all the operations of S and T , and all the equations, and no other equations. The *commutative tensor* $S \otimes T$ adds equations of the form

$$f(g(x_1, x_2), g(y_1, y_2), g(z_1, z_2)) = g(f(x_1, y_1, z_1), f(x_2, y_2, z_2))$$

for each operation f of one theory and g of the other. The *distributive tensor* $S \triangleright T$ adds to the sum equations of the form

$$\begin{aligned} f(g(x_1, x_2), y, z) &= g(f(x_1, y, z), f(x_2, y, z)) \\ f(x, g(y_1, y_2), z) &= g(f(x, y_1, z), f(x, y_2, z)) \\ f(x, y, g(z_1, z_2)) &= g(f(x, y, z_1), f(x, y, z_2)) \end{aligned}$$

for each operation f of S and g of T .

I also showed some examples:

- *global state* $S \rightarrow (1 + -) \times S$ arises as the sum of the theories *State* and *Failure*;
- *local state* $S \rightarrow 1 + (- \times S)$ arises as the commutative tensor $State \otimes Failure$;
- probability and nondeterminism interact via probabilistic choice $_w \oplus$ distributing over nondeterministic choice \square , but not vice versa, so arises as the distributive tensor $Prob \triangleright Nondet$;
- two-player games have both angelic choice \sqcup and demonic choice \sqcap , each of which distributes over the other, so arises as the two-way distributive tensor $Nondet \triangleleft \triangleright Nondet$ and some non-examples:
 - the *list monad* does not arise as any of these combinations of the theories *Nondet* (i.e., just binary choice, with associativity) and *Failure*;
 - the *list transformer done right* [3] applied to the monad arising from some theory T does not arise as any of these combinations of the monoidal theory of the list monad with T ;
 - symmetric bidirectional transformations [4] maintain two complementary data sources A and B in synchronization; they have the signature of two copies of the theory of *State*, but the two implementations are *entangled* [1]—the ‘get’ operations of A and B commute with each other, but the ‘set’ operation on one side does not commute with any operation on the the other.

I conjecture there are more such well-behaved and useful combinators on algebraic theories to be found, which might explain the above counter-examples and others like them.

References

- 1 Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Notions of bidirectional computation and entangled state monads. In Ralf Hinze and Janis Voigtländer, editors, *Mathematics of Program Construction*, volume 9129 of *Lecture Notes in Computer Science*, pages 187–214. Springer, 2015.
- 2 Kwok Cheung. *Distributive Interaction of Algebraic Effects*. Dphil thesis, University of Oxford, 2018.
- 3 Yitzhak Gale. ListT done right. https://wiki.haskell.org/ListT_done_right, 2007.
- 4 Martin Hofmann, Benjamin C. Pierce, and Daniel Wagner. Symmetric lenses. In Thomas Ball and Mooly Sagiv, editors, *Principles of Programming Languages*, pages 371–384. ACM, 2011.
- 5 Martin Hyland, Gordon D. Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1-3):70–99, 2006.
- 6 Martin Hyland and John Power. Discrete Lawvere theories and computational effects. *Theoretical Computer Science*, 366(1-2):144–162, 2006.

3.7 Handlers.Js: A Comparative Study of Implementation Strategies for Effect Handlers on the Web

Daniel Hillerström (University of Edinburgh, GB)

License © Creative Commons BY 3.0 Unported license

© Daniel Hillerström

Joint work of Sam Lindley, Robert Atkey, KC Sivaramakrishnan, Jeremy Yallop

Handlers for algebraic effects have steadily been gaining traction since their inception. This traction can be partly attributed to their wide application space which includes diverse programming disciplines such as concurrent programming, probabilistic programming, etc. They have been implemented in a variety of programming languages as either a native primitive or embedded via existing abstraction facilities.

The many different implementations have contributed to mapping out the implementation space for effect handlers. While the picture for how to implement effect handlers in native code is pretty clear, the picture for implementing effect handlers via embedding in high-level programming languages is more blurry. Embedding in JavaScript is currently the only viable option for implementing effect handlers on the Web.

In this talk, I will discuss and compare five viable different compilation strategies for effect handlers to JavaScript. Two of the five strategies are based on novel encodings via generators/iterators and generalised stack inspection, respectively. Although, I will discuss these compilation strategies in the context of JavaScript, they are not confined to JavaScript. The strategies are also viable in other high-level languages such as, say, Python.

3.8 First Class Dynamic Effect Handlers and Deep Finally Handling

Daan Leijen (Microsoft Research – Redmond, US)

License © Creative Commons BY 3.0 Unported license

© Daan Leijen

Main reference Daan Leijen, “First Class Dynamic Effect Handlers”, in TyDe’18, St. Louis, US, Sep. 2018.

URL <https://www.microsoft.com/en-us/research/publication/first-class-dynamic-effect-handlers>

We first show how “inject” combined with higher-ranked types can encode first-class polymorphic references. However, it is cumbersome to program this way as you need to “inject” carefully to select the right handler by position. To remedy this, we extend basic algebraic effect handlers with *first class dynamic effects* to refer to a handler directly by name. Dynamic effects add a lot more expressiveness but surprisingly only need minimal changes to the original semantics. As such, dynamic effects are a powerful abstraction but can still be understood and reasoned about as regular effect handlers. We illustrate the expressiveness of dynamic effects with first class event streams in CorrL and also model full polymorphic heap references without requiring any further primitives. Following this, we add “finally” and “initially” clauses to handlers to robustly deal with external resources. We show you generally need a form of “deep” finally handling to reliably invoke all outstanding “finally” clauses.

Note: the original title of the talk was “Algebraic Effects with Resources and Deep Finalization”; However, it was decided afterwards this naming caused too much confusion: “Resources” was already used for external resources in Matija’s thesis, and “Finalization” reminded of finalization in the object oriented world which is invoked by the GC and non-deterministic.

3.9 Encapsulating effects

Sam Lindley (University of Edinburgh, GB)

License  Creative Commons BY 3.0 Unported license
© Sam Lindley

3.9.1 Leaking effects

Naively composing effect handlers that produce and consume an intermediate effect leads to that effect leaking such that external instances are accidentally captured. I illustrate the problem in Frank and then show how Biernacki et al.’s lift operator resolves the issue. I then briefly discuss other possible solutions.

For the following, I assume basic familiarity with the Frank programming language [6]. Let us begin by defining in Frank a maybe data type, reader and abort interfaces along with effect handlers for them.

```
data Maybe X = nothing | just X

interface Reader X = ask : X
interface Abort = abort X : X

reads : {List S -> <Reader S>X -> [Abort]X}
reads []      <ask -> k> = abort!
reads (s :: ss) <ask -> k> = reads ss (k s)
reads _      x          = x

maybe : {<Abort>X -> Maybe X}
maybe <abort -> _> = nothing
maybe x           = just x
```

The `reads` handler interprets the `ask` command by reading the next value, if there is one, from the supplied list; if the list is empty then it raises the `abort` command. The `maybe` handler interprets `abort` using the maybe data type.

It seems natural to want to precompose `maybe` with `reads`. A naive attempt yields the following Frank code.

```
bad : {List S -> <Reader S, Abort>X -> Maybe X}
bad ss <m> = maybe (reads ss m!)
```

The `bad` handler handles `ask` as expected, yielding `just v` for some value `v` if input is not exhausted

```
bad [1,2,3] (ask! + ask! + ask!) == just 6
```

and `nothing` if input is exhausted:

```
bad [1,2] (ask! + ask! + ask!) == nothing
```

Alas, as indicated by its type, `bad` also exhibits additional behaviour. As well as handling any `abort` command raised by the `reads` handler, it also captures uses of `abort` from the ambient context:

```
bad [1,2,3] (ask! + ask! + abort!) == nothing
```

One might think that we could simply supply a different type signature to `bad` in order to suppress `Abort`. But the underlying problem is not with the types; it is with the dynamic semantics. Without some way of hiding the `Abort` effect there is no way of preventing the `maybe` handler from accidentally capturing any `abort` command raised by the ambient context.

The current version of Frank (April 2018) provides a solution to the effect encapsulation problem: Biernacki et al.’s `lift` operator [3], with which we can precompose `maybe` with `reads` as follows.

```
good : {List S -> <Reader S>X -> Maybe X}
good ss <m> = maybe (reads ss (lift <Abort> m!))
```

An effect (*ability* in Frank parlance), in Frank (much like Koka [5]) denotes a total map from interface names to finite lists of instantiations. Interfaces that are not present are denoted by empty lists. The head of a list denotes the active instantiation of an interface. The `lift` operator adds a dummy instantiation onto the head of the list associated with a given interface. Thus, the invocation `lift <Abort> m!` adds a dummy instantiation of `Abort` onto the ability associated with the computation `m!`. The dummy instantiation ensures that the `maybe` handler cannot accidentally capture `abort` commands raised by the ambient context. So

```
good [1,2,3] (ask! + ask! + abort!) : [Abort]Maybe Int
```

and:

```
maybe (good [1,2,3] (ask! + ask! + abort!)) == nothing
```

The `lift` operator provides a means for hiding effects reminiscent of de Bruijn representations for bound names. It generates a fresh instantiation of an interface by shifting all of the existing instances along by one.

3.9.2 Concurrency

In his Master’s dissertation, Lukas Convent identified the effect encapsulation problem in the context of a previous version of Frank [4] that did not provide support for `lift`. He presents a number of examples of trying to compose effect handlers together in order to implement various forms of concurrency. The resulting types expose many of the internal implementation details illustrating how important the problem is to solve.

To illustrate how `lift` helps to solve these problems I include an adaptation of code from Convent’s thesis to implement Erlang-style concurrency based on an actor abstraction in Frank by composing together a number of different handlers. The adapted code takes advantage of `lift`.

```
include prelude

-----
-- Queue interface and FIFO implementation using a zipper
-----

interface Queue S = enqueue : S -> Unit
                  | dequeue : Maybe S

-- zipper queue
data ZipQ S = zipq (List S) (List S)
```

```

emptyZipQ : {ZipQ S}
emptyZipQ! = zipq [] []

-- FIFO queue implementation using a zipper
-- (returns the remaining queue alongside the final value)
runFifo : {ZipQ S -> <Queue S>X -> Pair X (ZipQ S)}
runFifo (zipq front back) <enqueue x -> k> = runFifo (zipq front (x :: back)) (k unit)
runFifo (zipq [] []) <dequeue -> k> = runFifo emptyZipQ! (k nothing)
runFifo (zipq [] back) <dequeue -> k> = runFifo (zipq (rev back) []) (k dequeue!)
runFifo (zipq (x :: front) back) <dequeue -> k> = runFifo (zipq front back) (k (just x))
runFifo queue x = pair x queue

-- discard the queue
evalFifo : {<Queue S>X -> ZipQ S -> X}
evalFifo <t> q = case (runFifo q t!) { (pair x _) -> x }

-- start with an empty queue
fifo : {<Queue S>X -> X}
fifo <m> = evalFifo m! (emptyZipQ!)

-- discard the value
execFifo : {<Queue S>X -> ZipQ S -> ZipQ S}
execFifo <t> q = case (runFifo q t!) { (pair _ q) -> q }

-----
-- Definitions of interfaces, data types
-----

interface Co = fork : {[Co]Unit} -> Unit
              | yield : Unit

data Mailbox X = mbox (Ref (ZipQ X))

interface Actor X = self : Mailbox X
                  | spawn Y : {[Actor Y]Unit} -> Mailbox Y
                  | recv : X
                  | send Y : Y -> Mailbox Y -> Unit

data WithSender X Y = withSender (Mailbox X) Y

-----
-- Example actor
-----

spawnMany : {Mailbox Int -> Int -> [Actor Int [Console], Console]Unit}
spawnMany p 0 = send 42 p
spawnMany p n = spawnMany (spawn {let x = recv! in print "."; send x p}) (n-1)

chain : {[Actor Int [Console], Console]Unit}
chain! = spawnMany self! 640; recv!; print "\n"

-----
-- Implement an actor computation as a stateful concurrent computation
-----

-- Our syntactic sugar assumes that all instances of the implicit
-- effect variable are instantiated to be the same but they needn't be
-- the same as the ambient effects.
--
-- For liftBody we need exactly that all but the ambient effects be
-- the same.
liftBody : {[Actor X]Unit} -> [E |][[Actor X, Co [RefState], RefState]Unit]}
liftBody m = {lift <RefState, Co> m!}

act : {Mailbox X -> <Actor X>Unit -> [Co [RefState], RefState]Unit}

```

```

act mine      <self -> k> = act mine (k mine)
act mine      <spawn you -> k> = let yours = mbox (new (emptyZipQ!)) in
                                fork {act yours (liftBody you)!};
                                act mine (k yours)
act (mbox m) <recv -> k> = case (runFifo (read m) dequeue!)
                              { (pair nothing _)   -> yield!;
                                act (mbox m) (k recv!)
                                | (pair (just x) q) -> write m q;
                                act (mbox m) (k x) }
act mine      <send x (mbox m) -> k> = let q = execFifo (enqueue x) (read m) in
                                write m q;
                                act mine (k unit)

act mine      unit = unit

runActor : {<Actor X>Unit -> [RefState]Unit}
runActor <m> = bfFifo (act (mbox (new emptyZipQ!)) (lift <Co> m!))

bfFifo : {<Co>Unit -> Unit}
bfFifo <m> = fifo (scheduleBF (lift <Queue> m!))

-----
-- Scheduling
-----

data Proc = proc {[Queue Proc]Unit}

enqProc : {[Queue Proc]Unit} -> [Queue Proc]Unit
enqProc p = enqueue (proc p)

runNext : {[Queue Proc]Unit}
runNext! = case dequeue! { (just (proc x)) -> x!
                           | nothing       -> unit }

-- defer forked processes (without effect pollution)
scheduleBF : {<Co>Unit -> [Queue Proc]Unit}
scheduleBF <yield -> k> = enqProc {scheduleBF (k unit)};
                           runNext!
scheduleBF <fork p -> k> = enqProc {scheduleBF (lift <Queue> p!)};
                           scheduleBF (k unit)
scheduleBF unit         = runNext!

-- eagerly run forked processes
scheduleDF : {<Co>Unit -> [Queue Proc]Unit}
scheduleDF <yield -> k> = enqProc {scheduleDF (k unit)};
                           runNext!
scheduleDF <fork p -> k> = enqProc {scheduleDF (k unit)};
                           scheduleDF (lift <Queue> p!)
scheduleDF unit         = runNext!

main : {[Console, RefState]Unit}
main! = runActor (lift <RefState> chain!)

```

This code implements actors using a coroutining concurrency interface, which in turn is implemented using an interface for queues of processes, which are implemented using a simple zipper data structure. The example `chain` spawns a collection of processes and passes a message through the entire collection.

The crucial point is that the type of `runActor` mentions only the `Actor` interface that is being handled and the `RefState` interface that is being used to implement it. Convent's original code leaks out the `Queue` interface and the `Co` interface. Moreover, the type becomes particularly hard to read because some of the interfaces are parameterised by several effects.

3.9.3 Discussion

The designers of the Eff programming language [2] have explored several different designs for effect handlers and effect type systems for effect handlers. Some versions of Eff provide features that address the effect encapsulation problem to some degree. The earliest version of Eff [1] resolved the encapsulation problem by supporting the generation of fresh instances of an effect. This was relatively straightforward as at the time Eff did not provide an effect type system. A later version of Eff included a somewhat complicated effect type system with support for instances that used a region-like effect type system [7]. The latest version of Eff at the time of writing [8] does not appear to offer a solution to the effect encapsulation problem. It provides an effect type system with subtyping, without duplicate effect interfaces and no support for effect instances.

For effect systems that allow only one copy of each effect interface we might solve the effect encapsulation problem by adding a primitive for introducing a fresh copy of an interface. For instance, to hide the intermediate `Abort` interface we could generate a fresh copy of `Abort` – call it `Abort'` – which we could then use to rename any `abort` commands in the ambient context to `abort'` before renaming them back after running the `reads` and `maybe` handlers.

An as yet unimplemented Frank feature that is related to effect encapsulation is negative adjustments [6]. Currently an adjustment in Frank always adds interfaces to the ambient ability, and specifies which interfaces must be handled. Negative adjustments would allow interfaces to be removed, enabling the programmer to specify that a computation being handled does not support some of the interfaces in the ambient. Roughly, the `lift` operation is the inverse of a negative adjustment. It remains to be seen what the relative pros and cons of negative adjustments and `lift` are in practice.


More generally, there is a broad design space for effect type systems that support effect encapsulation and it is worth considering the full range of options.

References

- 1 Andrej Bauer and Matija Pretnar. Programming with algebraic effects and handlers. *J. Log. Algebr. Meth. Program.*, 84(1):108–123, 2015.
- 2 Andrej Bauer and Matija Pretnar. Eff, 2018.
- 3 Dariusz Biernacki, Maciej Piróg, Piotr Polesiuk, and Filip Sieczkowski. Handle with care: relational interpretation of algebraic effects and handlers. *PACMPL*, 2(POPL):8:1–8:30, 2018.
- 4 Lukas Convent. Enhancing a modular effectful programming language. Master’s thesis, The University of Edinburgh, Scotland, 2017.
- 5 Daan Leijen. Type directed compilation of row-typed algebraic effects. In *POPL*, pages 486–499. ACM, 2017.
- 6 Sam Lindley, Conor McBride, and Craig McLaughlin. Do be do be do. In *POPL*, pages 500–514. ACM, 2017.
- 7 Matija Pretnar. Inferring algebraic effects. *Logical Methods in Computer Science*, 10(3), 2014.
- 8 Amr Hany Saleh, Georgios Karachalias, Matija Pretnar, and Tom Schrijvers. Explicit effect subtyping. In *ESOP*, volume 10801 of *Lecture Notes in Computer Science*, pages 327–354. Springer, 2018.

3.10 Experiences with structuring effectful code in Haskell


Andres Löh (*Well-Typed LLP, DE*)

License  Creative Commons BY 3.0 Unported license
© Andres Löh

Effectful Haskell code that is written using monad transformers can easily become difficult to maintain. However, it is unclear whether algebraic effects do not suffer from the same problem. Both approaches seem to encourage specifying a minimal amount of effects for each code fragment, leading to effect constraints being propagated in a bottom-up fashion throughout the program, often without much thought for control. In this talk, I argue that sometimes, it is better to be less general, by identifying just a few meaningful interfaces in a program, corresponding to different sets of available effects, and keeping testing in mind. These interfaces are then pushed down, and code is merely checked to not use effects that are outside of the allowed subset.

3.11 Make Equations Great Again!

Matija Pretnar (*University of Ljubljana, SI*)

License  Creative Commons BY 3.0 Unported license
© Matija Pretnar
Joint work of Žiga Lukšič, Matija Pretnar

Algebraic effects have originally been presented with equational theories, i.e. a set of operations and a set of equations they satisfy. Since a significant portion of computationally interesting handlers overrides the effectful behaviour in a way that invalidates the equations, most approaches nowadays assume an empty set of equations.


At the Dagstuhl Seminar 16112, I presented an idea in which the equations are represented locally in computation types [1]. In this way, handlers that do not respect all equations are not rejected but receive a weaker type. In the talk, I presented the progress made and questions that remain open.

References

- 1 Matija Pretnar. Capturing algebraic equations in an effect system. In *Dagstuhl Seminar 16112*, pages 55–57. 2016. DOI: 10.4230/DagRep.6.3.44

3.12 Quirky handlers

Matija Pretnar (*University of Ljubljana, SI*) and Žiga Lukšič (*University of Ljubljana, SI*)


License  Creative Commons BY 3.0 Unported license
© Matija Pretnar and Žiga Lukšič

Programming language terms are usually represented with an inductive type that lists all their possible constructors. It turns out that most functions on such a type are routine. For example, the set of free variables in a given arithmetic expression is almost always the union of free variables in subterms (except if the expression itself is a variable). Still, we must treat every single case in the function definition, and this quickly becomes annoying.

There are many ways in which this problem can be avoided, for example using open recursion or type classes. In the talk, we will see how to use handlers to define such functions with as little boilerplate as possible, yet ensure that the compiler forces us to revisit each part of the code when the type definition changes.

3.13 What is coalgebraic about algebraic effects and handlers?

Matija Pretnar (*University of Ljubljana, SI*)

License  Creative Commons BY 3.0 Unported license
© Matija Pretnar

In this tutorial we reviewed the work of Gordon Plotkin and John Power [1] in which they proposed comodels of algebraic theories and tensoring of comodels and models as a mathematical model for the interaction of an effectful program with its external environment. A comodel of a theory T in a category C is a model of T in the opposite category C^{op} , and the category of comodels in C is the opposite of the category of models in C^{op} . We may define the tensor $M \otimes W$ of a comodel W and a model M , which is a certain quotient of the product $M \times W$. A pair $(p, w) \in M \times W$ may be viewed as a program p running in the external environment w . The comodel W provides resources needed for execution of algebraic operations in M . In the tutorial we emphasized the fact that comodels and tensoring are a more appropriate model of top-level behavior of effectful program than various notions of “top-level” or “default” handlers. A handler has access to the continuation, but at the top level this is not the case, or else programs would be able to control the external world. It has to be the other way around.

References

- 1 Gordon D. Plotkin and John Power. Tensors of comodels and models for operational semantics. *Electronic Notes in Theoretical Computer Science*, 218:295–311, 2008.

3.14 Effect Handlers for WebAssembly (Show and Tell)

Andreas Rossberg (*Dfinity Foundation, CH*)

License  Creative Commons BY 3.0 Unported license
© Andreas Rossberg

Integrating effects into Wasm involves a number of complications that do not exist in more high-level (and purer) languages. For example, the presence of branches interacts in interesting ways with try blocks, handlers, and continuations. It necessitates a stricter distinction between exceptions and effects. That in turn complicates the semantics and its formalisation.

We worked out a the semantics for handlers in Wasm as an extension to the existing proposal for exception handling. The next, far more challenging step will be to implement them in a production engine in order to validate their practical feasibility and evaluate their real-world performance.

3.15 Neither Web Nor Assembly

Andreas Rossberg (*Dfinity Foundation, CH*)

License © Creative Commons BY 3.0 Unported license
© Andreas Rossberg

WebAssembly (or “Wasm”) [1] is a portable high-performance code format that has been designed not just for the Web but for a broad range of embedding environments. It is standardised and fully formalised using well-established techniques from programming language theory. Version 1 of WebAssembly, which is currently available, was deliberately limited in scope to encompass low-level programming languages such as C++. For the next stage, more support for high-level languages will be added.

One central requirement is the ability to express the large variety of control abstractions that appear in high-level languages, such as coroutines, light-weight threads, generators, and asynchronous computations. At the lowest level, their commonality is the need to “switch stacks”. However, ad-hoc mechanisms for doing so are inadequate for WebAssembly, due to its nature of a code format that must be sufficiently high-level to guarantee safety, and due to the desire to maintain a high-assurance formal specification. That asks for a primitive with strong semantic foundations.

Effect handlers would fit the bill perfectly. But it is an open question how exactly they can be designed in the context of a low-level stack machine and whether they can be implemented efficiently under the constraints imposed on existing WebAssembly implementations.

References

- 1 A. Haas, A. Rossberg, D. Schuff, B. Titzer, D. Gohman, L. Wagner, A. Zakai, J.F. Bastien, M. Holman. *Bringing the Web up to Speed with WebAssembly*. PLDI 2017.

3.16 Efficient Compilation of Algebraic Effects and Handlers

Tom Schrijvers (*KU Leuven, BE*)

License © Creative Commons BY 3.0 Unported license
© Tom Schrijvers
Joint work of Amr Hany Saleh, Axel Faes, Georgios Karachalias, Matija Pretnar, Tom Schrijvers


The popularity of algebraic effect handlers as a programming language feature for user-defined computational effects is steadily growing. Yet, even though efficient runtime representations have already been studied, most handler-based programs are still much slower than hand-written code.

In this paper we show that the performance gap can be drastically narrowed (in some cases even closed) by means of type-and-effect directed optimising compilation. Our approach consists of two stages. Firstly, we combine elementary source-to-source transformations with judicious function specialisation in order to aggressively reduce handler applications. Secondly, we show how to elaborate the source language into a handler-less target language in a way that incurs no overhead for pure computations.

This work comes with a practical implementation: an optimizing compiler from Eff, an ML style language with algebraic effect handlers, to OCaml. Experimental evaluation with this implementation demonstrates that in a number of benchmarks, our approach eliminates much of the overhead of handlers and yields competitive performance with hand-written OCaml code.

3.17 Algebraic effects – specification and refinement


Wouter Swierstra (*Utrecht University, NL*)

License  Creative Commons BY 3.0 Unported license
© Wouter Swierstra

How should we reason about programs written with algebraic effects? As the meaning of a program is determined by its handlers, we need a way to specify the intended behaviour of handlers. In this talk, I sketched an approach based on predicate transformers that enables the calculation of effectful programs from their specification.

3.18 Adding an effect system to OCaml

Leo White (*Jane Street – London, GB*)

License  Creative Commons BY 3.0 Unported license
© Leo White
Joint work of Stephen Dolan, Matija Pretnar, Sivaramakrishnan Krishnamoorthy Chandrasekaran, Daniel Hillerström

Type systems designed to track the side-effects of expressions have been around for many years but they have yet to breakthrough into more mainstream programming languages. This talk focused on on-going work to add an effect system to OCaml.

