Will Computer Science become a Social Science?

Ingo Scholtes*, Markus Strohmaier†, Frank Schweitzer*
* Chair of Systems Design, ETH Zurich, Switzerland
† GESIS - Leibniz Institute for the Social Sciences, Cologne, Germany

February 28, 2017

When Tay - a Twitter chatbot developed by Microsoft - was activated this March, the company was taken by surprise by what Tay had become. Within less than 24 hours of conversation with Twitter users Tay had learned to make racist, anti-semitic and misogynistic statements that have raised eyebrows in the Twitter community and beyond. What had happened? While Microsoft certainly tested the chat bot before release, planning for the reactions and the social environment in which it was deployed proved tremendously difficult.

Yet, the Tay Twitter chatbot incident is just one example for the many challenges which arise when embedding algorithms and computing systems into an ever increasing spectrum of social systems. In this viewpoint we argue that, due to the resulting feedback loops by which computing technologies impact social behavior and social behavior feeds back on (learning) computing systems, we face the risk of losing control over the systems that we engineer. The result are unintended consequences that affect both the technical and social dimension of computing systems, and which computer science is currently not well-prepared to address. Highlighting exemplary challenges in core areas like (1) algorithm design, (2) cyber-physical systems, and (3) software engineering, we argue that social aspects must be turned into first-class citizens of our system models. We further highlight that the social sciences, in particular the interdisciplinary field of *Computational Social Science* [1], provide us with means to quantitatively analyze, model and predict human behavior. As such, a closer integration between computer science and social sciences not only provides social scientists with new ways to understand social phenomena. It also helps us to regain control over the systems that we engineer.

Regaining Control about Algorithms The design of algorithms is traditionally seen as a combination of mathematical problem solving, programming techniques, and performance optimization. While it is clear that space and runtime complexity of algorithms are key aspects that must guide their design, we lack proper language and methods to deal with their *social complexity*. This social complexity can refer to, e.g., legal, sociological or ethical implications of algo-

rithms which arise if computing is applied to analyze or – increasingly – influence and steer human behavior. As an example consider recommendation algorithms which recommend, for instance, news items, books, food, or even potential dating partners based on past user behavior. Including personal characteristics like education, income, age, gender, ethnicity or even sexual preferences may improve the predictive quality of such algorithms. At the same time it can reinforce adverse social phenomena like, e.g., socio-economic segregation, discrimination, echo-chambers or polarization, all of which recently triggered a debate about legal regulations of machine learning algorithms and the role of social media in our political landscape. Although issues like the responsible use of computing or values in the design of technical systems [2] have been discussed for decades[1], since algorithms become the "sensory gating" of both individuals and the digitized society as a whole, the problem has reached a new level of complexity and urgency. It opens up new avenues of research that require us to answer questions like: Can we quantitatively assess and predict the impact of recommender algorithms on public opinion formation and polarization? How to incorporate fairness, transparency and accountability aspects into machine learning and information retrieval algorithms? Can new classes of algorithms reconcile the undoubted economic opportunities of big data with the individual right of anonymity, with societal needs as well as ethical and legal constraints?

Both applied and theoretical computer scientists have taken on the challenge of addressing these questions. In the machine learning community this is documented in, e.g., the *Fairness, Accountability, and Transparency in Machine Learning (FAT ML)* workshop series[2], and recent works addressing problems like, e.g., implicit discrimination in data mining [3] or classification algorithms that incorporate fairness constraints [4]. Similarly, the development of privacy-enhancing technologies based on 'zero-knowledge' proofs [5], statistical databases and differential privacy [6] pose interesting challenges for