This effect system is primarily motivated by the desire to keep track of algebraic effects in the OCaml type system. Through the Multicore OCaml project, support for algebraic effects is likely to be included in OCaml in the near future. However, the effect system also allows for tracking side-effects more generally. It distinguishes impure functions, which perform side-effects, from pure functions, which do not. It also includes a tracked form of exception to support safe and efficient error handling.

This talk gave an overview of the effect system and demonstrated a prototype implementation on some practical examples.

3.19 Multi-Stage Programming with Algebraic Effects

Jeremy Yallop (*University of Cambridge, GB*)

License  Creative Commons BY 3.0 Unported license
© Jeremy Yallop

I showed how algebraic effects and handlers are useful in *multi-stage* programming (a kind of programmer-directed, annotation-driven form of partial evaluation). There is a long tradition of using continuations and continuation-passing style to improve the results of partial evaluation and multi-stage programming [4, 3]. However, algebraic effects and handlers lead to a particularly elegant formulation of various transformations of the code generated by multi-stage programs.

The running example in the talk was the staging of a standard functional program – typed `printf/scanf` [1] — using BER MetaOCaml’s staging facilities [3] and Multicore OCaml’s implementation of effects [2]. While naively staging the program produces some performance improvements, the generated code is still sub-optimal. Several transformations, conveniently expressed using algebraic effects, significantly improve the output:

- `let` insertion untangles nested computations
- normalization of destructuring `let` bindings avoids repeated tupling and detupling
- insertion of branches into the generated code exposes information about future-stage values in each branch (such as whether a boolean variable will have the value `true` or `false` in a particular region of the program), enabling further optimizations. This last transformation makes essential use of *multi-shot continuations*.

Some of these techniques are described in more detail in recent work [5].

References

- 1 O. Danvy. Functional unparsing. *J. Funct. Program.*, 8(6):621–625, Nov. 1998.
- 2 S. Dolan, L. White, K. Sivaramakrishnan, J. Yallop, and A. Madhavapeddy. Effective concurrency through algebraic effects. OCaml Users and Developers Workshop 2015, September 2015.
- 3 O. Kiselyov. The design and implementation of BER metaocaml – system description. In *Functional and Logic Programming – 12th International Symposium, FLOPS 2014, Kanazawa, Japan, June 4-6, 2014. Proceedings*, pages 86–102, 2014.
- 4 J. L. Lawall and O. Danvy. Continuation-based partial evaluation. *SIGPLAN Lisp Pointers*, VII(3):227–238, July 1994.
- 5 J. Yallop. Staged generic programming. *Proc. ACM Program. Lang.*, 1(ICFP):29:1–29:29, Aug. 2017.

4 Working groups

4.1 Denotational Semantics for Dynamically Generated Effects

Robert Atkey (University of Strathclyde – Glasgow, GB)

License © Creative Commons BY 3.0 Unported license
© Robert Atkey

We discussed a possible worlds / functor category semantics for a simple language with dynamically generated effect names and handlers. In this semantics, types are indexed by effect signatures, describing the set of possible effect names in scope, and computations are given a semantics in terms of a monad that supports ‘free’ operations from the current effect signature world, and the possible generation of new effect names in the style of Stark’s monads for name generation. Some interesting program equivalences were also discussed, and it was noted that they depend on whether or not effect names can “leak” out of their scope, usually via higher-order state. The encoding of ML-style references using handlers was also discussed. This requires that the “arities” of effects can also include effect names.

4.2 Reasoning with Effects

Jeremy Gibbons (University of Oxford, GB)

License  Creative Commons BY 3.0 Unported license
© Jeremy Gibbons

We discussed a number of topics around the equations of algebraic effects and handlers:

- general concerns about the equations of an algebraic theory often being ignored
- should the equations be thought of as being attached to the operations of a theory or to the handler(s) for that theory?
- where do the equations come from? the programmer's intentions? QuickCheck exploration of the properties of a handler?

Participants

- Amal Ahmed
Northeastern University – Boston, US
- Robert Atkey
University of Strathclyde – Glasgow, GB
- Lennart Augustsson
X Inc – Mountain View, US
- Andrej Bauer
University of Ljubljana, SI
- Oliver Bracevac
TU Darmstadt, DE
- Jonathan Immanuel
Brachthäuser
Universität Tübingen, DE
- Edwin Brady
University of St Andrews, GB
- Stephen Dolan
University of Cambridge, GB
- Jeremy Gibbons
University of Oxford, GB
- Daniel Hillerström
University of Edinburgh, GB
- Mauro Jaskelioff
National University of Rosario, AR
- Ohad Kammar
University of Oxford, GB
- Andrew Kennedy
Facebook – London, GB
- Oleg Kiselyov
Tohoku University – Sendai, JP
- Sivaramakrishnan
Krishnamoorthy Chandrasekaran
University of Cambridge, GB
- Daan Leijen
Microsoft Research – Redmond, US
- Sam Lindley
University of Edinburgh, GB
- Andres Löb
Well-Typed LLP, DE
- Žiga Lukšič
University of Ljubljana, SI
- Anil Madhavapeddy
Docker Inc. – Cambridge, GB
- Conor McBride
University of Strathclyde – Glasgow, GB
- Adriaan Moors
Lightbend Inc. – Lausanne, CH
- Matija Pretnar
University of Ljubljana, SI
- Andreas Rossberg
Dfinity Foundation, CH
- Tom Schrijvers
KU Leuven, BE
- Perdita Stevens
University of Edinburgh, GB
- Wouter Swierstra
Utrecht University, NL
- Leo White
Jane Street – London, GB
- Nicolas Wu
University of Bristol, GB
- Jeremy Yallop
University of Cambridge, GB



Towards Accountable Systems

Edited by

David Eysers¹, Christopher Millard², Margo Seltzer³, and
Jatinder Singh⁴

1 University of Otago, NZ, dme@cs.otago.ac.nz

2 Queen Mary University of London, GB, c.millard@qmul.ac.uk

3 Harvard University – Cambridge, US, margo@eecs.harvard.edu

4 University of Cambridge, GB, js573@cam.ac.uk

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18181 “Towards Accountable Systems”, which took place from April 29th to May 4th, 2018, at Schloss Dagstuhl – Leibniz Center for Informatics. Researchers and practitioners from academia and industry were brought together covering broad fields from computer and information science, public policy and law.

Many risks and opportunities were discussed that relate to the alignment of systems technologies with developing legal and regulatory requirements and evolving user expectations.

This report summarises outcomes of the seminar by highlighting key future research directions and challenges that lie on the path to developing systems that better align with accountability concerns.

Seminar April 29–May 4, 2018 – <http://www.dagstuhl.de/18181>

2012 ACM Subject Classification Applied computing → Law, Computer systems organization → Cloud computing, Security and privacy, Social and professional topics → Governmental regulations, Social and professional topics → Technology audits

Keywords and phrases accountability, compliance, audit, systems, engineering, cloud computing, internet of things, law, regulation, GDPR, security, privacy, data provenance

Digital Object Identifier 10.4230/DagRep.8.4.126

Edited in cooperation with Jennifer Cobbe


1 Executive Summary

David Eysers (University of Otago, NZ)

Christopher Millard (Queen Mary University of London, GB)

Margo Seltzer (Harvard University – Cambridge, US)

Jatinder Singh (University of Cambridge, GB)

License  Creative Commons BY 3.0 Unported license

© David Eysers, Christopher Millard, Margo Seltzer, and Jatinder Singh

Background and Motivation

Technology is becoming increasingly pervasive, impacting all aspects of everyday life. Our use of apps and online services is tracked and extensively processed (data analytics), and the results are used for various purposes, predominately advertising. Monitoring and surveillance by sensors in smart cities creates vast amounts of data, much of which can be identifiably linked with people. Smart home, health and lifestyle monitoring, and other sensor technologies yield sensitive personal data; mobile phones reveal people’s positions, and their calls are



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Towards Accountable Systems, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 126–163

Editors: David Eysers, Christopher Millard, Margo Seltzer, and Jatinder Singh



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

tracked leading to data that can be used to determine social linkages and sometimes mental wellbeing. Such collection and analysis of personal data raises serious privacy concerns. A key aspiration is to provide end-users with a means to understand their digital footprints, and control the propagation, aggregation and retention of their data.

Concerns over data movement, location, processing and access have led to increasing regulation, both national and international. An example is the recently adopted EU General Data Protection Regulation (GDPR) that reinforces and expands individual rights, as well as restrictions and obligations regarding personal data. However, data moves easily beyond geographical boundaries, and use of cloud computing resources may mean that stored data may be replicated in multiple locations worldwide, with potential for conflicts between applicable laws and jurisdictions. Governments may demand access to data (whether stored locally or remotely) and this may result in complex legal disputes. Regulations, codes of conduct, and best practices can incentivise the use of particular technical mechanisms for data management. Examples include encryption and anonymisation, for example when using medical data for research. However, there are often misalignments between legal/regulatory aims and the capabilities of the technologies.

Key issues concern how to demonstrate compliance with regulations, such as those regarding how data is handled and used, and, in cases of failure, how to hold the appropriate entities accountable. This is a particular challenge for wide-scale, federated, or cross-border systems. In large or complex systems, data may be handled by many different parties, falling under various management regimes and jurisdictions. Such concerns are not only horizontal (e.g., data being exchanged between parties, across geographic regions) but also vertical, where different levels of the services stack are managed by different parties (e.g., a company application running over a Heroku PaaS that runs over Amazon IaaS). Most end-users (people!) are oblivious to the potential complexity of such systems, let alone the complexity of the legal requirements that underpin such architectures. In general, the lack of transparency and uncertainty about the means for compliance with legal obligations, along with a lack of technical means for managing such concerns, may inhibit innovative technology development (a “chilling factor”), may escalate compliance costs, may trigger inappropriate policy responses, and may work to undermine public trust in technology.

These concerns will only grow in prominence, given the increasing deployment of sensors, generating ever-more data; actuators, giving systems physical effects; and the use of machine learning, facilitating automation. In response, this seminar brought together experts from the computer science and legal communities, spanning academia and industry, to explore issues of accountability as it relates to data and systems. The seminar aimed to: (i) raise awareness of and establish new research directions concerning issues of accountability as they relate to systems, given directions in systems technologies; (ii) explore developing legal and regulatory requirements; and (iii) investigate issues of user empowerment. A key goal was to increase awareness that law, regulation and requirements for data usage, management, security, confidentiality, quality and provenance should align with the technology, and *vice versa*: technologists should be legally-aware and lawyers should be technology-aware.

Seminar Structure

Due to the diverse backgrounds of the participants, the first day was focused on introductions and ensuring that everyone had a common grounding in key topics. This included a series of guided discussion sessions: Lilian Edwards provided an introduction to legal and regulatory considerations, particularly the European Union General Data Protection Regulation (GDPR); Jon Crowcroft introduced emerging technical architectures such as edge computing; Bertram

Ludäscher led a session exploring data provenance; and Ben Wagner introduced broader ethical and social concerns. A motivating case study was also presented highlighting how an apparently enthusiastic view of emerging Internet of Things technologies might obscure a plethora of questionable social and policy implications.

The structure of the week included multiple breakout sessions in which working groups examined particular topics (below) and reported back summaries of their discussions at plenary sessions. The working group sessions were interspersed with an interactive case study session, that focused on the technological compliance concerns of a hypothetical global hotel chain seeking to introduce a series of IoT and cloud technologies in the current regulatory environment, and a session in which participants were able to present their recent research, abstracts for (most of) which are included in this report.

Moving forward

The topics explored by the working groups at the seminar spanned policy, legal and technical considerations. The topics were seeded by the organisers but were ultimately gathered from the participants through a preference allocation process. The chosen topics included:

- *Trust in systems.*
- *Who is, could or should be accountable in complex systems?*
- *Engineering accountable systems.*
- *Is there a place for data provenance in accountable systems?*
- *Anonymity, identity and accountability.*
- *Thinking beyond consent.*
- *Automating the exercising of rights for collective oversight.*

Each group was asked to produce an abstract summarising the key issues, challenges and ways forward from the discussion. These abstracts are included in this report, and indicate many potential opportunities for research.

Generally, it was felt that *this seminar represented only the start of this important discussion*. It is clear that there is a substantial and urgent need for closer interactions between the technical and legal domains, such that (i) the computer science communities better understand the legal requirements and constraints that impact the design, implementation and deployment of technology; and (ii) the legal communities gain more of a grounding in the nature, capabilities, and potential of the technology itself. It was also recognised that there is potential for better collaboration amongst different computer science communities; for example, to have greater interactions between those working in systems, provenance and machine learning.

In light of this, key to moving forward is to work to form collaborative research proposals, and to organise relevant meetings, in order to drive progress on the topics, challenges and research opportunities identified during this seminar. As issues of accountability increase in importance and urgency, it is vital that researchers across academia, industry and civil society work together to proactively confront these challenges.

2 Table of Contents

Executive Summary

David Evers, Christopher Millard, Margo Seltzer, and Jatinder Singh 126

Working groups

Trust in Systems

Jennifer Cobbe, Jon Crowcroft, David Evers, Krishna P. Gummadi, Joshua A. Kroll, Derek McAuley, Michael Veale, and Michael Winikoff 131

Who is Accountable?

Ben Wagner, Virgilio Almeida, Tristan Henderson, Heleen Louise Janssen, Christopher Millard, and Barbara Staudt Lerner 132

Engineering Accountable Systems

Martin Henze, Virgilio Almeida, Jean Bacon, Melanie Herschel, Maximilian Ott, Frank Pallas, Thomas Pasquier, Silvia Puglisi, Margo Seltzer, Michael Winikoff, and Martina Zitterbart 137

Provenance for Accountable Systems

Lilian Edwards, Melanie Herschel, Bertram Ludäscher, Ken Moody, Thomas Pasquier, and Jatinder Singh 141

Anonymity, Identity and Accountability

Jean Bacon, Heleen Louise Janssen, Joshua A. Kroll, Silvia Puglisi, Jatinder Singh, and Martina Zitterbart 144

Post-Consent

Jennifer Cobbe, Martin Henze, Bertram Ludäscher, Ken Moody, Maximilian Ott, and Margo Seltzer 147

Automating Data Rights

Michael Veale, Lilian Edwards, David Evers, Tristan Henderson, Christopher Millard, and Barbara Staudt Lerner 149

Overview of Talks

Tutorial–Cloudy with a hint of accountability

Jon Crowcroft 156

Tutorial–ML : transparency, control, user rights and the GDPR

Lilian Edwards 156

Tutorial–A Brief Introduction to Provenance in Workflows and Databases

Bertram Ludäscher 157

Tutorial–Ethical/social: Rights, Ethics and Accountability

Ben Wagner 157

Raising Users' Awareness for their Exposure to Cloud Services

Martin Henze 158

Purpose-driven provenance solutions

Melanie Herschel 159

Accountable systems. Some practical experiences from a governmental perspective.

Heleen Louise Janssen 159

Establishing Requirements for Accountable Systems: The Case of Elections	
<i>Joshua A. Kroll</i>	160
Confidential Analytics – Use Cases and Building Blocks	
<i>Maximilian Ott</i>	161
Accountable systems, messy environments	
<i>Michael Veale</i>	161
Trust, Responsibility, and Explanation	
<i>Michael Winikoff</i>	162
Participants	163

3 Working groups

3.1 Trust in Systems

Jennifer Cobbe (University of Cambridge, GB), Jon Crowcroft (University of Cambridge, GB), David Eysers (University of Otago, NZ), Krishna P. Gummadi (MPI-SWS – Saarbrücken, DE), Joshua A. Kroll (University of California – Berkeley, US), Derek McAuley (University of Nottingham, GB), Michael Veale (University College London, GB), and Michael Winikoff (University of Otago, NZ)

License © Creative Commons BY 3.0 Unported license

© Jennifer Cobbe, Jon Crowcroft, David Eysers, Krishna P. Gummadi, Joshua A. Kroll, Derek McAuley, Michael Veale, and Michael Winikoff

At the outset the group noted the difficulty of defining trust, acknowledging that it means very different things in different contexts. It was noted that this is an inherently interdisciplinary area, and, with participants being from computer science and law backgrounds, it was felt that a proper study of trust in systems would benefit from the input of others from disciplines such as philosophy, psychology, and sociology. It was also noted that there may be circumstances in which distrust (which may not necessarily result in not using a system) is more appropriate than trust. However, trust was generally recognised as being the “beliefs/faith you need to have about the things that you cannot verify”, and the group eventually settled on trust in this context as involving “confidence in the behaviour of a system”.

Various factors which affect trust/distrust were identified. Social factors may affect which individuals, organisations, and systems are trusted. It was acknowledged that well-developed and properly-enforced regulatory frameworks can help engender trust. People often also rely on mental models and folk theories; this has been shown, for example, in relation to social media feeds, with people becoming upset when their theories fail [1][2]. Contextual factors were also felt to be important, given that the consequences of a system failure can be very different for different people—for example, a robot tripping up an elderly person may be a much bigger problem than if it was someone younger. Cultural factors are also important; concepts of privacy, for example, differ between geographies, communities, and demographics. Finally, it was noted that infrastructure, including security infrastructure, plays a role in engendering trust.

The group moved on to discuss technological features which could be engineered into systems to assist trustworthiness. Predictability, reliability, and repeatability were felt to help manage user expectations of their interactions with systems, as trust may result where systems behave as expected. While transparency was generally felt to be a useful means of improving trustworthiness, it was acknowledged that there are unresolved questions over the extent to which it is useful and over what kind of information should be exposed. Safety marks were considered, but the difficulty of verifying software poses a problem. Engineering for security and privacy were also considered to be important factors in developing trustworthy systems. Finally, giving users control was considered to be an important feature of trust, as it was felt that users may lose trust when a system is beyond their control.

Following this discussion, the case study selected was that of a voice-controlled robotic vacuum cleaner. This interacts in a complex way with its environment, and use of such a device requires trust in the robot, trust in the organisation who makes the robot, and trust in the infrastructure involved in the robot. It was felt important that it would do what was expected, but also that it would not do anything unexpected (it should work when turned on, should stop when it is turned off, etc.). It should take context into account so

as to be unobtrusive and safe, could refuse to take actions which would be unsafe, and the consequences of failure should be considered and mitigated so far as possible so as to avoid harm when faults occur. Given that the robot would be voice-controlled, it should be clear about when it is listening and when it isn't, and a clear off switch for the microphone should be available. The organisation who makes the robot should not retrospectively decide to sell user data from devices which were sold on the basis that this would not occur. The group also considered that a button which would explain why the robot had taken an action would help engender trust in its decisions.

In terms of next steps, it was felt that there was a need for significant research in this area, including on what trust means in different contexts, on what factors affect trust and distrust, and on what technical approaches could potentially engender trust so as to help engineer trustworthy systems. It was felt that there was a need for a toolkit which would help people assess their own interpretation of trust and trustworthiness. The group agreed that it was important that trustworthy components or features not be hoarded commercially, to avoid the development of an industry that is paid by the existence of the problem. Finally, the group concluded that it was important to avoid having to repeatedly start this work from scratch—there is a need for work to be done on developing a knowledge base which can be compositional in terms of concerns, intentions, solutions, and so on.

References

- 1 Motahhare Eslami, Karrie Karahalios, Christian Sandvig, Kristen Vaccaro, Aimee Rickman, Kevin Hamilton, Alex Kirlik, 2016, *First I “like” it, then I hide it: Folk theories of social feeds*, CHI 2016 – Proceedings, 34th Annual CHI Conference on Human Factors in Computing Systems
- 2 Taina Bucher, 2017 *The algorithmic imaginary: exploring the ordinary affects of Facebook algorithms*, Information, Communication and Society, 20(1), pp.30–44

3.2 Who is Accountable?

Ben Wagner (Wirtschaftsuniversität Wien, AT), Virgilio Almeida (Federal University of Minas Gerais-Belo Horizonte, BR), Tristan Henderson (University of St Andrews, GB), Heleen Louise Janssen (Ministry of the Interior and Kingdom Relations, NL), Christopher Millard (Queen Mary University of London, GB), and Barbara Staudt Lerner (Mount Holyoke College – South Hadley, US)

License © Creative Commons BY 3.0 Unported license

© Ben Wagner, Virgilio Almeida, Tristan Henderson, Heleen Louise Janssen, Christopher Millard, and Barbara Staudt Lerner

Who is or could or should be accountable / responsible in complex systems?

3.2.1 Identify specific issues that more precisely describe it in a specific use case

- **Specific use case:** a connected camera used in a smart home. The camera is used to open the door (for example, to delivery persons) based on facial recognition.
- **Which actors** could or should be accountable for the camera?
 1. Educators of designers and the public
 2. Research ethics boards (in academia and industry)
 3. Standards bodies

4. Regulators, including Data Protection Authorities (DPAs)–acting in determining norms
 5. Designers
 6. Manufacturers of hardware
 7. Developers of software
 8. Cloud Service Providers
 9. Connectivity providers
 10. Social media platforms
 11. Installers
 12. End-users
 13. Regulators, including DPAs–acting in an enforcement capacity
 14. Courts
- **Why** are or should these actors be accountable?
 - Knowledge
 - Control
 - Proximity
 - **At what stage in the processing** do they become accountable and how can they be made more accountable and responsible?
 - *Ex ante*: before the product design phase starts
 - Production: during the design, development, installation and deployment of the product
 - *Ex post*: after the product has been deployed

These actors are considered accountable because of their knowledge of systems in operation, because of proximity to the system and (at least a certain amount of) control at a certain moment in processing.

3.2.2 Propose mechanisms to address the issue (tech-soc-reg, etc.).

Under what conditions could or should these actors be held accountable and at what stages? To identify mechanisms that could be used to address the issue who is or could or should be accountable / responsible in complex systems, we discussed a variety of different actors in the home camera use case and how they might be made more accountable:

1. The home owner or occupier could be held responsible for ensuring the camera does not cover public spaces.
2. The designer or developer could be held responsible for non-automated opening of the door.
3. The system integrator could be held responsible for operation of the system.
4. The Software as a Service (SaaS) provider could be held responsible for the cloud service (e.g., facial recognition or authentication).
5. The camera designer could be held accountable and incentivised to conduct an ethical impact assessment.
6. The retailer (e.g., a department store such as John Lewis, or even Amazon Marketplace?) could be held accountable for bringing the product to market.

- ⇒ **All individual solutions can create greater accountability of the system**
- ⇒ **No one single solution fixes the entire accountability problem**

1. **The home owner or occupier could be made responsible for ensuring the camera does not cover public spaces.**
 - *Ex ante*
 - education, e.g. domestic science teaching in schools could include how to maintain a smart home
 - Production
 - proper installation procedures could prevent the house owner from mis-installing
 - additional information about obligations and potential for misuse provided to home owner (e.g. through the set-up interface or a manual)
 - a big red sticker / smartphone interface informing users that they need to set up the device correctly (e.g., “DO NOT CONNECT THIS DEVICE TO THE INTERNET UNTIL YOU HAVE SET A NEW PASSWORD”)
 - *Ex post*
 - liability for privacy violation in a public space (cf. Ryneš case)
2. **Designer or Developer could be made responsible for non-automated opening of the door.**
 - *Ex ante*:
 - engineering training/education to think about challenges of automation
 - Production:
 - information provided in UX during the set-up scheme
 - default set to non-automation
 - *Ex post*:
 - product recall for unfixable product
 - litigation and enforcement action by regulators
3. **System integrator could be made responsible for operation of the system.**
 - *Ex ante*:
 - a clear specification that conforms to all relevant state of art standards (ethical system design, ISO, IEEE)
 - Production:
 - update the system
 - keep users informed about problems/vulnerability and proper usage
 - employ user-centric design and extensive testing of system in controlled environment
 - do not collect data that are not needed (data minimization, select while you collect)
 - obtain user feedback and update product accordingly
 - *Ex post*:
 - professional standards developed further based on reports of system integrator
 - DPO reports of system integrator shared
4. **Software as a Service provider could be made responsible for the cloud service.**
 - *Ex ante*:
 - responsible for security of supply chain
 - cloud standards certification (ISO, CSA)
 - Production:
 - authentication and access controls
 - SaaS providers monitor for security vulnerabilities and notify home owners (push for competitive advantage, market power)
 - *Ex post*:
 - enforcement by regulators and courts

- breach notification if SaaS leaks data
 - could be forced to do product recall if product cannot be updated or fixed
- 5. **Camera designer could be accountable and conduct an ethical impact assessment.**
 - *Ex ante*:
 - camera designer follows a relevant code of conduct for software developers
 - designer conducts an ethical impact assessment to assess risks and document them
 - Production:
 - check that they are following the actions that were determined in the impact assessment
 - update impact assessment based on process of development
 - *Ex post*
 - self-motivated reasons to be ethical
 - market pressure to be ethical (e.g. offering a trustworthy/accountable product as a competitive advantage)
- 6. **Point of sales person (a department store such as John Lewis, or even Amazon Marketplace?) could be accountable for bringing product into market.**
 - *Ex ante*:
 - training for staff
 - sourcing appropriate products (don't sell insecure products)
 - know your supply chain and take responsibility for vendors
 - Production:
 - follow-up information to customer to ensure ethical usage of product and provide support in doing so
 - deploy an ethical chatbot
 - *Ex post*
 - product recall if it becomes insecure and cannot be patched
 - provide lifetime warranty for product and insurance mechanism
 - provide support for customers if they have subsequent problems (e.g. unable to update device)

3.2.3 Evaluate the efficacy of the mechanisms proposed / mapping the complexity:

Potentially contentious issues include:

- Large number of actors—who can be held accountable: designer/toolmakers—people having more knowledge about bottleneck problems?
- Reasonableness of assigning responsibility to tool makers (e.g. maker of motion detector sensor that is incorporated into the camera)
- Dual/multiple use challenge
- Non-legal versus legal mechanisms (What carrots or sticks might be needed to make non-legal mechanisms effective?)
- A conflict between legal norms and ethical frameworks may arise. How paternalistic should systems be to be effective?

3.2.4 Next steps, research directions, gaps in current knowledge

Research directions:

- Are current and proposed impact assessment mechanisms (e.g. GDPR Art 35 DPIA, AINow Algorithmic Impact Assessments) suitable for use by all of the actors in the system?
- Could there be certifications associated with the technology, and what organisation would do those certifications? For example, certification assuring that all data are encrypted.
- It might be beneficial to have standard APIs to support integration of IoT devices. For example, how to control a lock. Who should define those? Can there be some certification about whether a device satisfies those interfaces?
- What makes an actor accountable, e.g. are knowledge, proximity and control the most appropriate benchmarks to determine accountability? Can accountability vary/change in the process of operation? How does this relate to the role of the data controller in the GDPR, and to legal liability?

References

- 1 Opinion 3/2010 on the principle of accountability, Data Protection Working Party Article 29 (adopted on 13 July 2010)
- 2 Opinion 1/2010 on the concepts of “controller” and “processor”, Data Protection Working Party Article 29 (adopted on 16 February 2010)
- 3 A. Crabtree, T. Lodge, J. Colley, C. Greenhalgh, K. Glover, H. Haddadi, Y. Amar, R. Mortier, J. Moore, L. Wang, P. Yadav, J. Zhao, A. Browns, L. Urquhart and D. McAuley. Building Accountability into the Internet of Things: The IoT Data Box Model, in: *Journal of Reliable Intelligent Environments* (2018) 4, p. 39–55. See <https://link.springer.com/content/pdf/10.1007%2Fs40860-018-0054-5.pdf>.
- 4 R.H. Weber, Accountability in the Internet of Things, in: *Computer Law and Security Review* (2011), Volume 27, Issue 2, April 2011, Pages 133–138. <https://doi.org/10.1016/j.clsr.2011.01.005>.
- 5 L. Urquhart, T. Lodge and A. Crabtree, *Demonstrably doing accountability in the Internet of Things*, School of Computer Science, University of Nottingham, UK. See <https://arxiv.org/pdf/1801.07168.pdf>.
- 6 W. Kuan Hon, E. Kosta, C. Millard and D. Stefanatou, *Cloud Accountability: The Likely Impact of the Proposed EU Data Protection Regulation*, Queen Mary School of Law Legal Studies Research Paper No. 172/2014; Tilburg Law School Research Paper No 07/2014. See https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2405971
- 7 W. Kuan Hon, C. Millard and J. Singh, *Twenty Legal Considerations for Clouds of Things* (January 4, 2016). Queen Mary School of Law Legal Studies Research Paper No. 216/2016. Available at SSRN: <https://ssrn.com/abstract=2716966> or <http://dx.doi.org/10.2139/ssrn.2716966>.

3.3 Engineering Accountable Systems

Martin Henze (RWTH Aachen, DE), Virgilio Almeida (Federal University of Minas Gerais-Belo Horizonte, BR), Jean Bacon (University of Cambridge, GB), Melanie Herschel (Universität Stuttgart, DE), Maximilian Ott (CSIRO – Alexandria, AU), Frank Pallas (TU Berlin, DE), Thomas Pasquier (University of Cambridge, GB), Silvia Puglisi (The Tor Project – Seattle, US), Margo Seltzer (Harvard University – Cambridge, US), Michael Winikoff (University of Otago, NZ), and Martina Zitterbart (KIT – Karlsruher Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license

© Martin Henze, Virgilio Almeida, Jean Bacon, Melanie Herschel, Maximilian Ott, Frank Pallas, Thomas Pasquier, Silvia Puglisi, Margo Seltzer, Michael Winikoff, and Martina Zitterbart

Also from an engineering perspective, accountable systems raise certain challenges and research questions. In particular, we feel that engineers who should implement accountable systems require support in doing so. The more engineering-focused aspects of accountable systems were discussed in two consecutive working groups which are jointly reported on here. In both groups, the discussions on how to engineer (complex) accountable systems were guided by the question of what are the best practices, design guidelines, implementation guidelines, operational guidelines, and re-usable technical primitives required to provide transparency of operation, allow for verification against a specification, and assign responsibility for actions. We subsume respective research and development activities under the term “Accountability Engineering”.

In particular, we foresee the engineering of accountable systems to become relevant in settings comprising multiple actors and stakeholders with “accountability chains” having to be established across system, organizational, domain, and even regulatory boundaries. For instance, we envision scenarios from the field of logistics, where multiple companies are involved in the distribution of goods—employing technologies like autonomous and connected vehicles—and where responsibility must be assigned to the correct party as soon as shipped goods turn out to have been damaged (broken, not sufficiently cooled, ...) somewhere during the transportation or supply chain.

When discussing accountability in such scenarios, it is important to keep in mind that there are different perspectives and hence different motivations to cater for accountability. Motivations for striving towards accountability vary significantly—ranging from regulatory compliance to business-driven risk-benefit weighings (e.g., when deciding about whether to use a certain service or not [1]) and ethical considerations. Independently from this, however, an area that we believe has not received appropriate consideration so far is considering accountability as a higher level service or system property/quality for which one can charge extra or which can be used to differentiate from competitors in competitive markets.

As a foundation for realizing accountable systems using Accountability Engineering, we require a common understanding of “accountable systems” specifically for engineers. Here, we particularly distinguish between three different aspects of accountability which must be reflected in Accountability Engineering:

- **Transparency:** Accountable systems must provide evidence about relevant facts concerning involved systems [2, 3] as well as external events and incidents. Systems themselves as well as provided evidence must be auditable and must facilitate justifiability to multiple parties (and not only to the one party operating a system).
- **Verifiability:** Beyond mere transparency, accountability also requires that stakeholders are able to verify aspects such as the compliance with regulatory givens, the conformance

with agreements, etc. Accountable systems must reflect this need in the collection and provision of evidence.

- Responsibility: Finally, the concept of accountability is closely related to responsibility [4]. Aspects such as assigning ‘blame’? to the party actually responsible for a damage or the enforcement of reparations must be technically supported by accountable systems. Last but not least, this also includes the need for being able to ‘fix’ systems after a damage or malfunction has actually occurred.

To appropriately address all these aspects, Accountability Engineering requires a broad, multidisciplinary understanding that combines knowledge from domains such as technology, business, law, and others. In many cases, this understanding will also have to cover multiple contextual environments (e.g., when an international delivery or supply chain comprises multiple legal/regulatory regimes). As a further precondition for making Accountability Engineering successful, it is important that engineers understand that accountability is an integral part of a product or service specification and not an add-on that can be (easily) added at a later point of time, possibly even at an extra charge.

On this basis, we envision the field of Accountability Engineering to be concerned with important research questions such as

- What information to provide to an engineering team that is about to embark on an accountable systems project?
- What to teach students about how to build accountable systems?
- What regulations are in place that have implications for the system that is to be built?
- How to incorporate regulatory changes into the life-cycle of a product or service?
- How to manage different context-specific regulatory regimes?
- How to consider ethics when designing, implementing, and operating accountable systems?

Even though considerably related, Accountability Engineering will differ from traditional software or systems engineering in multiple respects. In particular, it will be shaped by a strong focus on multidisciplinary stakeholder analysis (calling for approaches such as *i** [5] or GRL [6], which capture stakeholders, their goals, and their dependencies on each other, and on the system-to-be).

Furthermore, the collection, flow, and use of evidence data as well as questions around proving the trustworthiness and provenance of such data will play a dominant role [7]. Different from other fields, the disclosure of such data across organizational boundaries will foreseeably be essential in the context of Accountability Engineering. This will lead to many challenges regarding organizations’ unwillingness to share internal data with business partners or—e.g., in case of a dispute going to court—with third parties [1]. However, achieving accountability and, thus, desirable overall outcomes, necessarily requires some evidence information to be revealed in a trustworthy and non-disputable manner. Especially for contexts involving multiple cooperating parties, there will also be a need for technically implemented ‘evidence chains’ (e.g. provenance record [8], information flow audit [9]) that make respective data available to multiple stakeholders in different granularities—e.g., allowing the final recipient of a perishable good to access all temperature measurements that indicate conditions that it was exposed to during transit while another party only sees values exceeding a certain, predefined temperature corridor.

Meeting such requirements in concrete technical systems will call for novel technical building blocks beyond measures already established in other contexts (such as security, for example) [10]. In particular, we consider further research on the following technical mechanisms to be highly important and valuable for the establishment of accountable systems and, thus, to play an integral role in the field of Accountability Engineering:

- Access control for different actors at different granularities: To achieve technically implemented accountability across organizational boundaries, one party (A) must be able to access (some) data collected by and possibly internal to another party (B). At the same time, it must be prevented that information about third parties (e.g., B's customers) are leaked through respective mechanisms.
- Demonstrably trustworthy data capture, storage, and aggregation: Parties will have to be able to demonstrate that they took reasonable effort in making their systems “correct”, secure, and tamper-“proof”. Noteworthy, “correctness” and “proof” are not necessarily to be understood in the rather strict meaning established in computer science. Instead, Accountability Engineering will presumably also refer to and significantly profit from the more differentiated understanding established in the legal domain.
- Verifiable (possibly distributed) ledgers at the point of interaction between different parties: Whenever accountability-related data crosses organizational boundaries in an evidence chain, there is the risk of data being manipulated in the interest of downstream parties, especially when an incident actually happened. Distributed ledger technologies are a promising approach for ensuring unalteredness of evidence data in such settings and could provide several benefits over traditional public key infrastructure schemes—especially in settings involving a multitude of potentially mutually mistrusting parties.
- Mechanisms for verifying / assessing the “truthfulness” of data in chains of evidence: Being able to assess the truthfulness of evidence data provided throughout evidence chains is essential for achieving accountable systems. Besides technologies that have long been discussed in the context of Trusted Computing, other mechanisms such as advanced plausibility checks will foreseeably also play a significant role in this regard. To be practically relevant, any such mechanism must also support different granularity layers so that necessary abstractions or data transformations (see first point above) do not completely hinder truthfulness assessments.
- Mechanisms and approaches for representing and interacting with cross-organizational chains of evidence on different levels of detail: Besides assessing truthfulness, chains of evidence must also be represented and interacted with for multiple purposes to achieve accountability. In particular, this includes identifying the causes of damage that may be recognized at later stages, demonstrating compliance with regulatory requirements or business agreements, etc. Such interactions must be possible at different levels of detail for technical and non-technical users as well as by automated means. This necessitates a broad range of technical mechanisms for querying and exploring evidence data that are particularly tailored to accountability-related problems.

On this more technology-focused level, we therefore envision Accountability Engineering to deal with additional questions, including:

- What specifically needs to change in the development process to realize accountable systems?
- What type of accountability queries will accountable systems (typically) have to answer?
- Does it suffice to capture (in machine processable form) only “base-level” facts, or are higher-level facts such as commitments between parties, contracts, etc. also necessary?
- What impact can be expected from the existence of autonomous systems? Do they require paradigmatically different information to be captured and provided? Is there a need for “explanation” of “decisions” autonomously being made by such systems?
- Assuming adequate evidentiary capture, how do we support accountability queries, including verification as well as the various forms of exploration and navigation outlined above?

These and many further engineering-related questions will have to be answered on the way towards actually accountable systems. As outlined above, however, the field of Accountability Engineering is inherently transdisciplinary and cannot be mastered from the perspective of engineering (technical systems) alone. Of the many possible paths coming into consideration as next steps towards the establishment of a novel field of Accountability Engineering, we therefore want to particularly highlight the following two:

- Conduct case studies together with experienced lawyers in the context of specific scenarios, rather than abstract considerations: In the course of such case studies, accountability-related conflicts should be anticipated as going to court. Based on a set of assumptions defined to avoid catch-all “It depends” conclusions, respective disputes should then be “emulated” as running through usual legal processes and procedures. On this basis, it should then be explored how specific technologies could change things to the better or even to the worse.
- Establish a research community exploring the increasingly relevant field of Accountability Engineering from different angles, including—at least—technical primitives, specific implications for engineering processes, legal requirements and implications, and educational aspects: In the light of past experiences with transdisciplinary research on novel technologies, we foresee a multitude of challenges to be overcome in this regard: Fostering participants’ willingness to actually accept and get into the possible contributions of “other” disciplines; avoiding the assumption of unchangeable or unrealistically overestimated givens and requirements from the “own” discipline; counteracting ever-repeating paths of argumentation without progress; establishing a critical mass of community members sufficiently experienced in multiple domains and so forth. All these challenges notwithstanding, fostering transdisciplinary activities is indispensable for paving the way towards accountable systems that actually matter in practice.

References

- 1 Frank Pallas. *An Agency Perspective to Cloud Computing*. In: 2014 International Conference on Grid Economics and Business Models (GECON), (pp. 36–51). Springer, 2014
- 2 Joshua A. Kroll, Joanna Huey, Solon Barocas, Edward W. Felten, Joel R. Reidenberg, David G. Robinson, and Harlan Yu. *Accountable Algorithms*. 165 U. Pa. L. Rev. 633 (2016–2017)
- 3 Martin Henze, Daniel Kerpen, Jens Hiller, Michael Eggert, David Hellmanns, Erik Mühmer, Oussama Renuli, Henning Maier, Christian Stübke, Roger Häußling, and Klaus Wehrle. *Towards Transparent Information on Individual Cloud Service Usage*. In 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), (pp. 366–370). IEEE, 2016.
- 4 Derick W. Brinkerhoff. *Accountability and health systems: toward conceptual clarity and policy relevance*. Health policy and planning 19.6 (2004): 371–379.
- 5 *Early requirements: i** [http://istar.rwth-aachen.de/tiki-view_articles.php]
- 6 *GRL* [<http://jucmnav.softwareengineering.ca/foswiki/UCM/DraftZ151Standard>]
- 7 Andreas Haeberlen. *A Case for the Accountable Cloud*. ACM SIGOPS Operating Systems Review, 44(2):52–57, 2010.
- 8 Thomas Pasquier, Jatinder Singh, Julia Powles, David Eysers, Margo Seltzer, and Jean Bacon. *Data provenance to audit compliance with privacy policy in the Internet of Things*. Personal and Ubiquitous Computing 22, no. 2 (2018): 333–344.
- 9 Thomas Pasquier, Jatinder Singh, Jean Bacon and David Eysers. *Information flow audit for PaaS clouds*. In IEEE International Conference on Cloud Engineering (IC2E), pp. 42–51, 2016.

- 10 Denis Butin, Marcos Chicote, and Daniel Le Métayer. *Strong Accountability: Beyond Vague Promises*. Reloading Data Protection: Multidisciplinary Insights and Contemporary Challenges, Springer, pp.343–369, 2014.

3.4 Provenance for Accountable Systems

Lilian Edwards (The University of Strathclyde – Glasgow, GB), Melanie Herschel (Universität Stuttgart, DE), Bertram Ludäscher (University of Illinois at Urbana-Champaign, US), Ken Moody (University of Cambridge, GB), Thomas Pasquier (University of Cambridge, GB), and Jatinder Singh (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license

© Lilian Edwards, Melanie Herschel, Bertram Ludäscher, Ken Moody, Thomas Pasquier, and Jatinder Singh

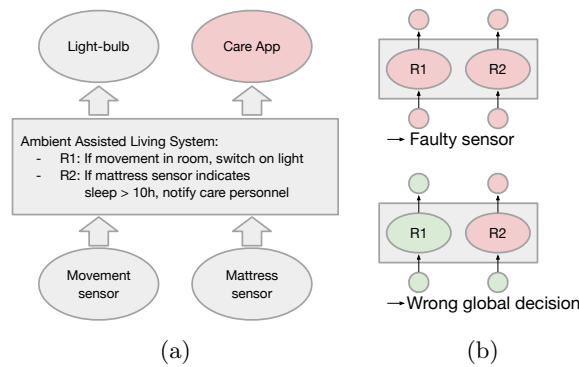
Provenance generally refers to metadata that describes the production process of some end product. The W3C PROV standard [3] defines provenance as “a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing.” [2, 1]

In the context of the accountability of systems that involve information technology, we are particularly interested in the provenance of outcomes of (partly) digital processing, e.g., decisions based on machine learning, *ad hoc* interactions in smart environments, output data of big data processing pipelines, or actuations in cyber-physical systems. This provenance includes for instance the processing history, starting from input data via intermediate results, all the way to the final outcome. Indeed, provenance may offer a useful source of audit and ‘evidence’ that has the potential to associate different actors of a process to their actions, indicate and possibly verify alignment of the actual processing with expectations, and assist compliance with legal obligations [4, 6]. More generally, provenance may support us in achieving transparency, verifiability, and indicating those responsible (or at least, where further investigation or explanations are due), which are three important dimensions of accountability. Provenance may be important to establishing strict (i.e., non-fault-based) legal liability: see the EC Product Liability Directive article 6 which states directly that “a product is defective when it does not provide the safety which a person is entitled to expect” and that this depends in part on “the use to which it could reasonably be expected that the product would be put” and the time when the product was put into circulation—both of these may depend on recording provenance. Based on an illustrative use-case, we argue that provenance plays a critical role in raising levels of accountability in systems. However, technical and legal challenges remain, and the interplays between the technical and legal aspects require further consideration.

In the following, we focus on the technical challenges, discussed based on an example use case.

3.4.1 Use Case: Ambient Assisted Living

As an example illustrating the potential benefits of provenance for accountable systems, let us consider a smart-home environment targeted towards ambient assisted living. This environment aims at supporting elderly people or people with disabilities in maintaining their independence by living at home as long as possible. Through the analysis of data collected by sensors, video, and context information provided for example through a medical history, and



■ **Figure 1** Examples of provenance in a sample ambient assisted living use case.

artificial intelligence, supportive actions may be automated. For instance, switching off the stove when nothing is cooked, setting off an alarm when a person remains suspiciously long in bed, switching on dimmed light when movement is noticed on the way to the bathroom at night, sending notifications to care givers, coordinating medical care, etc. Essentially, as depicted in Figure 1, data is collected and transmitted to the analysis system, which then triggers actions on various devices (e.g., the stove, the light bulb, or a smartphone app receiving notifications).

Given this scenario, imagine that at some point, a family member notices that no notification was sent, even though the person was spending too much time in bed. How can this system behavior be accounted for? Clearly, there are many possible causes to this (faulty) system behavior: sensors may be faulty, the data from sensors was not processed, contradictory data was processed leading to an uncaught exception in processing, the analysis system failed, etc. With the availability of proper provenance traces that track any data processing in the ambient assisted living environment, we can actually narrow down the causes for the system behavior. Different provenance scenarios are depicted in Figure 2, where green coloring indicates that data was traced in a component and red indicates that no data was associated with a component. In the upper figure, we see that no data is available at the sensor level, clearly indicating a faulty sensor. The lower figure shows that sensors produced data that was processed but then translated to another (possibly wrong action). This indicates that the decision model made a wrong decision.

Given the illustrative example above, we see that provenance indeed is important meta-data for accountability. But how to design systems that collect such provenance? In the following, we propose three system architectures that may be used to that effect.

3.4.2 Architectures for Provenance Capture

To model systems such as the one described above, we consider sensors and smart devices, the decision making system, and the vendor of the smart-home solution. A first solution to capture provenance would be a closed system, where the (proprietary) devices communicate directly with the vendor's cloud where the decision making, and the provenance capture, is based. Clearly, this offers no or very little visibility of the data and provenance, and raises privacy and security concerns. A second architecture involves a personal hub located in the home network that mediates between the smart-devices and the vendor cloud and that performs the decision making, thus putting the computing closer to the edge [5]. This way, only information that is relevant to the vendor services may be communicated, improving the

privacy compared to the first solution. Finally, a third option is to extend the capabilities of smart devices themselves, leveraging their increasing computing capabilities to perform computations right at the source and support provenance capture and communication. The three architectures have different characteristics concerning privacy, transparency, control of the data, disclosing different information available to capture provenance.

3.4.3 (Provenance) Data Disclosure

More generally, the information available to generate provenance data will be dependent on the will and incentives for parties to collaborate in transparently disclosing their actions. An ideal situation would be fully transparent parties willing to disclose any details of data processing or decision making. At the opposite end of the spectrum are uncollaborative parties refusing to participate in the generation of provenance data. In such a scenario provenance can only be inferred from the observation of systems events (or records thereof) that are within reach of the users (e.g., recording network activity). Neither end of the spectrum is desirable: total transparency presents security, privacy and business risk, while a total lack of transparency may reflect badly on the concerned party and a certain level of transparency is legally required. It remains to be determined what level of detail is required to permit the identification of a problem's root cause, while limiting risk where privacy, security and competitive advantage are concerned. We may even go further so as to state that one may want fine-grained provenance for internal root cause identification, while disclosing a more abstract representation to other parties.

3.4.4 Summary and Challenges

The above considerations demonstrate that provenance is valuable information for accountable systems, e.g., as a source of evidence that can assist in indicating responsibility, or help in debugging and fixing a system. However, various challenges lie ahead on the path to practically collecting and using provenance in accountable systems. Interesting research questions include: Can provenance be privacy preserving? How can we meet requirements for accountable systems without disclosing additional information? How can we guarantee the availability of the provenance data? What are the legal requirements regarding the management of provenance data? Similarly, how can we ensure the integrity of collected provenance? Also, to capture and use provenance in a complex system involving various parties, how can we support interoperability (at all levels: system, syntactic, semantic)?

References

- 1 Lucian Carata, Sherif Akoush, Nikilesh Balakrishnan, Thomas Bytheway, Ripduman Sohan, Margo Seltzer, and Andy Hopper. A Primer on Provenance. *Communications of the ACM*, 57 (5): 52–60 (2014)
- 2 Melanie Herschel, Ralf Diestelkämper, and Housseem Ben Lahmar. A survey on provenance: What for? What form? What from?. *The VLDB Journal*, 26 (6): 991–906 (2017)
- 3 Luc Moreau and Paolo Missier, eds. PROV-DM: The PROV Data Model. *W3C Recommendation*. <https://www.w3.org/TR/prov-dm>, 2013
- 4 Thomas Pasquier, Jatinder Singh, Julia Powles, David Eyers, Margo Seltzer, and Jean Bacon. Data provenance to audit compliance with privacy policy in the Internet of Things. *Personal and Ubiquitous Computing*, 22 (2): 333–344 (2018)
- 5 Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5): 637–646 (2016)

- 6 Jatinder Singh, Jennifer Cobbe, and Chris Norval. Decision Provenance: Capturing data flow for accountable systems. *arXiv preprint arXiv:1804.05741*, <https://arxiv.org/abs/1804.05741>, (2018)

3.5 Anonymity, Identity and Accountability

Jean Bacon (University of Cambridge, GB), Heleen Louise Janssen (Ministry of the Interior and Kingdom Relations, NL), Joshua A. Kroll (University of California – Berkeley, US), Silvia Puglisi (The Tor Project – Seattle, US), Jatinder Singh (University of Cambridge, GB), and Martina Zitterbart (KIT – Karlsruher Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license

© Jean Bacon, Heleen Louise Janssen, Joshua A. Kroll, Silvia Puglisi, Jatinder Singh, and Martina Zitterbart

Group questions: *Does accountability necessitate the identification and/or visibility of the actors involved? and In what contexts can a violation of privacy be justified to achieve accountability?*

3.5.1 Considerations about the relation between accountability and identity

Identity has a clear relation to accountability. Identity is important as a means for determining those to hold to account, which can be challenging in a complex system. Does accountability necessarily mean that it is associated with an identifiable person? Identification also poses considerations with regards to the ‘users’ of a system. Does an individual need to be identifiable—explicitly, or through a pseudonym? Or can (or should?) systems be built in a way so that the details of the individual’s identity are not required while still being accountable? And how do the design decisions relating to identity impact on the levels of accountability that systems help to support. Ultimately, ‘it depends’, issues of identity and anonymity relate to the particular context.

Considerations on accountability: If a system is to support accountability, it needs to record sufficient data to identify relevant events and actors as required by its specification. In a system-of-systems, this accountability data may be partitioned according to the system components and their functionalities. We assume that access to audit data is controlled so that only authorised parties can see it. We assume that a case has been made and authorised for the data to be gathered for legitimate purposes. In this section we assume, at least initially, that the audit data are correct.

3.5.2 Use case: check-in/check-out in public transportation systems

The group identified a use case in which the issues that come with accountability, anonymity and identity can be concretely explored. To investigate options available and trade-offs that might be made, it considered an urban transport system such as the Paris Metro or the Boston T. The group supposed that the initial system design allows people to be completely anonymous, for example, if they choose to use cash to buy tokens. A token will buy any length of journey at any time, and deposited on entry—no other information is available. Suppose such a system is due to be upgraded to meet the following requirements:

- Start and end points of journeys are to be recorded to aid resource planning.
- Different charging models for different journey lengths, different times of day, days of the week, etc.

Consequences of these changes (as well as aiding modelling and planning, as above) are that:

- Individual actions can be inferred—their home and work locations and travel habits.
- Police can be assisted in identifying who could have been present when some crime took place at some location.

If a system is to support accountability it needs to record sufficient data to identify relevant events and actors as required by its accountability specification. In complex system environments this accountability data may be partitioned according to the system components and their functionalities. We assume that access to audit data is controlled so that only authorised parties can see it. We assume that a case has been made and authorised for the data to be gathered for legitimate purposes. In this section we assume, at least initially, that the audit data is correct.

Tensions between accountability on one hand and privacy on the other may arise in these systems, as well as the question of how these tensions can be managed. Specifically *how can accountability in check-in/check-out in city public transportation systems be managed?* Three main systems were considered:

- Token-systems (system as used in the Paris metro). With a one-off deposit of a generic token, people travel unidentifiably.
- Card based systems (London Oyster Card system or credit card system). People can be identified with credit cards and whenever the London Oyster Card is connected to an identifiable person.
- Biometric system (e.g. a future airport with seamless flow systems), where people are identifiable, e.g. by cameras, device traces, other sensors, etc. This use case was not discussed in detail; the group took note of its emergence in the near future.

3.5.3 Drivers for the use of check-in&out systems

Transport related: The group discussed which specific drivers for ‘change’ exist –e.g. for taking the step away from token-based systems to those entailing check-in/check-out. Taking the Paris Metro with anonymous paper tickets as a starting example, the group noted that original purpose—solely securing that travellers pay for their journey—was less amenable towards city goals of better transport planning and management. As such, many public transport systems use check-in/check-out ticketing approaches: journey tracking provides for insight in understanding travelling patterns, potentially leading to saving money and better allocation of resources (e.g. by offering different services during rush hours for higher prices). The question arose to what extent a traveller’s identity is required for such insights. The group questioned whether the use of identities is necessary for transport planning. There may be different levels of aggregation regarding the identities used, or different methods for tracking flow of passengers once travelling, potentially leading to less of an impact on privacy/anonymity. For the detection of fraud by the transport organisation, knowledge of the identity of the travellers *might* assist—depending on the specifics of its implementation.

Not transport related: For crime prevention, detection and prosecution, knowledge of someone’s identity may be helpful as well, but it is not clear to what extent an identifiable ticket necessarily assists investigation. Generally, data mining and analytics, to give more information of the flow of people throughout the city (beyond transport purposes) was also identified as a driver.

Other drivers for check-in&check-out are that alternative forms of ticketing can be used, and services can be built on top. For instance, using personal credit cards for ticketing could (by tracking travellers), mean additional services may be offered, such as (targeted) services

such as advertisements for trips, experiences, food, etc. In order to realise these customised services, some degree of traveller identity will be required. At present, it appears that privacy and anonymity are being eroded due to commercial and financial incentives.

The group concluded that any next step seems at least to result in increased tracking possibilities and therefore a greater impact on privacy and anonymity.

Typical issues: The group identified typical issues and concerns coming with the use of identifiable check-in/check-out in city public transportation systems:

- All methods can bring statistics
- Public acceptability (agency, personal autonomy, self-determination)
- Identifiability/re-identifiability
- Simplified analytics
- Facilitated ‘function creep’
- Discouraging bad-behaviour (gaming the system)
- Anonymous travel will become less self-evident
- Unclear what more exactly happens to the data
- (Prior) design decisions (and constraints) are important
- Fundamental change of a system once in operation is very expensive

3.5.4 Mechanisms to address the tension between accountability and privacy in transportation systems

The main question considered was: *Does accountability necessitate for identification and/or visibility of travellers for the actors?* The group discussed the token and card-types of check-in and check-out systems in public transportation, and how they could be made accountable while respecting the privacy (anonymity and identity) of the individual. The group identified various types of cards, including:

- Fully-identified fare cards
 - Contactless credit card payments, ID cards (based on biometrics)
- Pseudonymous fare cards
 - e.g. Oyster cards
- Fully anonymous electronic fare cards
 - anonymous card paid by e-cash/Zcash
 - payments, trustworthiness of anonymity

A key takeaway was that while all solutions bring statistics, no one solution deals with all accountability and privacy concerns. In short, *there is a trade-off*. Potential barriers to improvements and change in city public transportation systems include:

- Fundamental change of a system once in operation is very expensive
- The increasing number of actors
- Challenges of dual/multiple use (i.e. data and services beyond transport)

3.5.5 General takeaway: context is key

The group concluded that context is key, highlighting some general questions and considerations regarding the identity–anonymity–accountability tradeoffs:

- Determine the actors involved in the processing of the personal data
- What values are in play? How much protection should each value deserve?
- Discrimination (e.g. when means are unreasonably allocated to certain zones based on data, or when certain passengers are unreasonably targeted/checked)
- Agency/personal autonomy of the individual

- Broader benefits of individual-oriented analytics
- The necessity of individual-oriented analytics, are aggregates as effective?
- Security implications, both technical and societal
- Preventing bad behaviour, defining the threat model
- What is necessary to record (considering each actor, each purpose)?
- How to resolve conflicts regarding data retention and data protection?
- How can technology contribute to support oversight practices while respecting privacy/anonymity?
- ID-management: what are the technical, management and ‘user’ considerations
- Reliability and accuracy of identification mechanisms

References

- 1 Pfitzmann, Andreas, and Marit Hansen. “A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management.” (2010).
- 2 Branscomb, Anne Wells. “Anonymity, autonomy, and accountability: Challenges to the first amendment in cyberspaces.” *The Yale Law Journal* 104.7 (1995): 1639–1679.
- 3 Christopherson, Kimberly M. “The positive and negative implications of anonymity in Internet social interactions: “On the Internet, Nobody Knows You’re a Dog”.” *Computers in Human Behavior* 23.6 (2007): 3038–3056.

3.6 Post-Consent

Jennifer Cobbe (University of Cambridge, GB), Martin Henze (RWTH Aachen, DE), Bertram Ludäscher (University of Illinois at Urbana-Champaign, US), Ken Moody (University of Cambridge, GB), Maximilian Ott (CSIRO – Alexandria, AU), and Margo Seltzer (Harvard University – Cambridge, US)

License © Creative Commons BY 3.0 Unported license
© Jennifer Cobbe, Martin Henze, Bertram Ludäscher, Ken Moody, Maximilian Ott, and Margo Seltzer

The group agreed that there seems to be a trend in academia, industry, and government to frame issues around the appropriate use of personal data in terms of consent, often drawing on experiences of the medical domain. However, the digital domain has characteristics which make it quite different from the medical domain; sufficiently so that it seems prudent to question the applicability of consent as a basis for processing personal data. Indeed, a significant body of research indicates that relying on consent for the processing of personal data is flawed in many ways [8, 7, 10, 11, 2, 9, 6, 4].

Unlike in the medical domain, users in the digital realm are subjected to a constant onslaught of consent requests from web sites and apps, each with their own privacy policy, as opposed to the comparatively small number of such documents presented by a trusted medical professional. Privacy notices are often very long, confusing, legalistic, and use euphemisms to describe the processing in question (e.g. describing the sale of user data as “sharing”). Given the length, complexity, and number of privacy policies it is unlikely indeed that anybody reads all of them. Indeed, a 2008 study concluded that it would take ten full days to read all the privacy policies with which users are confronted each year [5] (that number presumably having grown greatly in the last decade). Not only is it clear from the literature that people usually do not read privacy policies and often do not understand what they are consenting to, but research has also shown that whether users give consent is heavily dependent on

contextual clues [1] (giving data controllers significant influence over whether data subjects give consent to processing). The use of manipulative or deceptive practices in gaining the consent of data subjects is well-documented.¹ Finally, a problem which arises in the digital domain which does not necessarily arise in the medical domain (unless machine learning tools are being used), is that personal data are frequently used to build predictive models, the outputs of which are unforeseeable for data subjects and which may cause their own kind of privacy harms [3]. Given this, it is not clear how one could adequately inform and seek consent from data subjects for such processing.

As a result of these various issues, the group felt that, in order to develop accountable systems of the future, better approaches to providing a lawful basis for processing personal data which respects and protects the rights to privacy and data protection of data subjects is needed. Identifying potential legal bases which provide meaningful control for data subjects as well as developing technical means to protect personal data but which do not rely on consent were thought to be key areas for further research.

References

- 1 Acquisti, A., Brandimarte, L. and Lowenstein, G., 2015, *Privacy and human behaviour in the age of information*, Science, 347(6221), 30 January, pp.509–514
- 2 Brandimarte, L., Acquisti, A., Loewenstein, G., 2010, *Misplaced confidences: privacy and the control paradox*, Ninth Annual Workshop on the Economics of Information Security
- 3 Crawford, K. and Schultz, M., 2013, *Big Data and Due Process: Toward a Framework to Redress Predictive Privacy Harms*, New York University Public Law and Legal Theory Working Papers, Paper 429.
- 4 Doteveryone, 2018, *People, Power and Technology: The 2018 Digital Understanding Report*
- 5 McDonald, A. M. and Cranor, L. F., 2008, *The Cost of Reading Privacy Policies*, I/S: A Journal of Law and Policy for the Information Society, 4(3), pp.541–565
- 6 Mantalero, A., 2015, *The future of consumer data protection in the E.U. Rethinking the “notice and consent” paradigm in the new era of predictive analytics*, CLSR, Vol. 30, pp.643–660
- 7 Moores, T., 2005, *Do consumers understand the role of privacy seals in e-commerce?*, Communication of the ACM, 48(3), pp.86–91
- 8 Privacy Leadership Initiative, 2001, *Privacy Notices Research Final Results*, Harris Interactive, 2001
- 9 Solove, D., 2013, *Privacy Self-Management and the Consent Dilemma*, 126 Harvard Law Review, p.1880
- 10 TRUSTe, 2006, *Consumers have a false sense of security about online privacy—Actions inconsistent with Attitudes*, TNS, 2006
- 11 Turow, J., Hoofnagle, C.J., Mulligan, D.K., Good, N., 2007, *The federal trade commission and consumer privacy in the coming decade*, ISJLP, No. 3, pp.723–749

¹ See, for example, <https://darkpatterns.org/>

3.7 Automating Data Rights

Michael Veale (University College London, GB), Lilian Edwards (The University of Strathclyde – Glasgow, GB), David Evers (University of Otago, NZ), Tristan Henderson (University of St Andrews, GB), Christopher Millard (Queen Mary University of London, GB), and Barbara Staudt Lerner (Mount Holyoke College – South Hadley, US)

License © Creative Commons BY 3.0 Unported license

© Michael Veale, Lilian Edwards, David Evers, Tristan Henderson, Christopher Millard, and Barbara Staudt Lerner

This working group examined how individual rights in the law might, through computationally-mediated means, operate automatically and at scale, and the effect this might have on accountable data-driven systems more generally. The report that follows should be read as as a discussion piece rather than a finalised piece of work.

3.7.1 Discussed Problems

Today, a wide array of entities hold, transform and share personal data. The networks established through these processes are complex, and present a major barrier to accountable systems [4, 14]. Not only do individuals rarely know what data are held by organisations and the inferences made about them, but oversight bodies and regulators often lack the expertise to assess processing practices. As a result, it is likely that many of these actors are operating outside both the letter of the law and are violating consumers’ reasonable expectations. This is likely to continue to be the case, even notwithstanding the higher stakes under the General Data Protection Regulation (GDPR)² of fines of €20m / up to 4% of global turnover.

The core problem is how and whether, without significant legal change, the existing provisions might better serve consumers in fostering accountability within these systems.

3.7.2 Augmenting Individual Rights

Data protection rights and obligations are powerful in theory, providing building blocks for control of a vast array of data types and processing practices relating to an individual. Many powerful rights are present in current European data protection law. The right to access data that relate to an individual, and the right to “portability” of a narrower range of personal data, both enable a data subject to take copies of data and use them for their own purposes. The right to erasure, often referred to as the “right to be forgotten”, is a qualified right to ask a controller to delete or obscure data (for example, by delisting specific search engine results). There are a variety of ways that a data subject can prevent a data controller from processing their data for particular purposes, such as by withdrawing consent or using their rights to object to or restrict processing. Individuals are also entitled to a range of metadata, including retention time, data source, purposes of processing, and meaningful logic about certain types of automated decision-making. These ‘building blocks’ might be used individually or in combination to facilitate enhanced transparency and control.

Many of these rights have existed in some form and in some jurisdictions since the 1970s [10] and became more widespread (both within Europe and outside) since the adoption

² Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1.

of laws based on the EU 1995 Data Protection Directive (DPD).³ Despite their presence on the statute books of more and more countries, the rights, and associated obligations, have been subject to very little judicial scrutiny and there is considerable uncertainty about how the rules should be applied in specific circumstances, particularly concerning fast-moving technologies.

Furthermore, while theoretically strong, they tend to be burdensome for data subjects to use in practice, and are often ineffective as a result [7]. Many of these networked challenges are difficult to grapple with from an individual point of view, as even understanding the kinds of data out there in the world, the kinds of business relationships that exist, and the kinds of rights that are available, is challenging.

One major change in the GDPR from the DPD is the ability to submit requests electronically, and receive responses, including copies of the data, in commonly-used and/or machine-readable formats. This is a significant change from the *status quo*, where companies, even those with almost no offline presence for consumers, often attempted to force individuals to write a physical letter to their European headquarters, including a fee for the request (up to £10 in the UK), often requested through a cumbersome payment method such as a postal order rather than an online payment. These approaches, which may be employed to deter the exercise of individual rights, have now been more-or-less scrapped, meaning that requests can be made instantly and without cost.

In this working group, we discussed the possibilities and utility of electronic use of these rights, potentially through semi-automated means *de facto* delegated to a third-party. Delegating rights in this way might come with risks, such as to privacy, but also benefits, such as avoiding individualising the challenge of accountability. Furthermore, deploying emerging technologies may help balance these trade-offs. Some potential impacts and challenges are outlined below.

3.7.2.1 Information rights to understand provenance

Various information provisions in the GDPR allow a requester to learn where data came from (backward provenance), and to whom the data have been, or may be, transferred (forward provenance).

Article 13 specifies information that should be provided to the data subject at the point of collection. Article 13(1)(e) states that, when data are collected from a data subject, the data subject should be provided with forward provenance information: ‘the recipients or categories of recipients of the personal data, if any’.⁴ Ideally, the information provided here would be sufficient to allow a data subject to query the recipients of these data, and further expand their enquiries. Unfortunately, the proposed text was contentious, and ultimately retained language from the DPD that the data subject might also be provided with the *categories* of recipients rather than specific named recipients. European regulators, in their guidance on the topic, have however emphasised that the default should be *named* recipients, and if a data controller wishes to use categories of recipients, they should justify why they believe this to be fair, and provide sufficiently detailed descriptions including, for example, a breakdown by geographical location and sub-sector [2].

³ Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, OJ 1995 L 281/31.

⁴ An individual can also obtain this information on request using Article 15(1)(c).

Learning where data came from (backward provenance) however has considerably less ambiguity compared to learning where it was going. Where data are not directly collected from a data subject, Article 14 applies, requiring similar information to be provided to the data subject as under Article 13. Where the individual is difficult to communicate with without disproportionate effort (for example, if re-identifiable but not identified data are transferred), such information might be published more generally, such as on a website. Article 14(2)(f) provides a backward provenance provision that states that ‘the controller shall provide the data subject with the following information necessary to ensure fair and transparent processing in respect of the data subject [...] from which source the personal data originate, and if applicable, whether it came from publicly accessible sources’.⁵ The *source* of the data is not qualified to give a data controller the option to substitute more general information, such as categories. As a result, the GDPR appears to be stronger in its provision for backward provenance than it is for forward provenance.

For the purposes of accountability, backward provenance has unfortunate downsides compared to forward provenance, as the data subject is more likely to be able to identify the data controllers that have collected data from them directly compared to those that have been passed data, potentially through long chains of interactions. Nevertheless, the GDPR provides for some transparency regarding both types of data collection, and automated requests can therefore be attempted automatically in relation to data provided both directly and indirectly.

If regulators and courts take this direction, Article 14 appears to oblige many firms to publish detailed information on where data came from. In that case, systems designed to scrape websites and privacy policies might be able to help trace back provenance trails and establish further oversight around the movement of personal data. Failing this, individual requests will likely remain necessary. The next section considers some of the socio-technical challenges associated with making such requests in automated and semi-automated ways.

3.7.3 Challenges

3.7.3.1 Identification and verification

A key challenge in all of this is being able to connect data accurately with an individual. In some databases, an individual might be easily recognised via an email address, phone number, or other obviously recognisable identifier. In other cases, an individual’s identity may be inferred from information less obvious to the individual, such as an ID associated with a phone or other tracking device.

When logging into an online service, a user may be able to choose from among several IDs, such as a site-specific ID, a government ID, a Google ID, or a Facebook ID. Are these all equivalent, or is the user exposed to different risks depending on which identity is chosen? Is the identity service, like Google, used solely to authenticate access, or do other data leak between the identity service and the service being logged into? Can the average user be expected to know or understand these risks? It is also common to ‘single users out’ using information such as identifiers, cookies and web fingerprints which are difficult or even impossible for users to find or send, but comparatively easy for data controllers to use [5, 15].

To complicate matters further, devices and logins are often shared. A child might borrow a parent’s phone. A couple might share an online subscription. The presence of a visiting guest might result in changes to electricity usage or TV watching patterns. In many cases

⁵ Article 15 provides a similar provision upon request.

like these, it would be easy to be mistaken as to whom the data relate. As a result, there is often considerable uncertainty regarding the identity of the relevant data subject, yet this uncertainty is generally not acknowledged.

3.7.3.2 Dialogue with data controllers

Access requests are rarely smooth and simple to make. Recent research has highlighted that users face a difficult task in obtaining rights they are entitled to. Firms frequently point users to an automated data download tool when they request access to their data, even though such a tool is often non-exhaustive in terms of the categories of data it provides [3]. Similarly, requests can be confusing for those receiving them, or identification (see above) can be challenging. Smaller data controllers may simply lack the expertise or resources required to respond appropriately or fully to requests.

Such difficulties are similar to those experienced by those making freedom of information (FOI) requests to public bodies under the variety of FOI laws in place around the world. A range of tacit and codified knowledge has emerged around how to make successful FOI requests [6], and civic technology platforms such as *WhatDoTheyKnow.com* and *AskTheEU.org* have been set-up to help guide individuals through their ping-pong email exchanges with officials until they get the information they are seeking. The most successful and revealing FOI requests are frequently those in which the requester already knows a significant amount regarding the information that exists, such as the title, date or even the reference for a document [6]. This knowledge barrier has led FOI to become a tool-of-the-trade for journalists, but a right much less frequently utilised by laypersons.

Data protection rights have historically had a similar fate. While limited information is available on the use of data protection rights across sectors, given the lack of any reporting obligations for the majority of the organisations that are subject to the law, the evidence that exists points to a highly professionalised use of rights. The Law Society of England and Wales, for example, has highlighted the extensive use of subject access rights in immigration proceedings against organisations such as the UK Home Office [11]. For successful access requests in sectors with very different forms and categories of data, it seems likely that specific help to target and obtain desired outcomes will be required.

Many of these back-and-forth interactions will be undertaken via email, and there is a need for technologies to support individuals in asking the right questions, and having access to the relevant legal arguments that allow them to clarify the obligations of data controllers (who may not be aware of the extent of their obligations, such as the breadth of what is considered ‘personal data’ [12]) and to be aware when the arguments being made by data controllers may not have substantive legal backing, and could be grounds to complain to a data protection authority. Such technologies in other fields have been popular in recent years, such as the well-publicised chatbot *DoNotPay*, which helps individuals to overturn parking tickets [8]. Response prediction technologies have also entered common public use, such as Google’s *Smart Reply* feature, which predicts an appropriate short phrase on the basis of the content of a received message [9]. Yet given that data protection regulations are more complicated than parking violations (which can be managed via a relatively simple rule-based system), and that the aim is not to predict a realistic response like *Smart Reply* but to achieve a particular outcome, in many ways this issue is more challenging, and is reminiscent of work in legal expert systems.

Significant interdisciplinary research is required in this field. Such a platform to enable data subjects to utilise their rights must be legally sound and up-to-date with the latest case law, particularly as more cases are expected in the Court of Justice of the European

Union now that rights will be considerably easier to exercise. It must also be usable for data subjects; both easy to understand and providing practical assistance to them to achieve their desired outcomes. To enable this, it likely that a considerable amount of advanced natural language processing will be required.

The difficulty of developing a usable system to solve this complex problem is compounded by privacy concerns. The requests that an individual is making, and what they care about, are likely to be highly revealing for some, especially where they concern sensitive data such as health status, or data used to prove identification. Centralising these data for the purposes of improving the training of, for example, a machine learning text classifier is risky. Failing to combine such data, however, is likely to make it difficult to improve the system in question, and may make it impossible to develop a system with appropriate usability. It may be useful to draw upon privacy-preserving, federated machine learning techniques to build models and run tests privately ‘at the edge’, however deploying these technologies is also far from straightforward, and replicating all the functionality possible in centralised data systems efficiently and effectively is still a subject of heavy research.

3.7.3.3 Data cleaning

The data formats returned by the access and portability rights are likely to pose challenges to the effective automation of data protection provisions. The right of access specifies that when a request is made by electronic means, data should be returned in a commonly used, electronic form (Article 15(3)). The right of portability goes further than this to specify that data be returned in a “structured, commonly used and machine-readable format” (Article 20(1)), although the categories of data that can be requested through a portability request are limited in the text to those which a data subject has “provided to a controller” and based on consent or contract, unlike the right of access which concerns a wider “copy of the personal data undergoing processing” regardless of its lawful basis (Article 15(3)).⁶

One potential outcome is that many access requests will be fulfilled with a format such as PDF, which is notoriously difficult to extract information from reliably and without human intervention. Even seasoned data analysts find table extraction from formats such as PDF difficult, and a considerable research literature exists around how to do this reliably and repeatedly given the wide variety of ways in which data are presented [13]. Furthermore, structures of data which are readable easily by humans, such as ‘wide’ tables can be difficult to do structured analysis on with data manipulation grammar [16]. Transforming data between these formats can be a challenging task even for seasoned analysts, let alone for laypeople. As above, this data is likely to be highly sensitive, and support tools for transformation will be likely required to run locally with challenges in applying learning technologies to automatically process different controllers’ datasets due to the difficulties in amassing sanitised examples.

3.7.3.4 Children

Children present a special class of user community. What data should be collected about children? How do children exercise their subject access rights, or can parents do this on their

⁶ There is some controversy over this, as the pan-European group of regulators, the Article 29 Working Party has stated in its guidelines on portability that Article 20 refers to data that have been both “actively and knowingly provided by the data subject” and “[o]bserved data provided by the data subject by virtue of the use of the service or the device”. Whether the Court will agree with the regulators on this apparent broadening of the original text remains to be seen. See [1].

behalf? Some “competent” children may be able to make requests on their own behalf, but how is competency determined?

Consent in GDPR is country-specific. In many countries, 16 is considered the minimum age for consent but there is considerable deviation from this norm, with the age ranging as low as 13.⁷ In general, GDPR is not clear about children exercising rights, so one needs to look at non-binding recitals (e.g. recital 65) for guidance and not just the GDPR’s core text (the articles).

The text and guidance do not anticipate children exercising rights directly, and yet these are among the most vulnerable groups for whom we might want to ensure strong accountability and oversight. The ICO has emphasised that data controllers should “allow competent children to exercise their own data protection rights.”⁸ However, it is important to consider how competence is measured, both in a legal sense as well as the potential risks that might be present from giving children access to all their personal data which they might then be pressured to give away to companies or other parties.

3.7.3.5 Additional issues

Additional issues that were discussed include

- how to ensure the data requested under these rights does not become a security risk in and of itself when held locally by the users;
- how data controllers might be empowered to ensure that automated data rights do not cripple the digital economy (e.g. what sort of APIs should be promoted?);
- what kind of analysis of requested data might be undertaken with which methods, how revealing it might be and how useful it might be to a regulator;
- whether authentication methods might be possible using zero-knowledge proofs or similar approaches;
- the roles of third party intermediaries (such as technology giants) in providing ‘scalable access request management’, and how this might affect privacy and competition, and whether it might be possible to undertake using secure cloud and enclave approaches;
- standardisation efforts for data portability and access.

3.7.4 Conclusions and open questions

Many of the open questions in this field will only be dealt with by trying to build systems that attempt to automate data rights. To what extent those systems will find support in the law, such as in provisions around data protection by design, remains unclear. Furthermore, some controllers might see such systems as a threat to trade secrets and confidentiality, and may take protective measures to prevent the aggregation of release. In other sectors, standardisation might be welcomed for the purposes of economic efficiency and reduced compliance burden.

In the short term, we identified needs for

- An ongoing meta study of all the GDPR access request research that will be written after everyone starts collecting data on 25 May;

⁷ Mapping the GDPR age of consent across the EU: April 2018 update at <https://www.betterinternetforkids.eu/web/portal/practice/awareness/detail?articleId=3017751>

⁸ <https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/applications/children/>

- Estimations of the rates of chaos from email-based systems at scale to provide evidence and support for standardisation and automation from the data controllers' perspectives;
- Connection to research in personal data containers such as DataBox, openPDS, the Hub-of-All-Things, among others;
- Usability research into how individuals grapple with their own rights, what their personal goals are and what support tools they need and find useful;
- Local and sectoral differences in subject access request fulfilment and enforcement;
- The role of identity providers and verification, working with regulators to ascertain appropriate security-data rights trade-offs and to implement them technically in different contexts;
- The structure of an open source personal data management system founded on automated rights: what core technologies (e.g. privacy-enhancing technologies) are best suited, and which open research questions in those domains are a pre-requisite to some of the aims (such as analytics at scale for systemic accountability) we have here.

References

- 1 Article 29 Data Protection Working Party. *Guidelines on the right to “data portability”* (wp242rev.01). 2017.
- 2 Article 29 Data Protection Working Party. *Guidelines on Transparency under Regulation 2016/679* (wp260rev.01). 2018.
- 3 Jef Ausloos and Pierre Dewitte. Shattering one-way mirrors – data subject access rights in practice. *International Data Privacy Law*, 8(1):4–28, February 2018.
- 4 Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. Third party tracking in the mobile ecosystem. In *Proceedings of the 10th ACM Conference on Web Science*, pages 23–31. ACM, 2018.
- 5 Frederik J Zuiderveen Borgesius. Singling out people without knowing their names—behavioural targeting, pseudonymous data, and the new data protection regulation. *Computer Law & Security Review*, 32(2):256–271, 2016.
- 6 Matthew Burgess. *Freedom of Information: A Practical Guide for UK Journalists*. Routledge, June 2015.
- 7 Lilian Edwards and Michael Veale. Slave to the Algorithm? Why a ‘Right to an Explanation’ is Probably Not The Remedy You Are Looking For. *Duke Law Technol. Rev.*, 16(1):18–84, 2017.
- 8 Samuel Gibbs. Chatbot lawyer overturns 160,000 parking tickets in London and New York. *The Guardian*, June 2016.
- 9 Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 955–964, New York, NY, USA, 2016. ACM.
- 10 Christopher J Millard. *Legal protection of computer programs and data*. Sweet & Maxwell, Toronto and London, 1985.
- 11 The Law Society of England and Wales. *Written evidence from the Law Society of England and Wales to the House of Commons Public Bill Committee considering the Data Protection Bill*. 2017.
- 12 Nadezhda Purtova. The law of everything. broad concept of personal data and future of EU data protection law. *Law, Innovation and Technology*, pages 1–42, April 2018.
- 13 Roya Rastan, Hye-Young Paik, and John Shepherd. TEXUS: A task-based approach for table extraction and understanding. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 25–34. ACM, September 2015.

- 14 Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem. *arXiv preprint arXiv:1609.07190*, 2016.
- 15 Michael Veale, Reuben Binns, and Jef Ausloos. When data protection by design and data subject rights clash. *International Data Privacy Law*, 8(2):105–123, 2018.
- 16 Hadley Wickham. Tidy data. *J. Stat. Softw.*, 59(1):1–23, 2014.

4 Overview of Talks

4.1 Tutorial–Cloudy with a hint of accountability

Jon Crowcroft (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license
© Jon Crowcroft

Joint work of Haddadi Mortier et al.

Main reference Richard Mortier, Jianxin R. Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James A. Colley, Tom Lodge, Tosh Brown, Derek McAuley, Chris Greenhalgh: “Personal Data Management with the Databox: What’s Inside the Box?”, in Proc. of the 2016 ACM Workshop on Cloud-Assisted Networking, CAN@CoNEXT 2016, Irvine, California, USA, December 12, 2016, pp. 49–54, ACM, 2016.

URL <http://dx.doi.org/10.1145/3010079.3010082>

This was an introductory talk about the use of different approaches to improve the confidentiality and integrity of cloud based systems. On the one hand, in large data centers, we can use Trusted Execution Platforms running on newer hardware (both SGX on Intel CPUs and Trustzone on Arm processors) to provide much stronger assurances about the privacy, and attest to integrity of code and data processing in the cloud. On the other hand, we can use edge-cloud (or personal cloud) to run systems in the same way, but keeping data and processing in its home (smart home, car, building city, etc.) improving availability, latency, energy efficiency, and reducing the attackable footprint (“surface”) of the system. Linking this with accountability is a work for the future. Of course, systems can still shoot themselves in their feet.

4.2 Tutorial–ML : transparency, control, user rights and the GDPR

Lilian Edwards (The University of Strathclyde – Glasgow, GB)

License © Creative Commons BY 3.0 Unported license
© Lilian Edwards


Main reference Lilian Edwards and Michael Veale: “Slave to the Algorithm? Why a ‘Right to an Explanation’ Is Probably Not the Remedy You Are Looking For” 16 Duke Law & Technology Review 18 (2017)

URL <https://dx.doi.org/10.2139/ssrn.2972855>

The tutorial outlined the basic principles and structure of the General Data Protection Regulation (GDPR). The conditions for lawful processing, including, but not restricted to, consent, were outlined as were the definitions and import of the terms personal data, processing and data controller/data processor. The rest of the tutorial concentrated on the DP problems that arise out of “Big Data”, profiling and machine learning. Particular attention was paid to the so-called “right to an explanation” which might be derived from art 22 or art 15(h); and issues arising from its implementation in EU member states.

4.3 Tutorial—A Brief Introduction to Provenance in Workflows and Databases

Bertram Ludäscher (*University of Illinois at Urbana-Champaign, US*)

License  Creative Commons BY 3.0 Unported license
© Bertram Ludäscher

Computational notions of data provenance have been studied in different contexts such as databases, programming languages, and scientific workflows. While there are considerable differences in the assumptions, perspectives, and problems studied by the various communities, a common underlying theme is that of transparency and comprehensibility of the computational processes that yield a given data output. When trying to make complex systems “accountable” for data-driven, algorithmic actions and decisions, it is necessary to capture relevant provenance information, e.g., by “looking inside” of black-box computations and capturing not only retrospective provenance but also prospective provenance, i.e., the dataflow dependencies of the computations and workflows that make up the computational system. Hybrid forms of provenance, combining workflow specifications and retrospective provenance provide new opportunities to gain insights into the intended and actual workflow executions, and can be used to account for and explain system behavior. New research challenges arise from the inherent trade-off between transparency and provenance (necessary elements of accountability) on one hand, and requirements for privacy and data protection on the other.

References

- 1 Ludäscher, Bertram. A Brief Tour Through Provenance in Scientific Workflows and Databases. In *Building Trust in Information*, edited by Victoria L. Lemieux, 103–26. Springer Proceedings in Business and Economics, 2016.
- 2 Dey, Saumen C., Daniel Zinn, and Bertram Ludäscher. Propub: Towards a declarative approach for publishing customized, policy-aware provenance. In *International Conference on Scientific and Statistical Database Management (SSDBM)*, pp. 225–243. Springer, Berlin, Heidelberg, 2011.

4.4 Tutorial—Ethical/social: Rights, Ethics and Accountability

Ben Wagner (*Wirtschaftsuniversität Wien, AT*)

License  Creative Commons BY 3.0 Unported license
© Ben Wagner

Main reference Ben Wagner, “Algorithms and Human Rights: Study on the human rights dimensions of automated data processing techniques and possible regulatory implications” (No. DGI(2017)12). Strasbourg, France: Council of Europe.

URL <https://rm.coe.int/algorithms-and-human-rights-study-on-the-human-rights-dimension-of-aut/1680796d10>

The debate about ethical and social dimensions of automated systems is frequently limited to bias. The following talk provides an overview of the debate around automation/algorithms/AI in information systems and the confusion around concepts of ethical and regulatory solutions within it. It suggests that there is a need to answer key questions about algorithmic accountability are answered: accountable to whom, where and what for. It also suggests some key shifts in the debate to ensure meaningful accountability rather than just a fig leaf. The talk is based on the two publications listed below.

References

- 1 Ben Wagner. “Algorithms and Human Rights: Study on the human rights dimensions of automated data processing techniques and possible regulatory implications (No. DGI(2017)12).” Council of Europe, 2018 <https://rm.coe.int/algorithms-and-human-rights-study-on-the-human-rights-dimension-of-aut/1680796d10>.
- 2 Ben Wagner. “Ethics as an Escape from Regulation: From ethics-washing to ethics-shopping?” In *M. Hildebrandt (Ed.), Being Profiling. Cogitas ergo sum*. Amsterdam University Press, 2018.

4.5 Raising Users’ Awareness for their Exposure to Cloud Services

Martin Henze (RWTH Aachen, DE)

License © Creative Commons BY 3.0 Unported license

© Martin Henze

Main reference Martin Henze, Jan Pennekamp, David Hellmanns, Erik Mühmer, Jan Henrik Ziegeldorf, Arthur Drichel, Klaus Wehrle: “CloudAnalyzer: Uncovering the Cloud Usage of Mobile Apps,” in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, ACM, 2017.

URL <http://dx.doi.org/10.1145/3144457.3144471>

Users are often unaware of their exposure to cloud services, e.g., when sending and receiving emails or when interacting with mobile apps on their smartphones. However, only if users are aware of (the extent of) their exposure to cloud services, they can make informed decisions and exercise their right to privacy. As a foundation to put users back into control over their privacy, we hence consider it necessary to uncover their exposure to cloud services and raise their awareness of resulting privacy risks. In this talk, we present approaches to provide users with transparency over their individual exposure to cloud services along two deployment domains for cloud services even less technically proficient users interact with on a daily basis: email and mobile apps on smartphones. Furthermore, we discuss how to apply the concept of comparison-based privacy to enable users to put their cloud usage into context through comparison with their peers in a privacy-preserving manner.

References

- 1 Martin Henze, Ritsuma Inaba, Ina Berenice Fink, and Jan Henrik Ziegeldorf. Privacy-preserving Comparison of Cloud Exposure Induced by Mobile Apps. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. ACM, 2017.
- 2 Martin Henze, Jan Pennekamp, David Hellmanns, Erik Mühmer, Jan Henrik Ziegeldorf, Arthur Drichel, and Klaus Wehrle. CloudAnalyzer: Uncovering the Cloud Usage of Mobile Apps. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. ACM, 2017.
- 3 Martin Henze, Mary Peyton Sanford, and Oliver Hohlfeld. Veiled in Clouds? Assessing the Prevalence of Cloud Computing in the Email Landscape. In *Proceedings of the 2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE/IFIP, 2017.

4.6 Purpose-driven provenance solutions

Melanie Herschel (*Universität Stuttgart, DE*)

License © Creative Commons BY 3.0 Unported license
© Melanie Herschel

Provenance in general is meta-data about the production process of an end product, including digital data. Depending on the intended purpose, the type of process, the underlying data, and the processing environment, different provenance solutions to capture, store, and use provenance are conceivable. This talk argues on the necessity to design, adapt and optimize provenance solutions to specific environments. We briefly present two examples of such purpose-driven provenance solutions: provenance for debugging data processing to account for processing errors or ensure more robust processing and provenance used to document and analyze visual computing program behavior.

References

- 1 Melanie Herschel, Ralf Diestelkämper, Houssem Ben Lahmar: A survey on provenance: What for? What form? What from? *VLDB Journal*, 26(6): 881–906 (2017)

4.7 Accountable systems. Some practical experiences from a governmental perspective.

Heleen Louise Janssen (*Ministry of the Interior and Kingdom Relations, NL*)

License © Creative Commons BY 3.0 Unported license
© Heleen Louise Janssen

How are constitutional law and digital technologies being reconciled from an ‘on the ground perspective’? Focus in this lightning talk is on fundamental rights. Fundamental rights are supposed to be solid, at least for a generation. Because of their broad application and large time-span they are inherently vague. Fundamental rights apply in principle solely to public entities, but may still have impact on private relationships (e.g. between companies and consumers or users) via ordinary legislation or in the explanation of tort laws in court cases.

In my advisory practice from the Constitutional Affairs and Legislation Department concerning the use of technological tools with the aim to execute policy, we strongly recommend that fundamental rights issues are already being solved in the design stage. Artefacts have politics–technological applications are value laden.

As regards the reconciliation of constitutional legal reality and technological reality, the modernisation of the right to communication secrecy was presented. The constitution (dating from 1983) only protects content that is transported or transmitted by physical means of mail, telephone and fax (a closed list of means). The modernisation entailed a broadening to all possible means transporting or transmitting content, covering also the electronic means of communication to make it future-proof. Various technological and legal issues had to be reconciled. Is content that is transmitted via WhatsApp or Facebook being protected, are they ‘means’ like a telephone or a letter? Why is an identifiable receiver important in a world where communication can be multiplied a billion times? When does meta-data (from a technological perspective not content) come close to or even become (legal) content, and what are the consequences for its legal protection? Is content transmitted in IoT protected by communication secrecy?

In the second example about the modernisation of the Intelligence and Security Services Act (ISSA), comparable issues from a fundamental rights perspective had to be solved. Issues dealt with in the context of privacy and communication secrecy concerned the question whether ‘scraping’ the internet as an open source would invade privacy. Another question that had to be solved was how the standing big data processing methods could be reconciled with data protection requirements such as purpose limitation and data minimisation. The law was presented in an internet consultation (1100 reactions) and a Privacy Impact Assessment (done by independent researchers) that was presented with the Bill to Parliament.

Research undertaken by the University of Utrecht (NL) at the request of the Ministry of the Interior has demonstrated that not only privacy, but also the right to equal treatment and procedural rights (access to court, right an effective remedy) are under severe pressure due to the application of big data analytics, IoT and AI.

4.8 Establishing Requirements for Accountable Systems: The Case of Elections

Joshua A. Kroll (University of California – Berkeley, US)

License  Creative Commons BY 3.0 Unported license
© Joshua A. Kroll

In setting requirements for designing a system, designers must consider how the system’s mechanisms support its high-level goals. These goals can include the reflection of important human values, such as accountability or consistency with social, political, and legal norms. It is also important to hold the process that sets these requirements accountable so as to minimize the gap between the requirements as desired by stakeholders and the requirements as they are articulated or realized in the final fielded system.

We consider the case of designing an election system, which has received much attention in the security literature [2, 1]. Elections provide an excellent test case for thinking about design that supports accountability and other values. In addition to the functional requirement of registering and tallying voter intent, election systems must attest convincingly to the integrity of results even in difficult, adversarial political and security environments, and often need to do so while maintaining the privacy of ballot information.

To accomplish this, election systems must track eligible voters from registration through casting and also maintain a strong and verifiable chain-of-custody on ballot materials from ballot design through audits that may come after the certification of results. Additionally, election systems need to be sensitive to usability for all voters, to avoid inadvertently disenfranchising certain sub-populations. Election systems must also be highly available and resilient to many types of failure so as to meet the requirement that they capture voter intent reliably within a short, fixed window of time. Post-election audits can be used to increase confidence in a result or to challenge the outcome. At every stage, the system must generate sufficient detailed evidence of correct operation so that subsequent challenges do not undermine the legitimacy of the announced outcome.

References

- 1 Joseph Lorenzo Hall. *Policy mechanisms for increasing transparency in electronic voting*. PhD thesis, 2008.
- 2 Joseph Lorenzo Hall. Election auditing bibliography, 2010. https://josephhall.org/papers/auditing_biblio.pdf.

4.9 Confidential Analytics – Use Cases and Building Blocks

Maximilian Ott (CSIRO – Alexandria, AU)

License © Creative Commons BY 3.0 Unported license
 © Maximilian Ott
URL <https://n1analytics.com>

Data collaboration between organisations using sensitive data is not only increasing in volume but also in the problems it creates. While there are extremely interesting ethical questions raised about the intent of many of those collaborations, most of today’s problems are caused by the “process” leading up to the desired outcome.

Specifically, today’s analytics tools require that all the input data is brought together in a single place, requiring at least one party to “disclose” potentially sensitive data.

In the first part of this talk, we describe a few use cases for data collaboration. We then briefly sketch an alternative analytics approach where all the sensitive data can remain with the data owner. This is followed by a short description of the required building blocks: arithmetic with encrypted numbers, federated algorithms and protocols, as well as private record linkage.

4.10 Accountable systems, messy environments

Michael Veale (University College London, GB)

License © Creative Commons BY 3.0 Unported license
 © Michael Veale
Joint work of Michael Veale, Max Van Kleek, Reuben Binns, Lilian Edwards
Main reference Michael Veale, Max Van Kleek, Reuben Binns: “Fairness and Accountability Design Needs for Algorithmic Support in High-Stakes Public Sector Decision-Making”, in Proc. of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018, p. 440, ACM, 2018.
URL <http://dx.doi.org/10.1145/3173574.3174014>

Responding to concerns around ‘algorithmic harms’, fields such as law and computer science have doubled down on efforts to create facilitating technologies and environments for accountable systems. Canonical problems include biased, opaque automated decision-making systems, and canonical solutions include discrimination-aware machine learning, user-facing explanation facilities, or data protection information rights. In this talk, I present two areas where these canonical visions of these challenges clash with practice. Firstly, some results from a study interviewing 27 public sector machine learning practitioners is presented, where their attempts to cope with accountability issues are shown to be more textured in practice than commonly suspected. Secondly, I consider the information rights in the EU General Data Protection Regulation, and how they might be able to support provenance and transparency efforts, and may be promising at scale or where used collectively, but are considerably messier in practice than the text would lead a reader to believe.

4.11 Trust, Responsibility, and Explanation

Michael Winikoff (University of Otago, NZ)

License  Creative Commons BY 3.0 Unported license
© Michael Winikoff

Joint work of Virginia Dignum, Frank Dignum, Michael Winikoff

My talk considered the overarching issue of trusting autonomous systems, and the factors that lead to appropriate levels of trust in autonomous systems. I particularly focussed on the role of explanation, and described an explanation mechanism and its evaluation.

Participants

- Virgilio Almeida
Federal University of Minas
Gerais-Belo Horizonte, BR
- Jean Bacon
University of Cambridge, GB
- Jennifer Cobbe
University of Cambridge, GB
- Jon Crowcroft
University of Cambridge, GB
- Lilian Edwards
The University of Strathclyde –
Glasgow, GB
- David Evers
University of Otago, NZ
- Krishna P. Gummadi
MPI-SWS – Saarbrücken, DE
- Tristan Henderson
University of St Andrews, GB
- Martin Henze
RWTH Aachen, DE
- Melanie Herschel
Universität Stuttgart, DE
- Heleen Louise Janssen
Ministry of the Interior and
Kindom Relations, NL
- Joshua A. Kroll
University of California –
Berkeley, US
- Bertram Ludäscher
University of Illinois at
Urbana-Champaign, US
- Derek McAuley
University of Nottingham, GB
- Christopher Millard
Queen Mary University of
London, GB
- Ken Moody
University of Cambridge, GB
- Maximilian Ott
CSIRO – Alexandria, AU
- Frank Pallas
TU Berlin, DE
- Thomas Pasquier
University of Cambridge, GB
- Silvia Puglisi
The Tor Project – Seattle, US
- Margo Seltzer
Harvard University –
Cambridge, US
- Jatinder Singh
University of Cambridge, GB
- Barbara Staudt Lerner
Mount Holyoke College –
South Hadley, US
- Michael Veale
University College London, GB
- Ben Wagner
Wirtschaftsuniversität Wien, AT
- Michael Winikoff
University of Otago, NZ
- Martina Zitterbart
KIT – Karlsruher Institut für
Technologie, DE



Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research

Edited by

Pekka Abrahamsson¹, Jan Bosch², Sjaak Brinkkemper³, and
Alexander Mädche⁴

1 University of Jyväskylä, FI, pekka.abrahamsson@jyu.fi

2 Chalmers University of Technology – Göteborg, SE, jan@janbosch.com

3 Utrecht University, NL, s.brinkkemper@uu.nl

4 KIT – Karlsruher Institut für Technologie, DE, alexander.maedche@kit.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 18182 “Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research”.

Seminar April 29 – May 2, 2018 – <http://www.dagstuhl.de/18182>

2012 ACM Subject Classification Software and its engineering → Software organization and properties; Software and its engineering → Software creation and management; Software and its engineering → Software creation and management

Keywords and phrases Software Engineering, Software Ecosystems, Software-intensive Business, Software Production, Software Startups


Digital Object Identifier 10.4230/DagRep.8.4.164

Edited in cooperation with Slinger Jansen and Karl Werder

1 Executive Summary

Slinger Jansen (Utrecht University, the Netherlands, slinger.jansen@uu.nl)

Karl Werder¹ (Universität Duisburg – Essen, Germany)

License  Creative Commons BY 3.0 Unported license
© Slinger Jansen and Karl Werder

Software producing organizations (SPOs) face challenges every day. Whether they are open source consortia or commercial software product companies, they all face the challenges of changing demands, rapidly evolving technology, and a dynamic ecosystem in which their products and services need to operate. SPOs need to rethink their operating models and benefit from current and future trends. E.g. agile software development and DevOps allow them to respond swiftly to changes in their environment, embracing uncertainty. Particularly in conjunction with machine learning and artificial intelligence, SPOs can generate strategic competitive advantages. Particularly companies with a long history in a given domain, such as SAP and Volkswagen, seem to be too comfortable with their status quo. Meanwhile, smaller companies drive innovation on many fronts. Examples are Provenance that benefits from blockchain technology to revolutionize trust in goods, or Tesla and Local Motors that push autonomous cars into consumer markets.

¹ Now at University of Cologne, Germany, werder@wiso.uni-koeln.de



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research, *Dagstuhl Reports*, Vol. 8, Issue 04, pp. 164–198

Editors: Pekka Abrahamsson, Jan Bosch, Sjaak Brinkkemper, and Alexander Mädche



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The challenge to make these organizations successful is multi-disciplinary. First, there exist technology challenges, such as eliciting and prioritizing requirements, dealing with platforms and technology standards, and operating in complex technology landscapes that constrain and enable their technology. Secondly, there exist adoption challenges: organizations need to find ways to convince their target users to adopt their technologies and to coordinate evolving technologies to provide the most valuable end-user experience. Thirdly, there exist business model challenges, where these organizations must find ways to maximize profit from their innovations and technologies. Because of the pervasiveness of software, the challenges are observed everywhere in the economy, whether it is logistics, online marketing, or e-health. Furthermore, they are applicable to organizations in every stage of development, whether it is a software startup or a software giant that has influenced the market consistently for decades.

Hence, this Dagstuhl Seminar invited thought leaders from academia and industry to share their knowledge and experiences. Participants were asked to share a short position statement of max 300 words and participate in the development of a groundbreaking research agenda. These efforts aimed to increase visibility and impact of software production research and to set a course for the next decades. In addition, the seminar helped bringing together scholars and industry practitioners from different communities, such as product management, technology management, information systems, software engineering, and human-computer interaction in order to sharpen and define the joint community of Software-intensive Business (see Section 4.3.1).

A central outcome of the seminar was the agreement to use the term Software-intensive Business in order to describe the joint community with members of great diversity. Furthermore, the seminar focused on

- defining core concepts and identifying a roadmap
- Software-intensive Business and technology artifacts
- research needs in continuous experimentation & innovation
- lifecycle and research of software ecosystems
- research data for Software-intensive Businesses

As a major result from the seminar, the following achievements have been identified:

1. research a clear agenda for the field of Software-intensive Business research
2. carving out trends and research challenges in further depth
3. forming groups for continuous collaborations on different elements of the research agenda
4. organize bi-weekly meetings on-line for community building and research sharing.

2 Contents

Executive Summary

<i>Slinger Jansen and Karl Werder</i>	164
---	-----

Position Statements


Understanding Software Ecosystems through Visual Analytics and Machine Learning <i>Rahul C. Basole</i>	168
Transforming To A Software Business <i>Jan Bosch</i>	169
Innovation + Velocity + Pivoting in Software Production: The New Normal <i>Christoph Bussler</i>	170
Platform versus Non-Platform Company Performance: Some Exploratory Data Analysis, 1995-2015 <i>Michael A. Cusumano</i>	171
Platform Elasticity for Fast Time-to-Critical Mass and Take-Off <i>Samuel Fricker</i>	171
Research statement <i>Jens Foerderer</i>	172
Feature-Oriented Development in Industrial Automation Ecosystems <i>Paul Grünbacher</i>	173
Analyzing the Mutual Quality Impact of Business Processes and Information Systems <i>Robert Heinrich</i>	174
Research Statement <i>Armin Heinzl</i>	174
Software Engineering evolution <i>Mika Helenius</i>	175
Position Statement <i>Georg Herzwurm</i>	176
From Managing Your Ecosystems to Repositioning Your Business <i>Helena Holmström Olsson</i>	176
Some Research Directions for Software Ecosystems & Platforms <i>Sami Hyrynsalmi</i>	177
Engineering FLOSS Ecosystems for Impact and Sustainability <i>Zhi Jin</i>	178
Academic Structures and Terminology <i>Hans-Bernd Kittlaus</i>	179
Position Statement <i>Thomas Kude</i>	180
Position Statement <i>Alexander Mädche</i>	180

Challenges in Software Ecosystems and Product Development	
<i>Efi Papatheocharous</i> ,	181
Elements of Platform-based Ambidexterity: An Empirical Investigation	
<i>Balasubramaniam Ramesh</i>	182
Minimum Viable Products – The Road Ahead	
<i>Guenther Ruhe</i> ,	183
Only few can be platform owners	
<i>Kari Smolander</i>	183
Data for Performing Research on Software Business, Platforms, and Ecosystems	
<i>Diomidis Spinellis</i>	184
Software Ecosystems Research Agenda Update	
<i>Slinger Jansen</i>	185
AI and Software Business	
<i>Pasi Tyrväinen</i>	186
High-speed and Sustainable Development of Software Startups	
<i>Xiaofeng Wang</i>	187
Understanding new development trends, exploring large data, and pushing the boundaries of innovation	
<i>Karl Werder</i>	188
Improving handling business model changes for software-intensive organizations	
<i>Krzysztof Wnuk</i>	189
Working Groups	
Working Group on the Software Ecosystem Research Agenda	
<i>Paul Grünbacher, Jens Förderer, Zhi Jin, Samuel Fricker, Rahul Basole, Slinger Jansen</i>	190
Working Group on Software Business and Technology Lifecycle Artifacts	
<i>Sjaak Brinkkemper, Armin Heinzl, Robert Heinrich, Alexander Maedche, Kari Smolander</i>	192
Working Group on Software-Intensive Business Research: Definition and Roadmap	
<i>Karl Werder, Sjaak Brinkkemper, Jan Bosch, Michael A. Cusumano, Georg Herzwurm, Helena Holmström Olsson, Sami Hyrynsalmi, Hans-Bernd Kittlaus, Thomas Kude, Tiziana Margaria, Efi Papatheocharous, Diomidis Spinellis, Pasi Tyrväinen, Xiaofeng Wang</i>	193
Working Group on Health Measurement of Open Source Projects and Ecosystems	
<i>Slinger Jansen, Paul Grünbacher, Efi Papatheocharous, Diomidis Spinellis</i>	196
Working Group on Research Data for Software Intensive Business	
<i>Slinger Jansen, Diomidis Spinellis, Krzysztof Wnuk</i>	197
Participants	198

3 Position Statements

3.1 Understanding Software Ecosystems through Visual Analytics and Machine Learning

Rahul C. Basole (Georgia Institute of Technology – USA, basole@gatech.edu)

License  Creative Commons BY 3.0 Unported license
© Rahul C. Basole

The software industry is fiercely competitive and highly dynamic with companies of all sizes and geographic location battling for market share and new entrants emerging constantly. To survive in this hypercompetitive environment, companies must continuously innovate, not just in terms of software functionalities but also in ways they are designed, developed, offered and licensed. Ecosystemic thinking is thus critical.

My research examines the complexity of different types of software ecosystems from micro- to macro-level perspectives using emerging computational and visual analytic approaches. My core argument is that the scale, scope, and evolving dynamics of software ecosystems demand novel data-driven research methods and that we can support our understanding and augment decision making through interactive visual analytic approaches.

Some of my recent and ongoing studies include the examination of API and SDK ecosystems, digital platforms, digital infrastructures, dynamics of developer ecosystems, software alternatives, microservices, and global software startup ecosystems. Our investigations are enabled and driven by large-scale, heterogeneous (structured and unstructured) publicly available and proprietary data. Since the goal of my research is to create actionable insights, and not just archival knowledge, my lab develops interactive, visual, human-centered tools (e.g., ecobox, graphiti, epheno, pulse, etc.) that enable exploration, discovery, and sense-making of the structure and dynamics of such software ecosystems. A set of sample (static) visualizations at different software ecosystem levels is shown below.

There are many exciting open research opportunities in the study of software businesses, platforms, and ecosystems using visual analytics and machine learning that would be worthy of further discussion.

- What are evolutionary patterns of software startups, platforms, and ecosystem and how do they relate to success and failure? Are there segmental or geographic differences?
- How do developer ecosystems react and organize to software launches and changes?
- How do APIs and SDKs complement, enhance, or constrain interfirm relationships?
- How can software platforms orchestrate complex evolving ecosystems and shield against disruption? What role do developers play?
- How do firms adopt, experiment, and discard digital infrastructure technologies?
- How can you anticipate, prepare, and adapt to changes in software ecosystems?

3.2 Transforming To A Software Business

Jan Bosch (Chalmers University of Technology – Sweden, jan@janbosch.com)

License © Creative Commons BY 3.0 Unported license
© Jan Bosch

Digitalization is concerned with creating new revenue and value producing opportunities through the use of digital technologies. In practice, this typically refers to increased product value by adding software, licensing software as a standalone product or services using the data generated by users and products.

Although the literature is filled with examples of companies that were born digital, such as Google, Uber, Booking and AirBnB, the fact of the matter is that there are thousands of companies that need to transform their business model and product portfolio in response to the emergence of digital technologies and the digitalization trend. The data shows that these companies are not very successful at this. For instance, the duration of companies on the Fortune 500 has now shortened to 10 years and digitalization is the predominant cause of disruption for the companies that have disappeared from the list.

Our experience in Software Center (www.software-center.se) shows that there are several challenges that companies need to contend with:

- Top leadership lacks knowledge and has a quarterly results focus: most top leaders have established their careers in non-digital technologies and have, for decades, been trained to focus on delivering on the quarterly results.
- The ecosystem is holding back companies: Even if the company sees the need and wants to change, its customers and partners often are unwilling to change with the enlightened company. And the company can't implement the change without alienating its customers and partners. This leads to a catch 22 situation for many companies.
- Disruptive innovation is unpredictable: Established companies are used to predictable return of investment on investments in sustaining innovations. Disruptive innovation typically follows a power function meaning that most innovations fail, but the few that succeed generate outsized returns. Having to go through dozens of attempts until striking gold is difficult to stomach for most leaders.
- Subsidizing one side of the market is hard: In many cases, a company aspiring to become a platform company needs to subsidize at least one side of a multi-sided market and potentially all sides for a period of time in order to reach the "ignition point". Companies that have become successful in a traditional value chain have difficulty in subsidizing its customers as the focus tends to be on margins.
- Data ownership: Especially in the B2B space, it is often unclear who owns the data and the customer relationship and if it is clear, the company needs to provide something of value in return for gaining access to relevant data. Especially for traditional companies, sacrificing certain revenue from existing customers for potential revenue from a nascent business around data is difficult to stomach.

Although the above does not necessarily constitute a research agenda, it is a representative overview of some of the challenges that traditional product companies looking to transition towards a software, data and/or platform business experience. As a research community, rather than only focusing on new companies that are born digital, we should also study the challenge of transformation and provide solutions to existing companies looking to continue to be successful in a digitalized world.

3.3 Innovation + Velocity + Pivoting in Software Production: The New Normal

Christoph Bussler² (Oracle Corporation – USA, christoph.bussler@oracle.com)

The new normal in software production is that significant innovation has to be delivered at rapid development velocity with the ability to pivot at any time reflecting customer preferences, feedback and uptake.

The predominant software engineering methodology supporting the new normal in software production is a combination of agile methodology, A/B testing (if possible at all) and daily releases into production.

Why is software production and the resulting software products and services still rough sailing despite the advances in software engineering methodology?

- Observation 1: Everybody immediately recognize and appreciates software or services that are fast, easy and simple to use as well as exhibit consistency in terminology, behavior and actions.
- Observation 2: How often do you come across “good” software products or services? And how often do you have a negative reaction? Software production teams are in general multicultural, multilingual, distributed (across time and geography) as well as differ greatly in level of ambition, education and experience.

The agile methodology gets in the way of producing “good” software: it does not provide a general development direction, does not enforce consistent use of terminology, does not foster a consistent software architecture, and documentation as well as planning take a back seat – if present at all.

Time pressure gets in the way as well: shortcuts are taken and functionality is implemented incompletely focusing mostly on the main execution path (the Happy Path).

An interesting research topic to support the new normal would be “improvement infusion” by providing feedback through continuous automated observation of software production engineering activities. For example, a terminology analysis environment can observe the use of terminology in team communication and point out (possibly) inconsistent use.


Areas of analysis can be team communication, code and code changes, engineering and end user documentation, as well as test case stability and performance.

Types of feedback can be highlighting of inconsistent use of terminology, incomplete functionality implementation, requirements vagueness and instability, use case incompleteness and instability, test execution success rate degradation and missing test coverage. The result of “improvement infusion” would be the increase of software production quality based on observations, not regulation and constraints. “Improvement Infusion” supports the velocity and pivoting without attempting to change the predominant engineering culture.

² The views expressed here are my own and do not necessarily reflect the views of Oracle.

3.4 Platform versus Non-Platform Company Performance: Some Exploratory Data Analysis, 1995-2015

Michael A. Cusumano (MIT Sloan School of Management – USA, cusumano@mit.edu)

License  Creative Commons BY 3.0 Unported license
© Michael A. Cusumano

Numerous publications have described platform companies and their strategies and operations. However, there has not been a large-sample statistical study answering questions such as: Are platform companies more profitable than non-platforms? Are they more valuable? Are there differences among types of platform?

As an exploratory analysis, we divided all platforms into two basic types for innovation and transactions. Innovation platforms provide common building blocks that ecosystem partners can use to create “complementary” products and services. Microsoft Windows, Google Android, Apple iOS, and Amazon Web Services are commonly used operating systems and cloud computing services that serve as innovation platforms for computer and smartphone ecosystems. Transaction platforms make it possible for people to access or buy and sell a variety of goods and services, or to share information. Google Search, Amazon Marketplace, the Facebook Social Network, Twitter, and Tencent’s WeChat are examples of commonly used transaction platforms.

We defined a platform company as a firm that had at least 20 percent of revenues from businesses driven by network effects. We analyzed the Forbes Global 2000 list for 2015 and counted 46 platform companies, with 19 innovation and 27 transaction platforms. We then created a data set of over 30,000 yearly firm observations from 2005 through 2015.

We conducted simple regressions and T-tests. The means were significantly different between platform and non-platform companies on several dimensions: Operating profits divided by sales; market value multiples (ratio of sales to market value and price-to-earnings ratios); and absolute sale levels. Compared to transaction platforms, the innovation platforms had higher market values, sales, operating income, employee numbers, and R&D as well as sales and marketing expenditures. Transaction platforms had higher market values.

We confirmed that publicly listed platform companies were more profitable and more valuable than non-platform companies. However, we also identified several problems with this type of study that warrant further discussion.

3.5 Platform Elasticity for Fast Time-to-Critical Mass and Take-Off

Samuel Fricker (FHNW University of Applied Sciences and Arts Northwestern Switzerland – Switzerland, samuel.fricker@fhnw.ch)

License  Creative Commons BY 3.0 Unported license
© Samuel Fricker

We propose that keystones should design platforms for elasticity if they want to achieve a short time-to-critical mass for network effects to take off. We came to that position in our work for building a marketplace for open development of systems of artificial intelligence (AI) ³. There, the extent of third-party data, talent, and AI algorithm offerings as well as

³ H2020-ICT-01-2016 project www.bonseyes.com

the effort and convenience of trying and abandoning such offerings seems to affect platform adoption.

Elasticity is a key characteristic of Cloud computing and stands for rapid on-demand automated provisioning of capabilities (scaling out), possibly in a self-service mode, and rapid releasing of these capabilities (scaling in) ⁴ for efficient resource management ⁵. The Cloud is considered an essential platform for digital businesses and could generate a revenue of EUR 44.8 billion in Europe in by 2020 ⁶.

Elasticity could be brought to any platform with mechanisms to provision, use, and release assets on-demand while enforcing business models, preventing misuse, and enabling trusted choice. We posit that the convenience of value access offered by elasticity could affect the threshold of individuals for getting engaged. The smaller the threshold is, the faster the critical mass of adopters is reached and the network effects take off, letting the platform and a healthy ecosystem self-sustain ^{7 8}.

3.6 Research statement

Jens Foerderer (University of Mannheim – Germany, foerderer@uni-mannheim.de)

License  Creative Commons BY 3.0 Unported license
© Jens Foerderer

Researchers are interested in platform strategies because they involve a fundamentally different set of decisions than conventional “pipeline” strategies. Pipeline strategies create value via a linear series of activities in the sense of the classic value-chain model. Inputs at one end of the chain (e.g., resources) are transformed in various steps into a valuable output: the finished product. In contrast to pipeline strategies, platforms create value by leveraging the innovation capabilities of an independent “crowd” outside of the focal firm’s boundary. When firms follow a platform strategy, value-creating activities are less concerned with the coordination of production and supply but rather with the orchestration of complementary products and services. Thus, platform strategies require the focal firm to focus less on designing, developing, and distributing products but rather to focus on implementing an effective governance of third parties.

The governance of third parties requires a more fundamental and empirically assessed understanding with regards to **cooperation and competition** mechanisms. The phenomenon of platform owners actively competing with complementors has attracted attention substantial attention by researchers and policy-makers. Yet, our understanding today is particularly limited with regards to three questions:

- **Under which conditions does competition with complementors hurt or promote complementary innovation?** Extant research so far has yielded contradictory findings, suggesting that platform owners’ competition with complementors can crowd out

⁴ P. Mell, T. Grance (2010): “The NIST Definition of Cloud Computing,” *Communications of the ACM* 53, 6: 50.

⁵ G. Galante, L. de Bona (2012): “A Survey on Cloud Computing Elasticity,” *IEEE/ACM 5th Intl Conf on Utility and Cloud Computing*, Chicago, Illinois, USA.

⁶ P. Wauters et al (2014): “Measuring the Economic Impact of Cloud Computing in Europe,” *Final Report for the European Commission*. Deloitte.

⁷ E. Rogers (1995): *Diffusion of Innovations*, Free Press.

⁸ S. Jansen (2014): “Measuring the health of Open Source Software Ecosystems: Beyond the Scope of Project Health,” *Information and Software Technology* 56, 11: 1508-1519

innovation, but also increase the overall innovation within the ecosystem by attracting new consumers to the market and setting stronger incentives for differentiation [1]. It appears timely to study the conditions under which competition is innovation-promoting or innovation-hurting.

■ **How can intra-platform competition be effectively regulated by policy-makers?**

Almost overnight, antitrust and platform regulation has become a widely debated topic with policy-makers and regulators around the world considering platform regulation (EU, US, Japan) and some even implementing regulations (Russia). The regulations considered are mostly derived from standard antitrust models in the early 20th century, making it questionable whether they apply to two-sided platform markets. It appears timely to assess whether conventional antitrust approaches are effective in limiting intra-platform competition.

■ **How can platform owners effectively set agendas for the overall ecosystem?**

Platform ecosystems are characterized by independent complementors cooperating more or less at arm's length with a central platform owner. Yet, the platform's success (and competitive advantage compared to rival platforms) deliberately depends on coordinating not only individual complementors, but also coordinating the overall ecosystem. This is, however, a theoretically complex undertaking, as ecosystems often encompass thousands of firms that are impossible to coordinate via the mechanisms we know from conventional interfirm coordination literature. It appears therefore timely to understand the mechanisms (technological, informational, organizational) by which platform owners implement the overall agenda for the ecosystem.

References

- 1 George Valença, Carina Alves, Virgínia Heimann, Slinger Jansen, and Sjaak Brinkkemper. Competition and collaboration in requirements engineering: a case study of an emerging software ecosystem. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 384–393. IEEE, 2014.

3.7 Feature-Oriented Development in Industrial Automation Ecosystems


Paul Grünbacher (Johannes Kepler Universität Linz – Austria, paul.gruenbacher@jku.at)

License © Creative Commons BY 3.0 Unported license
© Paul Grünbacher

Feature-oriented development has been proposed as an approach for engineering large-scale, variant-rich software systems. For instance, features models are widely used to capture the knowledge about product lines and configurable software systems. Features exist at different levels of granularity and define the perspectives of product management, technical solution architecture, and product configuration. We report ongoing work towards a feature-oriented software development approach we are currently developing with an industry partner in the domain of industrial automation ecosystems. Our work is based on empirical studies we conducted on the characteristics and use of features in industry. We present the FORCE modeling approach, which supports modularizing feature models for different purpose and different levels. Our tool environment exploits feature-to-code mappings and configuration-aware analysis. We further present our plans for supporting developer awareness on evolving features in ecosystems.

3.8 Analyzing the Mutual Quality Impact of Business Processes and Information Systems

Robert Heinrich (Karlsruher Institut für Technologie – Germany, robert.heinrich@kit.edu)

License  Creative Commons BY 3.0 Unported license
© Robert Heinrich

In complex technology landscapes new opportunities for the evolution of business goals and processes come up due to novel capabilities of software. Business processes (BPs) and information systems (ISs) mutually affect each other in non-trivial ways. The complex interrelations between BPs and ISs, however, are not adequately researched so far. Especially interrelations between quality properties (e.g., performance or maintainability) concerned with business people and those concerned with IS developers are not well understood. Frequently, the representation of quality aspect differs in the BP and IS domain.

Engineering methods for aligning one domain to the quality objectives of another are missing. One major reason for insufficient quality engineering is that current approaches lack an integrated consideration of quality aspects among several domains. Frequently, BPs and ISs are not well aligned, meaning that BPs are designed without taking IS impact into account and vice versa. Neglecting the mutual impact between BPs and ISs in development leads to serious practical issues. On the one hand, it is not known whether a particular requirement can be satisfied by a proposed IS design, because it is not known how the system is used in the BP, and how this usage affects the IS quality. On the other hand, it is unknown whether a particular requirement can be satisfied by a proposed BP design, because it is unknown whether involved ISs adequately support the adherence of the requirement. Decisions in IS development are not reliably made since important BP-related information may not be considered. This may decelerate IS development due to the rework needed in subsequent development phases. The same applies to neglected IS properties in BP development.

In our research we target the alignment of BP and IS designs by developing simulation and analysis techniques based on design models to predict the quality impact of mutual interrelations between BP and IS.

3.9 Research Statement

Armin Heinzl (University of Mannheim – Germany, heinzl@uni-mannheim.de)

License  Creative Commons BY 3.0 Unported license
© Armin Heinzl

Software Development Research is an intriguing topic which has a superior role in the digital age where digital products and services are uprising. Recent developments like the agile manifesto, Scrum, and Scrum in Scrum are prominent contemporary examples.

Cognitive and mental processes in agile software development teams and the question how agile teams scale, have been among my personal research topics during the past years. They embody the question how knowledge work enfolds in a knowledge centric society.

Today's software solutions are part of sophisticated (enterprise) software ecosystems. Exchanging and aligning knowledge between platform owners and complementors is another fascinating field of investigation. These practices are part of an overall innovation strategy of the participating firms which should be explored concurrently.

3.10 Software Engineering evolution

Mika Helenius (Finnish Computer Science Research Foundation & Finnish Information Processing Association TIVIA – Finland, mika.helenius@iki.fi)

License © Creative Commons BY 3.0 Unported license
© Mika Helenius

Software is the world's most powerful and pervasive general-purpose technology of modern times defined by competency, coordination and capability. Software creates new jobs, markets and industries that did not exist before at faster phase than ever. Software makes possible to create new intangible needs, service, spheres and forms of use anywhere to be exploited in the environment before they are understood. This new industrial development is called software platform based business and economy. Due its significant economic impact on industries and societies it is highly relevant and important for business and academia to understand software business, platforms, and ecosystems evolve. How these complex modular systems-of-systems platforms are engineered – conceived, designed, implemented and operated? What kind of competencies, capabilities, tools, methodologies and technologies are need in the rapidly evolving engineering process in rigor global competition? How we should define platforms in context of economic competition, hybrid warfare and humanity?


Platform are complex software system markets and industries. They are expensive, risky and time consuming to create. It is critical for owners, executives and investors to know how platform businesses are created to understand contextual productivity and innovation aspects. Current research has mainly focused on analyzing the existing business model, ecosystem and technology layers separately. Little attention has been paid on how complex platform systems are created with less risks using platform architecture as strategy management theory to gain business and societal results. Architectural thinking has not become main stream in the management discourse even it has been highlighted as key source of value. Platform architecture as strategy comprises of three layers industry and market layer, business and ecosystem layer, and technology system layer. Multisided platform architecture as strategy support past, present and future properties in the dynamic in vivo or simulated environment.

Software engineering research needs to be expanded to cover realistic and very large-scale engineering and production challenges by establishing large scale problem based learning environment for students, practitioners and researchers. This expansion of empirical software engineering to large-scale complex systems would allow future engineering graduates and researcher study digital grand challenges in realistic settings to understand what is needed in the ecosystems to deliver value in and for the environment.

Societies need knowledge and skills to create sustaining and balanced ecological platforms, which comply with current norms, values and ethics to redefine how value is spread and cultures are saved. Platform architecture as strategy is key in solving grand challenges.

3.11 Position Statement

Georg Herzwurm (University of Stuttgart – Germany, herzwurm@wius.bwi.uni-stuttgart.de)

License  Creative Commons BY 3.0 Unported license
© Georg Herzwurm

Digital goods such as software are specific in economies of scale and network effects and thus require specific strategies and concepts for design, development and marketing. Due to the progress of digitalization in almost any market, the observed phenomena and measures gain relevance and importance in business and academia. Hence the emerging convergences in industries, suppliers, businesses and products cause a game change in markets for digital or digitized goods.

The door to the digital world is opening for more tangible goods employing the Internet of Things (IoT), actuating convergence of digital and analogue markets and value creation systems, enabling the atomization of products / services (e.g. bundled microservices instead of apps) and increasing the number of products and services and their providers. Platforms enable cooperation (i.e. development and sales) of the value creation partners. Since these partners may be cooperation partners and competitors coincidentally, there is coopetition.

Digitization is leading to a sustainable change in the software business towards a platform economy offering a huge potential for innovative business models, creating and satisfying customer needs for business success. However, successful digitized business models require on one side mastery of technology on the other side commercial expertise and HR skills for a mindset change of employees and customers.

Current research is mainly driven by technology and fosters digital innovations which will lead to business success only if technology and innovations meet customer needs, whereas needs and benefits may not be revealed yet. Its disclosure may manifest the innovation and provide the key to business success. Hence successful solutions require at first thorough understanding of customer needs and then design and development of matching processes and products.

We thrive for methods geared to design, development and market successful digitized products and business models within the paradigm of customer-centricity and embedded into a quality management framework.

We aim at systematic approaches and concepts creating sustainable customer benefits and value, designing and developing competitive, solid and profitable solutions (business value) employing customer-centered requirements, quality and product management.

3.12 From Managing Your Ecosystems to Repositioning Your Business

Helena Holmström Olsson (Malmö University – Sweden, helena.holmstrom.olsson@mau.se)

License  Creative Commons BY 3.0 Unported license
© Helena Holmström Olsson

To engage with external actors and to exchange value as part of a larger business ecosystem is one of the most prevailing trends in today's business environment and companies across domains are increasingly realizing the many benefits with engaging with external partners. To proactively engage with suppliers, vendors, distributors, retailers and customers brings with it a number of opportunities that do not present themselves when serving customers in

a one-to-one relationship which, up until now, has been the most common strategy for most companies.

In previous work and as part of the Software Center collaboration, we identified three different types of ecosystems that companies operate in. These ecosystems are related to *innovation, differentiation and commodity* and they are inherently different in nature. Typically, companies seek to involve with others and use collaborative strategies when it comes to innovation, they exclude partners and use competitive strategies when it comes to differentiation and they utilize external resources and, again, use collaborative strategies when managing their commodity ecosystem. In our work, we developed strategies for helping companies position themselves in ways that help them gain competitive power, maximize value and utilize their partner network.

However, for a company to manage – and to position itself within its existing ecosystems – is only one part of the challenge. The other – and even more important part – is to be able to re-position oneself in order to shift the power balance between oneself and the ecosystem partners when needed. Key reasons for re-positioning include e.g. to extract even more value out of the ecosystem, to avoid commoditization of one's role in the ecosystem and to avoid potential disruption.

Recently, and based on our work with the Software Center companies, we have identified a number of challenges that companies face when trying to reposition themselves, and with previous studies focusing primarily on how to manage ecosystems rather than how to reposition in ecosystems we see this as an area to explore further in future research.

3.13 Some Research Directions for Software Ecosystems & Platforms

Sami Hyrynsalmi (Tampere University of Technology – Finland, sami.hyrynsalmi@tut.fi)

License © Creative Commons BY 3.0 Unported license
© Sami Hyrynsalmi

Software ecosystems, platforms and application stores, consisting of various different organisations, have changed how software products and services are produced, distributed and maintained. The emergence of this new platform economy has been painful for some of the old kings of the castle while some newcomers have been able to build successful and sustainable business in these new environments. Nevertheless, to support software producing organisations (SPOs) in the evolving platform economy, further research work with empirical evidence is needed. In the following, I will present areas that I believe would be fruitful for further inquiries.

First, evaluating the sustainability and well-being of an ecosystem. The current work on business and software ecosystem health has produced various measures, yet there is only little empirical evidence available to support the current researches.


Second, studying the influence of multi-homing on software ecosystems. One of the key characteristic differentiating software ecosystems from business ecosystems is the relative easiness of a SPO to offer their products and services in several competing ecosystems at the same time. However, this area has been studied only a little.

Third, understanding software start-ups as a part of niche creation and renewal of an ecosystem as well as the whole software industry. The majority of existing software ecosystem and platform literature has focused on the keystones and ecosystem orchestrators whereas there are studies addressing independent SPOs, yet their number remains small.

In our research, we have thus far focused on software ecosystems and often from the perspective of independent SPOs. In addition, we have studied the financial aspects of software start-ups and we are interested to focus more on new entrepreneurs entering an ecosystem. However, in starting software companies, business development practices are often tightly intertwined with software development activities and thus a holistic view is needed to understand the start-ups.

3.14 Engineering FLOSS Ecosystems for Impact and Sustainability

Zhi Jin (*Peking University – China, zhijin@pku.edu.cn*)

License  Creative Commons BY 3.0 Unported license
© Zhi Jin

Millions of participants, from independent volunteers to paid representatives of companies or government organizations, are creating and maintaining a huge number of open source software (OSS) eco-systems, such as the Linux Kernel, Android, and OpenStack, which have had a significant impact, not only on the software industry, but also on software-intensive organizations in both the public and private sectors. Despite the substantial amount of research on FLOSS in disciplines such as software development, organizational science, management, and social sciences, it remains unclear how and why OSS ecosystems form, how they achieve their impact, or how they sustain themselves. The data recorded in vast open source and commercial software repositories provide rich opportunities to investigate how people develop software and how they interact with each other and with their environment to accomplish their tasks, and how large-scale projects grow and sustain adapting to the ever-evolving environment. The following shows our studies on this area ranging from the learning trajectory of developers to the participation of companies, and to the health and sustainability of communities and ecosystems.

- Q1: How to retain people?
 - In GNOME and Mozilla, over 10 years, more than 70% of contributors are One-Time-Contributors. Only 3.6% in Gnome and 0.9% in Mozilla joiners become Long Term Contributors.
 - People behave differently when joining. The intension for joining the community depends on the willingness, the macro-climate, and micro-climate.
 - Willingness and climates impact the chance of becoming Long Term Contributors. Practice of the first month affects chance of becoming long term contributors.
 - This can be used to predict who will stay with the project for long term based on the initial behavior of the newcomer. And the FLOSS community could devote their valuable attention to people who are more likely to be useful to the sustainability of the project.
- Q2: How companies participate in FLOSS?
 - For all the types of projects, the full-solution-oriented companies lead the development. Their proportion of commits is 80.99% in median. Together with the specific-business-oriented companies they contribute more than 88% of the commits and approximately 89% of the developers for almost all project types in median.
 - Community-oriented organizations help the team building: focus on developing infrastructure, deployment etc.

- Usage-oriented companies improve user experiences: have a preference on the development of infrastructure, deployment, management tools, document, etc.
- Q3: How do Ecosystems Evolve and Scale?
 - How fast does the Linux grow? The amount of work continues to grow. Tasks for drivers contains most of the changes of the system. The number of joiners has been decreasing.
 - How do the maintainers and their workload evolve? The number of maintainers keeps increasing. The average workload of maintainers seems to decrease instead of growing. 80% work is done by 20% people in drivers, modules have a much more even distribution of work.
 - How well does the Linux ecosystem scale? Adding more maintainers to a file yields only a power of 1/2 increase in productivity, e.g., four parallel maintainers are needed to double the overall output. This suggests limits to scalability that can be achieved by adding multiple maintainers to the same subsystem.

3.15 Academic Structures and Terminology

Hans-Bernd Kittlaus (InnoTivum Consulting & ISPMA (International Software Product Management Association) – Germany, hbk@innotivum.de)

License © Creative Commons BY 3.0 Unported license
© Hans-Bernd Kittlaus

Research in this area faces some structural and fundamental problems that we need to address before it makes sense to discuss individual items of a research agenda on a more detailed level.

The existing academic structures do not sufficiently support research in the area of business aspects and product management of software-intensive products

One of ISPMA's objectives from the start almost 10 years ago was the establishment of software product management as a discipline of its own on both the academic and industry side. While we have made a lot of progress on the industry side, we have failed on the academic side. The vast majority of computer science faculties and economic faculties do not want to deal with this research area (exceptions granted). This is a problem all over the world despite the extreme importance of the area for more and more industries. We may come up with the most wonderful research agenda, but that will not help if there are not enough researchers to do the work. One of the results of this seminar needs to be a manifesto to change this situation in academia, maybe with political and/or industry help.


Our terminology is not sufficient and seriously lacking in relation to what is happening in this area on the industry side

It is a core responsibility for any academic discipline to create and standardize a domain-specific language. Our terminology is seriously lacking. The description of this Dagstuhl seminar is a case in point. Why do we use the term “software production”? While the implied analogy with the manufacturing industry may be politically helpful, we all know that it is semantically misleading. In manufacturing “production” does not include the development of the product as a “type”, but only the creation of the physical “instances”. That is fundamentally different from what we (probably) mean by software production. When we talk about “software producing organizations”, do we really only mean open source consortia and commercial software product companies (like the text says), or do we

follow the broader semantics of the term which includes companies that produce software for software-intensive products (the text does mention Tesla and Volkswagen), as well as corporate IT organizations and professional service companies who develop custom software? We need to start an initiative to develop a state-of-the-art terminology for our field.

3.16 Position Statement

Thomas Kude (ESSEC Business School – France, kude@essec.edu)

License  Creative Commons BY 3.0 Unported license
© Thomas Kude

As a result of digitization, questions related to technology have become ubiquitous. While previously mostly dealt with in IT departments and tech companies, almost all organizations are now seeking guidance as to what capabilities are needed in a digital world and to what extent and how these organizations should transform into digital businesses. These developments provide ample opportunities for research in technology-related fields, such as information systems and software engineering, to make important contributions.

To do so, a sociotechnical view on digitization is needed. Currently, there seems to be a tendency to give primacy to technology alone, e.g., in the context of machine learning. Notwithstanding the undisputable benefits resulting from technological advances, there are indications that digitization reinforces the need for carefully establishing systems comprising technology, individuals or groups of individuals, as well as related activities.

In my current research, I take such a sociotechnical perspective to study different aspects of digital business at different levels of analysis. For example, I examine the governance of platform ecosystems in different empirical contexts. Some of my recent work was set in the context of enterprise software platforms and studied the motivation of complementors to join an ecosystem as well as governance practices and knowledge boundaries between platform owners and complementors. Some other recent work examined mobile platforms and the governance moves of platform owners along with consequent behaviors of complementors.

As another example, I study agile software development teams. Relying on survey studies and participant observations, I examine the implications of agile development practices, such as pair programming and code reviews, on the performance of software development teams. My focus is on the teamwork factors through which these effects are carried. Recent work includes studies related to the shared and distributed cognition and the backup behaviors among developers.

3.17 Position Statement

Alexander Mädche (Karlsruher Institut für Technologie – Germany, alexander.maedche@kit.edu)

License  Creative Commons BY 3.0 Unported license
© Alexander Mädche

The digital transformation of businesses and society makes most of today's organizations to some kind of software producing organizations (SPO). Driven by accelerating internal and external digitalization, organizations develop and deploy software in order to increase productivity, extend and enrich existing products and services as well as create entire new business models.

In my research I specifically focus on **Information Systems Engineering (ISE) following a socio- technical paradigm**. My research topics are allocated in the domain of scaling software producing organizations. Specifically, I look into three major fields: 1) data-driven ISE, 2) user-centered ISE, and 3) individuals and teams in ISE.

First, the basic idea of following a more **data-driven approach** to ISE goes to back to my PhD project on “Ontology Learning for the Semantic Web”. There, I investigated methods and techniques in order to semi-automatically construct ontologies from existing unstructured and structured data sources. In the last years, I also looked into questions of requirements elicitation following a semi-automated mining approach (requirements mining) as well as the semi-automatic construction of business models (business model mining) from structured data of organizational information systems. Second, I look into **user-centered ISE** because I strongly believe that usability and UX should be much more valued and emphasized by software producing organizations. Therefore, I investigate the role of design techniques as well as user involvement and participation in (agile) software development processes. Third, I’m interested on the social side in the form of **individuals and teams in ISE**. In my research group, we did carry out a number of empirical studies in this context, e.g. on age stereotypes in agile teams, emergence of team agility, coordination in large-scale agile software development. Recently, we leverage NeuroIS concepts (e.g. physiological signals, eye-tracking) in order to capture affective- cognitive states of developers and on this basis adapt the work environment. E.g. we are currently running a field study in cooperation with SAP SE in order to measure the flow state of developers in a SCRUM team and on this basis to intelligent adapt IT-mediated interruptions at the work place (e.g. emails).

3.18 Challenges in Software Ecosystems and Product Development

Efi Papatheocharous, (RISE SICS – Sweden, efi.papatheocharous@ri.se)

License © Creative Commons BY 3.0 Unported license
© Efi Papatheocharous,

The German computer science pioneer Karl Steinbuch in 1966 remarked: “In a few decades time, computers will be inter-woven into almost every industrial product.” The increasing prevalence of software ecosystems and platforms today calls for the ability to **augment solutions and support an emerging portfolio of leading technology solutions and trends**. It is unquestionable to design or use any software technology without taking into account digitalisation trends the emerging technological innovations (e.g., Big Data, Internet of Things, Systems of Systems) and without considering standing on the shoulders of a multitude of layers of platforms and ecosystems.

In our research we investigate efficient ways to organise and carry out product development in software ecosystems with the target to satisfy mutual and conflicting requirements from the involved parties.

This led to the formulation of the overall research questions (RQs):

- RQ1. What are the implications on the business models of the different actors, when moving from a traditional supply chain to a dynamic SECO?
- RQ2. What are the options for improved design of product architectures to handle the contradictory requirements of openness, flexibility and dependability, and to allow efficient product line management?


We identified challenges with respect to 3 categories: a) organizational, b) technical, and c) business and use a schema to conceptualise an ecosystem for Federated Embedded

Systems encompassing of four layers: actors, business processes, services and components. We described in an explorative case study (based on interviews with 15 senior staff members at 9 companies related to Embedded Systems) our findings mapped according to the Business Model Canvas (BMC) to highlight the interrelated parts and characteristics of the domain. Openness in SECO was evaluated in 7 companies including 8 practitioners taking into account their practices and methods.

Moreover, we target efficient and informed architecture formulation through the selection of existing components and services, and fast architectural adaptations which is crucial for companies' success, with a systematic approach in the decision-making process with respect to components, services and platforms.

3.19 Elements of Platform-based Ambidexterity: An Empirical Investigation

Balasubramaniam Ramesh (Georgia State University – USA, bramesh@gsu.edu)

License  Creative Commons BY 3.0 Unported license
© Balasubramaniam Ramesh

The notion of platforms has gained significant attention in both research and practice in information systems. Platform based approaches range from developing a family of products that address common and variable needs of a market to platform-based ecosystems that include open platforms and multi-sided markets. Much of the prior work on platforms focuses on challenges involved in platform-based development and elements of managing platform ecosystems. Platforms have been considered to play a role in facilitating organizations to handle the needs of various market segments efficiently, thereby supporting exploitation. Platforms have also been considered as a driver of innovation, a way to support organizations in their explorative endeavors. We argue that the notion of platforms can be leveraged in developing an approach that will help organizations simultaneously achieve both exploitation and exploration.

Organizations have long recognized the need to address tradeoffs when faced with constraints in meeting conflicting demands. Extensive research on organizational ambidexterity highlights antecedents and practices that play an important role in enabling organizations to balance exploration and exploitation. We posit that platform-based approaches can help organizations achieve ambidexterity.

This leads us to our key research question: “How can organizations leverage the notion of platforms to achieve ambidexterity?” Through a multi-site case comparative case study, we answer this question by identifying specific aspects of a platform-based approach to achieve ambidexterity. Our framework brings together the three important elements of a platform-based approach – development of product platforms, development of process platforms, and development of value-based platforms. We outline various elements of organizational context that drive the different aspects of a platform-based approach. Our findings detail how various factors such as the structure of the organizational units, flexibility of processes, environmental constraints faced, commonality and uniqueness of market needs, risk propensity of the platform developer and the other stakeholders, and culture/value that shape the development of platform based approach shape the ability of the organization to achieve ambidexterity.

3.20 Minimum Viable Products – The Road Ahead

Guenther Ruhe, (University of Calgary – Canada, ruhe@ucalgary.ca)

License © Creative Commons BY 3.0 Unported license
© Guenther Ruhe,

The dynamics of software products has enforced changes in the way products are developed. “Minimum Viable Products” stands for the development of products being viable to the users and customers, but are minimum in terms of the effort and functionality invested. The idea is to accelerate customer feedback not only early, but with minimum effort. Independently, “Technical Debt” is happening in all forms of conscious and non-conscious compromises done in the process of developing a product (version), deviating from what is understood the process “should be” in comparison to how the process actually.

There are numerous research questions around MVP. They relate to questions like:

- How to define experiments being the backbone for a MVP variant?
- Do we run MVP experiments concurrently or just incrementally?
- How often do we change MVP’s?
- Where the inspiration for a specific MVP comes from?
- How sensitive the definition of MVP’s is to different groups of stakeholders?
- How far are features of MVP’s implemented and evaluated to make a decision for their inclusion in future products?
- How many new features are accommodated in a MVP?

The seminar discussed some of these questions and their practical relevance from an industry perspective.

3.21 Only few can be platform owners

Kari Smolander (Lappeenranta University of Technology – Finland, kari.smolander@lut.fi)

License © Creative Commons BY 3.0 Unported license
© Kari Smolander

There is a growing interest on software-based platforms and platform economy and much knowledge has been collected on platform ecosystems and their governance. However, there are fewer attempts to investigate the enterprises that are not dominant players in the platforms, but need to integrate to various platforms to sustain or extend their business capabilities.

In the digital economy, integration has become more important than ever. The emerging technologies, such as the Internet of Things and Big Data, strongly emphasize integration. Improving the efficiency and responsiveness by integrating information systems within and outside the company is unavoidable in the modern collaborative business environment. Enterprise applications and systems can no longer exist as stand-alone entities, but instead they must interact with other information systems inside and outside the company walls. Integration has become a necessity to satisfy customers.

Our activities and transactions are increasingly happening in software-based platforms and ecosystems, such as Facebook, Google, WeChat, AliPay and various industry-specific platforms, that are not directly controllable by single enterprises. We lack understanding of how enterprises should approach and manage their integration to software-based platforms. This integration is often a necessity, because these external platforms increasingly guide the

actions of customers and business partners of enterprises. There is a wealth of studies on platforms themselves and their evolution, platform governance and leadership of keystone players like Google, Amazon, and Apple and on platform creation strategies, but we lack in-depth knowledge of integration to platforms. The integration problems of enterprises that are not dominant players in the platforms have received only little attention. Still, they are the vast majority of enterprises and they need to integrate to various platforms to sustain or extend their business capabilities. This integration can bring business and security risks and a platform lock-in is often a consequence. There is an urgent need to study this, since only extremely few can be platform owners.

3.22 Data for Performing Research on Software Business, Platforms, and Ecosystems

Diomidis Spinellis (Athens University of Economics and Business – Greece, dds@aueb.gr)

License  Creative Commons BY 3.0 Unported license
© Diomidis Spinellis

An identified research challenge in the area of software business, platforms, and ecosystems research is the lack of easily accessible research data [1]. Scientists require these data first, to gain insights regarding theory of software production processes, and second, to empirically validate proposed theories. The lack of data can be addressed by collecting, processing, curating, and making available suitable data sets.

Businesses are understandably reticent about sharing data concerning their processes, operations, outcomes, and strategy. Yet, many types of data are either already available or can be obtained from openly accessible sources, such as corporate web sites, software forges, and app stores.

- Data and metadata of a company’s web presence as well as web log data can reveal details regarding its software development processes and the adoption of technology platforms [2].
- App stores can be mined to gather information regarding products, reviews, and advertising networks [3].
- Open source software forges, such as GitHub [4], can provide data regarding corporate open source software strategy, contributions, and development processes [5].
- Continuous integration [3], Q&A forums [3], and code review servers [6] can be further mined to extract more detailed product and process data.

The outlined data can provide insights on how diverse companies deal with technology challenges, such as software complexity, security, and reliability, the extent and dynamics of platform and technology adoption, and the performance of specific business models. As this approach leans toward empiricism rather than rationalism, it requires disciplined data analysis protocols, such as registered reports [7] and close alignment with theory building.

To move forward as a community in the proposed directions we must collect, curate, and publish existing data sets;⁹ determine areas where new data are required; encourage the development, release, and publication of new data by recognising their scientific value; extract data from untapped data sources; and build upon the data showcasing their utility through advances in theory and practice.¹⁰

References

- 1 Pekka Abrahamsson, Jan Bosch, Sjaak Brinkkemper, and Alexander Mädche. Dagstuhl seminar: Software business, platforms, and ecosystems: Lifecycle challenges of software producing organizations. Available online <https://materials.dagstuhl.de/files/18/18182/18182.SWM.Other.pdf>, 2018.
- 2 Diomidis Spinellis and Vaggelis Giannikas. Organizational adoption of open source software. *Journal of Systems and Software*, 85(3):666–682, March 2012.
- 3 Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. Answering questions about unanswered questions of Stack Overflow. In *MSR 2013: 10th Working Conference on Mining Software Repositories*, pages 97–100. IEEE Press, 2013.
- 4 Georgios Gousios and Diomidis Spinellis. GHTorrent: Github’s data from a firehose. In Michele Lanza, Massimiliano Di Penta, and Tao Xie, editors, *9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 12–21. IEEE, June 2012.
- 5 Diomidis Spinellis. Tools and techniques for analyzing product and process data. In Tim Menzies, Christian Bird, and Thomas Zimmermann, editors, *The Art and Science of Analyzing Software Data*, pages 161–212. Morgan-Kaufmann, 2015.
- 6 M. Mukadam, C. Bird, and P. C. Rigby. Gerrit software code review data from Android. In *MSR 2013: 10th Working Conference on Mining Software Repositories*, pages 45–48, May 2013.
- 7 Christopher D Chambers. Registered reports: A new publishing initiative at Cortex. *Cortex*, 49(3):609–610, 2013.

3.23 Software Ecosystems Research Agenda Update

Slinger Jansen (Utrecht University – the Netherlands, slinger.jansen@uu.nl)

License  Creative Commons BY 3.0 Unported license
© Slinger Jansen

Increasingly, software producing organizations collaborate in networks that have become known as software ecosystems [1]. The intricate structures of platforms upon platforms [2] enable rapid innovation like never before. In our research laboratory, we explore how these platforms collect knowledge about the platform itself, the applications running on it, and the end-users that make use of these applications. Through examples and case studies is shown that software operation knowledge in software ecosystems is essential for creating better software, happier users, and more productive developers [3].

In research, the term ‘ecosystem’ is popular. Terms such as the sales force automation ecosystem, enterprise resource planning ecosystem, and artificial intelligence ecosystem are thrown around freely. This is a double edged sword: the term becomes increasingly popular

⁹ See e.g. <https://github.com/awesomedata/awesome-public-datasets> and <https://github.com/dspinellis/awesome-msr>.

¹⁰ The project associated with this work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 732223.

but it also diffuses the meaning of the term in its context. Simultaneously, the term “software ecosystem” is sometimes attacked for being too specific, when terms such as business or digital ecosystem suffice. However, with the word ‘software’ we emphasize the phenomena caused by the critical component of software underlying these ecosystems, thereby scoping our work around themes such as software business, software engineering, software as a service, and open source.

One of the big challenges of the field is its multi-disciplinarity. It includes work about open source from software engineers, works about automotive platforms from management and information scientists, and works about visualizations from computer scientists. We reiterate that this domain is relevant from different perspectives, and should thus be considered multi-disciplinary.

References

- 1 Slinger Jansen, Eko Handoyo, and Carina Alves. Scientists’ needs in modelling software ecosystems. In *Proceedings of the 2015 European Conference on Software Architecture Workshops, ECSAW ’15*, pages 44:1–44:6, New York, NY, USA, 2015. ACM.
- 2 S. Brinkkemper, I. Soest, and S. Jansen. Modeling of product software businesses: Investigation into industry product & channel typologies. *Information Systems Development*, pages 307–325, 2009.
- 3 Joost te Molder, Ben van Lier, and Slinger Jansen. Clopenness of systems: The interwoven nature of ecosystems.

3.24 AI and Software Business

Pasi Tyrväinen (University of Jyväskylä – Finland, Pasi.Tyrvaainen@jyu.fi)

License  Creative Commons BY 3.0 Unported license
© Pasi Tyrväinen

Changes in software business are driven mainly by demand for new applications, development of hardware technology, development of new reusable software artefacts (platforms etc.) as well as the need to combine the previous three into solutions with business value. The short history of software business has shown that the variation in each of these drivers requires effort of skilled software personnel to capture the functionality into form of a software artefact. The promise of reuse, end-user programming, model driven software development, components, and code generation have not been able to remove the need of human intelligence in the process and the volume of software development effort has continuously had a growing trend. So far, the majority of business activity related to software has been in bespoke software tailored for enterprise use regardless the grooving emphasis on platforms and standardized products delivered as a service over the Internet.

Recently, there has been public discussion on the possibility to use artificial intelligence (AI) as a means to reduce or even replace the human effort in software engineering. However, as majority of the AI effort (55%) is focusing on analytics rather than symbolic methods needed for increasing the automated part of software creation process, this statement seems not to be valid. Further, when recalling the AI solutions to operate dominantly under closed world assumption, the impact of such systems is likely to remain in the phases starting from requirements specification while most of the drivers for new software dominantly demand major software engineering effort in defining the need and elaborating it to a requirements

specification sufficient for automated code creation. Thus AI is unlikely to respond to the three change drivers mentioned above.

3.25 High-speed and Sustainable Development of Software Startups

Xiaofeng Wang (Free University of Bozen-Bolzano – Italy, xiaofeng.wang@unibz.it)

License © Creative Commons BY 3.0 Unported license
© Xiaofeng Wang

“Software is eating the world”, and software startups are disrupting the world. Uprising startups, such as Airbnb, Uber and Spotify, are extremely active and volatile elements in the ecosystem of the global economy. However, building a successful software startup is extremely difficult and the failure rate is strikingly high. Based on the research of our software startup research network, we have identified three grand challenges that software startups and ecosystems are facing:

- How to build a software startup in a high-speed and sustainable manner?

Specific challenges come from various directions, including software development, entrepreneurial team building, business model definition and fund raising. Product and market related issues demand that a software startup acts and reacts with high-speed in extreme uncertainty, whereas other issues, e.g., building entrepreneurial teams, or maintaining acquired customer base, mandate that a startup works in a sustainable manner as it pursues its grand visions. These two aspects are not always compatible and need to be considered simultaneously and balanced properly. In addition, early stage startups are different compared to those that are scaling up. Different knowledge and practices are needed to succeed through different stages. How to maintain vision, passion and innovation momentum, typically the driving forces to initiate a startup, in the later stages of the startup lifecycle, is a challenge faced by software startups. Continuous innovation is therefore essential.

- How to provide better support to software startups?

As startups never live in isolation, their survivability depends on the startup ecosystems they are located in. Organizations such as business incubators, accelerators, business angels and venture capitalists positively influence their chances to survive and grow. However, these ecosystem players need to allocate their resources effectively by figuring out what supporting measure will have the greatest impact for each particular startup, considering its stage of development and individual strengths and weaknesses. Investors equally need to be able to reliably assess the investment risk and possible return. Intermediaries have to deal with a wave of new startup support and development tools such as crowdfunding platforms which are becoming an increasingly important source of funding. Initial Coin Offering (ICO), a phenomenon closely linked to blockchain technology, is another emerging fundraising mechanism for startups at very early development stages. Decentralization and deeper personalization are new types of support needed by software startups. However how to utilize these mechanisms to support software startups is yet to be understood fully.

- How to better train current and future software startup founders?

Many software startup founders lack necessary knowledge to initiate their startup journey and blindly follow only gut feelings and/or a trial-and-error approach. No validated learning is obtained and accumulated to guide their practice. On the other hand, training and education offerings are diverse across different organizations and institutes, making it difficult to share best pedagogical practices.

These challenges are intertwined and have to be tackled collaboratively by all startup ecosystem stakeholders. Besides, there are more fundamental questions that the researchers interested in software startups need to answer before diving into the battle against these challenges. For example, what are software startups exactly? Are they fundamentally different from other types of startups? what research disciplines are relevant to obtain necessary and updated knowledge from? These fundamental questions need to be answered before we could start tackling the challenges listed above.

3.26 Understanding new development trends, exploring large data, and pushing the boundaries of innovation

Karl Werder (University of Cologne – Germany, werder@wiso.uni-koeln.de)

License  Creative Commons BY 3.0 Unported license
© Karl Werder

Software is an important element of the digital transformation. Software producing organizations (SPO) lead the way in product innovations and new ways of working. Let me share my elaborate on my perspective towards software producing organizations:

First, SPOs leading **new development trends**. Having XP, Scrum, or DevOps, SPOs continuously challenges their operating modes and find new innovating techniques to conduct development and design activities. These, as in the case of agile development, inspire industries beyond the software industry, partly due to their proven benefits to both, management and developers [1], and due to their applicability to all organizational sizes, from startups to large international enterprises.

Second, SPOs providing grounds for highly impactful research insights based on **large data**. Using a central repository is a standard tool in SPOs. These provide deep insights into the organizations development processes and enable researchers to answer questions that previously have been left unanswered [2]. Using this rich data source enables researchers from IS and SE domain to provide research contributions beyond their own field of research towards other parts of social sciences, such as project management, team research, leadership, and human resource management. It also helps scholars to better understand large scale software development practices.

Third, SPOs pushing the boundaries of **innovation**. A key assumption of innovation research is that innovation is a well-bounded phenomenon, focusing the investigation of fixed products. However, given software ecosystems and technological innovations such as block chain, innovation is much more fragmented and less defined. Open innovation facilitates the collaboration of manifold people without limitation to a product or timeframe, (e.g. [3]), as in contrast to (new) product development.

In my research, I focus on enhancing understanding the software development process and the people involved in it.

References

- 1 Karl Werder and Alexander Maedche. Explaining the emergence of team agility: a complex adaptive systems perspective. *Information Technology & People*, 31(3):819–844, 2018.
- 2 Karl Werder. The Evolution of Emotional Displays in Open Source Software Development Teams: An Individual Growth Curve Analysis. In *2018 IEEE/ACM 3rd International*

Workshop on Emotion Awareness in Software Engineering (SEmotion), Gothenburg, SE, 2018. ACM.

- 3 Karl Werder, Remko W. Helms, and Slinger Jansen. Social Media for Success: A Strategic Framework. In *Pacific Asia Conference on Information Systems 2014*, page Paper 92, Chengdu, PRC, 2014. AISeL.

3.27 Improving handling business model changes for software-intensive organizations

Krzysztof Wnuk (Blekinge Institute of Technology – Sweden, krw@bth.se)

License © Creative Commons BY 3.0 Unported license
© Krzysztof Wnuk

Software-intensive companies are recently undergoing significant transformations and are struggling with the alignment of business and technology change. Until recently, these companies handled increasing size and complexity by: 1) clearly distinguishing between the planning and realization layers for company strategy, product portfolios and individual products; and 2) handling change mainly in the realization layer and ensuring that the planning layer remains reasonably stable. Frequent changes into the realization layer were efficiently handled by various engineering paradigms and principles, e.g. software architecture, Software Product Lines, variability and configuration management, just to name a few.

Whatever change arriving to the strategy, portfolio or product level could be either earlier anticipated or received sufficient accumulation time in the realization layers, both of which helped to handle substantial size and complexity of software-intensive products.

However, the core of the recent transformation is that the speed of changes in the planning layer increases substantially and, in many cases, reaches the speed of changes in the realization layer.

A primary driver for this transformation is the digitalization of the business environment. For example, at the product level, the introduction of agile development and user communities, user groups for high-valued customers, and similar forums allow customers to interact directly with product development. Such frequent interaction transforms much of the traditional product road-map planning work, into a continuous software release including both feature-based upgrades and bug fixes. At the Product Portfolio level, the need to innovate and increase the value creation drives a transition towards Product Service Systems (PSS) and use-models.

New services are mixed with software products and re-usable components to create new product and solution offerings that can either be delivered as a cloud or as a more traditional product sale. Finally, at the strategy level, companies create or get engaged in various business ecosystems where they reinvent the value and work together with other ecosystem stakeholders to co-produce value. The same pattern is seen in most industries today, e.g., Porsche launch an open innovation platform aiming at taking the lead in an electric car business ecosystem, similarly as Amazon once turned their struggling online bookstore into a global shopping ecosystem by launching an online, cloud-based Web Service platform (AWS) in 2002. By the re-launch of AWS in 2006, with their Elastic Compute Cloud (EC2), Amazon popularized the term 'cloud computing' and became an active leader in a new IT industry. Similar examples are also recognized by Bharadwaj et al. as they coin the term digital business strategy as the fusion of business strategy and IT strategy. They point at limitations

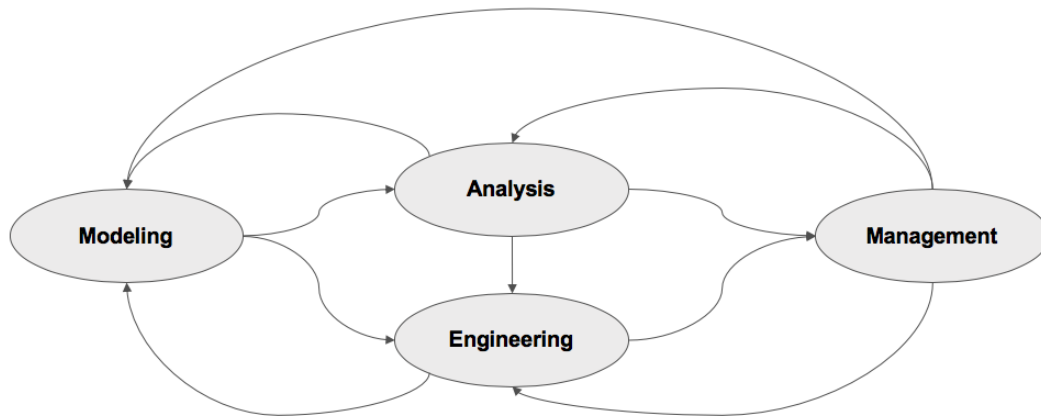


Figure 1 We identify four domains in which the software ecosystems research challenges can be categorized.

of traditional business models as “we need richer models that delineate inter-dependent ecosystems that evolve more rapidly than what we have seen in traditional settings”.

4 Working Groups

4.1 Working Group on the Software Ecosystem Research Agenda

Paul Grünbacher (Johannes Kepler Universität Linz – Austria, paul.gruenbacher@jku.at)

Jens Förderer (University of Mannheim – Germany, foerderer@uni-mannheim.de)

Zhi Jin (Peking University – China, zhijin@pku.edu.cn)

Samuel Fricker (FHNW University of Applied Sciences and Arts Northwestern Switzerland – Switzerland, samuel.fricker@fhnw.ch)

Rahul Basole (Georgia Institute of Technology – USA, basole@gatech.edu)

Slinger Jansen (Utrecht University, the Netherlands, slinger.jansen@uu.nl)

License © Creative Commons BY 3.0 Unported license

© Paul Grünbacher, Jens Förderer, Zhi Jin, Samuel Fricker, Rahul Basole, Slinger Jansen

Ten years after the Software Ecosystems research agenda at ICSE [1] it is time to take score and establish a research direction for the next decade. The domain has grown significantly, both in publications and interest, and overall there are signs of maturation of the domain. As the community is increasing its research effort, it is relevant to articulate themes that give direction to the research, avoid redundancy, and provide novel research avenues. In this report we express themes, challenges, research questions, and propositions, based on ten years of literature and research agendas in the field.

A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. Software ecosystems are pervasive and software producing organizations increasingly realize that it is the ecosystem that makes them and their technologies successful [2].

Digital business has become an essential pillar under most economies and it has been a driver of innovations for several decades. The introduction of new technologies and convergence of the Internet of Things, cloud technologies, and artificial intelligence, lead to a

	Definition	Issues	Recommendations
Modeling	Act of defining the primitives, hierarchies, the granularity, actors, structure, arrangement, behavior in the ecosystem.	No common language	harmonization of existing languages
		No scalable modeling method	Build a typology/classification of ecosystems and the actors involved
		Insufficient understanding of tipping points and genesis events	Advance a hierarchical view of platforms, develop a hierarchical modeling approach
		Little understanding of dynamics, adaptive, and emergent behavior	Simulate evolutionary models, perhaps using independent agents
		Insufficient understanding of ecosystem boundaries	Predictive modeling
Analysis	Define, collect, analyze metrics	modeling of human and crowd behaviour	Develop a model of the boundaries involved in an ecosystem (parallel to the onion model)
		metrics interpreted differently (what is centrality etc) --> data-dependent, no consensus regarding the interpretation	leverage agent-based modeling and innovation/diffusion techniques
		Situational awareness and sense-making	converge on measures, go to a standardization organization, adopt and compare existing measures, bring communities together and formulate recommendations, have regular exchanges
		Analysis of metrics over time	Conceptualize ecosystem health
		Data access and availability	Create visual representations at all levels
Engineering	The definition and application of governance mechanisms for software ecosystem development.	lack of design knowledge for starting SECOS, difficulty of studying ecosystems, limited number of attempts for launching an ecosystem	Leverage data science, machine learning techniques
		insufficient understanding of architectures that enable ecosystems	Build a library of (temporal) SECO knowledge
		Lack of understanding of the relationship between ecosystem health and technology adoption	- derive principles, e.g. from existing examples - simulation techniques and Quasi experiments for SECOS - allow in-laboratory research of SECOS - a catalogue of knowledge discovery mechanisms
Management	Orchestration of the ecosystem	Insufficient insight into value flows and human centered design	Create a software architecture pattern catalogue for software ecosystems (e.g., extension mechanisms)
		Lack of understanding regarding the theoretical mechanisms by which organizational actions/governance mechanisms (competition, cooperation, knowledge sharing etc.) shape ecosystem outcomes	Derive principles and theories about adoption and health
		Insufficient insight into renewal of ecosystems	Create a value model for ecosystems Study Community/ecosystem canvas Empirically test cause consequence models Create a collection of governance mechanisms

■ **Figure 2** During the Dagstuhl event these challenges were identified and categorized over the four domains.

myriad of new possibilities, but require an ecosystem approach for extensive adoption. These technologies are rapidly adopted, in large part due to the “ecosystemification” of the digital business; software producing organizations depend on each other to enable faster adoption of new technologies supplied by new entrants in the market.

Society, organizations, and economies are experiencing and anticipating fundamental changes that are shaped, embedded, and influenced by ecosystems. Ecosystems are social, technical, and economic systems that are large, multi-level, complex, dynamic, adaptive, emergent and global in nature, and concern a wide range of stakeholders (managers, policy-makers, and society), each with different perspectives and incentives. An interesting finding is that ecosystems cannot be created, but must be cultivated and fostered.

The complexity in scale, scope and dynamics makes systematic modeling, analysis, engineering, and management challenging. It requires multi-disciplinary perspectives in research, such as computer science, economics, management, information systems, innovation sciences, engineering and policy. Given its economic and societal relevance, successful ecosystem research requires collaboration by scholars, practitioners, and individuals. The value and impact of engineered ecosystems is manifested through mobilization, participation, and facilitated collaboration enabling growth, innovation, and welfare.

One of the big challenges of the field is its multi-disciplinarity. It includes work about open source from software engineers, works about automotive platforms from management and information scientists, and works about visualizations from computer scientists. We reiterate that this domain is relevant from different perspectives, and should thus be considered multi-disciplinary.

The overview of challenges in Figure 2 functions as an inspiration for a future research agenda, which is currently under development.

References

- 1 S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 187–190. IEEE, 2009.
- 2 Slinger Jansen, Michael A Cusumano, and Sjaak Brinkkemper. *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar Publishing, 2013.

4.2 Working Group on Software Business and Technology Lifecycle Artifacts

Sjaak Brinkkemper (Utrecht University)

Armin Heinzl (University of Mannheim)

Robert Heinrich (Karlsruhe Institute of Technology (KIT))

Alexander Maedche (Karlsruhe Institute of Technology (KIT))

Kari Smolander (Lappeenranta University of Technology)

License © Creative Commons BY 3.0 Unported license

© Sjaak Brinkkemper, Armin Heinzl, Robert Heinrich, Alexander Maedche, Kari Smolander

In Software-intensive Business many different artifacts along the software lifecycle are created. This covers both, business-oriented artifacts such as business models, roadmap, user stories or business process models as well as technology-oriented artifacts, such as technical architecture diagrams, class diagram or test cases is created.

Artifacts of the business domain depend on artifacts of the technology domain and vice versa. Furthermore, there are dependencies between artifacts used along the lifecycle of development, deployment and operations of software-intensive systems. Understanding and making these dependencies explicit in the entire development and management of large-scale software systems is important. However, currently there neither a classification of relevant artifacts nor an explicit description of their dependencies along the entire lifecycle available. Thus, there is no possibility to trace links between the artifacts and there are no comprehensive tools supporting navigating through different artifacts and propagating changes from one artifact to another. This leads to limiting focus on a small subset of artifacts and neglecting substantial side effects between the artifacts.

The working group addressed this important issues and suggested a set of activities in order to solve this problem. Specifically, an initial classification of existing artefacts (business and technology) covering the whole life-cycle was created. Furthermore, ideas were discussed on how to apply the classification, e.g. for monitoring development and operations, systematic life-cycle data collection, as well as interconnecting, tracing and optimizing of all artefacts. In a follow-up activity, the goal is to come up with a first conceptualization including a classification of artefacts and their dependencies in the field of Software-intensive Business. The conceptualization will formally represented by (partial) metamodels and their composition. Initial prototypical tool support for visualizing and navigating through the conceptualization will be provided.

4.3 Working Group on Software-Intensive Business Research: Definition and Roadmap

Karl Werder (University of Cologne, Germany, werder@wiso.uni-koeln.de)

Sjaak Brinkkemper (Utrecht University – the Netherlands, s.brinkkemper@uu.nl)

Jan Bosch (Chalmers University of Technology – Sweden, jan@janbosch.com)

Michael A. Cusumano (MIT Sloan School of Management – USA, cusumano@mit.edu)

Georg Herzworm (University of Stuttgart – Germany, herzwurm@wius.bwi.uni-stuttgart.de)

Helena Holmström Olsson (Malmö University – Sweden, helena.holmstrom.olsson@mau.se)

Sami Hyrynsalmi (Tampere University of Technology – Finland, sami.hyrynsalmi@tut.fi)

Hans-Bernd Kittlaus (InnoTivum Consulting & ISPMA (International Software Product Management Association) – Germany, hbk@innotivum.de)

Thomas Kude (ESSEC Business School – France, kude@essec.edu)

Tiziana Margaria (University of Limerick – Ireland, tiziana.margaria@ue.ie)

Efi Papatheocharous (RISE SICS – Stockholm, SE, SE)

Diomidis Spinellis (Athens University of Economics and Business – Greece, dds@aueb.gr)

Pasi Tyrväinen (University of Jyväskylä – Finland, Pasi.Tyrvainen@jyu.fi)

Xiaofeng Wang (Free University of Bozen-Bolzano – Italy, xiaofeng.wang@unibz.it)

License © Creative Commons BY 3.0 Unported license

© Karl Werder, Sjaak Brinkkemper, Jan Bosch, Michael A. Cusumano, Georg Herzworm, Helena Holmström Olsson, Sami Hyrynsalmi, Hans-Bernd Kittlaus, Thomas Kude, Tiziana Margaria, Efi Papatheocharous, Diomidis Spinellis, Pasi Tyrväinen, Xiaofeng Wang

In today's digital environment companies are forced to participate in a process often coined digital transformation. As a result, companies expect to stay relevant and harness digital technologies for their competitive advantage and sustainable value creation. A central element of this transforming process is the software that companies develop, purchase or customize in order to support their business. These challenges stretch beyond the information and technology industry, as businesses use digital technology to compete in traditions industries. Popular examples are omnipresent, with Uber revolutionizing the taxi industry, AirBnB forcing new legislations to protect the hotel industry, and Spotify becoming a single source for music with a monthly subscription model. The Dagstuhl Seminar “Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research” organized by Pekka Abrahamsson, Jan Boch, Sjaak Brinkkemper, and Alexander Mädche took place from 29th of April until 02nd of May. The seminar's objectives were i) to strengthen cross-community research efforts, ii) to increase accessibility of research data and results, iii) to exchange on current and future research developments and discussions, iv) to initiate project ideas between scholars and with industry that evolve into project proposals.

4.3.1 Toward a definition

A Software-intensive Business creates, captures, and delivers value through digital technologies. Software-intensive Businesses create value through the development of new software technologies. When operating a platform, they often capture value through their established network of partner. When a software is shipped to and operated by a customer, the value is delivered. The academic community around Software-intensive Businesses investigates two aspects, i) arrangements and methods, and ii) responses to environmental changes. When investigating arrangements and methods, the community distinguishes between phenomena within and between organizations. Example organizations are software firms, start-ups, data

businesses and other Software-intensive Businesses. Within such organizations, product management, business models, agility, and DevOps are example arrangements and methods of interest. Between organizations, ecosystems, platforms, and OSS communities can be the subjects of investigation.

Given the manifold environmental changes a Software-intensive Business is subject to, the community researches three sources of change. First, the general and overarching trends and changes stemming from political, economical, societal, technological, environmental, and legal changes. Second, changes in their competitive environment, which may stem from a competitor dominating the market, employee shortages, or from characteristics of an industry segment. Third, customer trends, such as the digitalization, consumerization of information technology, and changing values lead to changing requirements. Hence, we formulate the following definition for Software-intensive Business research:

*The scientific field of **software-intensive businesses** studies sustainable software-based value creation, capture, and delivery a) through **arrangements and methods** i) within organizations (e.g. product management, business models, agility) (e.g. established, software firms, startups, data businesses, other firms), and ii) between organizations (e.g. ecosystems, platforms, app stores, OSS communities) and b) in response to **environmental changes** related to i) political, economical, societal, technological (e.g. cloud, IoT), environmental, and legal (regulation, GDPR, IPR); ii) competitive environment (e.g. market dominance, employee shortage, industry segments); and iii) customer trends (e.g. digitalization, consumerization, values).*

4.3.2 Toward a research agenda

As a result of the Dagstuhl seminar, the participants identified a 3x3 focus matrix. The x-axis of the matrix depicts the unit of analysis, i.e., a software system, a human system, or an ecosystem. The software system can be investigated as a whole or in parts, such as component or modules. The human system refers to a software organization, a development project or team team, or an individual, such as a developer or user. The y-axis represents the value, the lifecycle stages, and enablers we need to understand. Innovation can stem from technological innovation, business-driven innovation, or design innovation by creation new means of human-computer interaction in order to create and deliver more value. Innovation can also relate to the speed in which ideas and features are turned around, often referred to time-to-X, such as time-to-market, time-to-release, etc.. Lifecycle refers to the different stages a Software-intensive Business can find itself. Beginning with its inception as a startup, progressing toward a mature company, managing its rich ecosystem or responding to a crisis and need for a transformation. Enablers are prerequisites that help us to better understand the impact or facilitation of different factors, that some may call success factors.

	Software System	Human System	Ecosystem
Value	Deliver more value to its users	Reduce time-to-market, time-to-release	Provide value through an established network
Lifecycle Stages	From prototype to stable release	From student to expert	From beta to market dominance
Enablers	Knowledge base and research platform	Trainings, workshops and phd courses	Data mining and data analytics

Using this matrix, we suggest six areas that require further research in the future: 1) definition and reuse of core concepts, 2) Software-intensive Business lifecycle, 3) future business models, 4) benefits of new technologies, 5) driving innovation, and 6) enablers that support these research trends.

- **Definition and reuse of core concepts:** In order to advance our understanding of the nomological net in the field, core concepts need to be identified and reused. The community investigates concepts, such as ecosystems, platforms, development methods and tools, and product management, to name only a few. While the investigation of new concepts explore new research avenues for the community, the difference to established and better understood concepts needs to be clear. When investigating established concepts, the community progresses towards a deeper understanding of such concepts, their antecedents, outcomes, and boundary conditions. Hence, more research is needed that reflects the status quo in regards to core concepts of the community and simultaneously suggests the quo vadis for the research efforts of the investigated concept.
- **Software-intensive business lifecycle:** historically, the community centered around the term software business. Hence, it is not surprising that research shed more light on the business related activities. These were often limited to well-established software businesses in order to better understand how these differ from other businesses. While we have a better understanding of the differences and unique characteristics of the software business, more research is needed that investigates different lifecycle stages of these businesses. Example are research into software start-up, the effective management of platforms, or the management of crisis situations.
- **Future business models:** Traditional business model focused on the sale of a software license and the corresponding service. While there has been a major shift in the sales of software products towards a service oriented approach, as for example through software as a service. Further business models have evolved. For example the case of Uber, being a provider of a digital platform that users' approach in order to be linked with a taxi driver nearby. These examples suggest that the business models of Software-intensive Businesses keep changing as they are reinventing themselves. Hence, more research is needed in order to understand the driving forces behind these transitions and sometimes pivoting processes.
- **Benefits of new technologies:** Software technologies evolve at a astonishing rate. Internet and mobile technologies facilitated an increasing access and utilization of software. On the one hand internet technologies facilitated the introduction and use of software as a service concepts in which software installations become obsolete. Mobile devices result in the omnipresence of information technology in today's lifestyles and continuous access to messaging, social media, and finance applications. Yet, more recently, technologies such as machine learning, artificial intelligence, big data, internet of things and blockchain have been introduced amongst others. More research is needed in order to investigate their role in managing Software-intensive Businesses.
- **Driving innovation:** Given the increasing rate of change, software-intensive businesses need to find new ways to collect data, analyse such data, and derive meaningful conclusions. In response, continuous experimentation has been suggested in which the software provider tests different version of the software over time in order to analyse the data and to understand what works best. Little do we know about continuous experimentation with software and its possible extension to other subjects, such as business related decisions. Hence, more research is needed.

- **Enablers that support the trends:** Given these research trends to not exist in a vacuum, more research is needed that investigates enablers supporting these trends. For example, we need to understand whether existing measurement instruments still apply to these trends, If not, what adjustments to we need to make in order to assure reliable measurements? How can we assure that the next generation of Software-intensive Business scholars have the right skills and tools to progress the research with the rigor and granularity needed to advance the field?

4.4 Working Group on Health Measurement of Open Source Projects and Ecosystems

Slinger Jansen (Utrecht University)

Paul Grunbacher (Johannes Kepler University)

Efi Papatheocharous (RISE, Sweden)

Diomidis Spinellis (Athens University of Economics and Business – Greece, dds@aueb.gr)

License  Creative Commons BY 3.0 Unported license

© Slinger Jansen, Paul Grunbacher, Efi Papatheocharous, Diomidis Spinellis

Open source projects and ecosystems can be studied due to the public availability of their data. The main reasons for studying this data is to collect operationalizable metrics that can be used for the improvement of the project or ecosystem. We can for instance use these metrics to do prediction, study adoption rates, and perform scenario modeling.

Presently, in literature, the reigning health factors that are acknowledged are Robustness, Productivity, Niche creation. It is also common to look at ecosystem health from two dimensions: the partner/network level versus the system/project level. Each dimension provides a unique perspective on open source health and enables improvement in a different manner: one focuses on the activity within the platform, whereas the other focuses on the activity outside of it.

Typically, in open source ecosystem health research the metrics are characterized along several axes: they are evaluated for availability, collectability, generalizability, comparability, user friendliness, etc. Examples of metrics are interactions between developers, clones, branches, and numbers of commits. We also find that metrics that are typically easy to collect are not very meaningful. Also, the need arises for a meaningful compact subset of metrics, instead of throwing the kitchen sink at evaluation projects. Also, we suspect that “typical” developer behaviors can be extracted from the correlations between different metrics. Finally, we find that the goal-question-metric approach is insufficiently employed in the study of the health of ecosystems.

One of the bigger challenges in assessing ecosystem health is the myriad of perspectives on ecosystems. For instance, we can look at network health versus economic health. Furthermore, ecosystems themselves are made up of ecosystems, and we need to establish beforehand what the best manner is of decomposing an ecosystem.

4.5 Working Group on Research Data for Software Intensive Business

Slinger Jansen (Utrecht University)

Diomidis Spinellis (Athens University of Economics and Business – Greece, dds@aueb.gr)

Krzysztof Wnuk (Blekinge Institute of Technology, Greece)

License © Creative Commons BY 3.0 Unported license
© Slinger Jansen, Diomidis Spinellis, Krzysztof Wnuk

One of the largest challenges in the Software-intensive Business domain is the collection of data for research purposes. Mostly, the data is generated by proprietary entities, who are already challenged with the task of making the data publicly available. Software intensive businesses typically also do not see the use in sharing their data, as they reveal information about products, teams, and persons.

In intense collaborations organizations tend to be much more willing to share their data with the researcher, and often also with the entire research community. There are different ways to achieve this. First, researchers must avoid strategic topics for the company, such as new product strategy and infighting. Secondly, people within companies love sharing success stories. Thirdly, it is generally easy to anonymize the case data. Fourthly, it is possible to set up consortia in a distinct market, which enables collaboration between the companies and provides researchers with a trove of data. Finally, researchers can use historical data that is no longer problematic for the organization. There are also open data sources, such as economic meta-data, app stores, open source repositories, code review servers, continuous integration servers, and testing servers that provide data.

As a community we must find ways to curate data and publish it to colleagues. Furthermore, we need to provide researchers with incentives to publish case studies.

Participants

- Pekka Abrahamsson
University of Jyväskylä, FI
- Rahul C. Basole
Georgia Institute of Technology –
Atlanta, US
- Jan Bosch
Chalmers University of
Technology – Göteborg, SE
- Sjaak Brinkkemper
Utrecht University, NL
- Christoph Bussler
Oracle Labs – Redwood Shores,
US
- Michael Cusumano
MIT – Cambridge, US
- Jens Förderer
Universität Mannheim, DE
- Samuel A. Fricker
FH Nordwestschweiz – Windisch,
CH
- Paul Grünbacher
Johannes Kepler Universität
Linz, AT
- Robert Heinrich
KIT – Karlsruher Institut für
Technologie, DE
- Armin Heinzl
Universität Mannheim, DE
- Mika Helenius
TIVIA – Espoo, FI
- Georg Herzwurm
Universität Stuttgart, DE
- Helena Holmström Olsson
Malmö University, SE
- Sami Hyrynsalmi
Tampere University of
Technology, FI
- Slinger Jansen
Utrecht University, NL
- Zhi Jin
Peking University, CN
- Hans-Bernd Kittlaus
Inno Tivum Consulting –
Rheinbreitbach, DE
- Thomas Kude
ESSEC Business School – Cergy
Pontoise, FR
- Alexander Mädche
KIT – Karlsruher Institut für
Technologie, DE
- Tiziana Margaria
University of Limerick, IE
- Efi Papatheocharous
RISE SICS – Stockholm, SE, SE
- Balasubramaniam Ramesh
Georgia State University –
Atlanta, US
- Guenther Ruhe
University of Calgary, CA
- Kari Smolander
Aalto University, FI
- Diomidis Spinellis
Athens University of Economics
and Business, GR
- Pasi Tyrväinen
University of Jyväskylä, FI
- Xiaofeng Wang
Free University of Bozen-Bolzano,
IT
- Karl Werder
Universität Duisburg – Essen,
DE
- Krzysztof Wnuk
Blekinge Institute of Technology –
Karlskrona, SE

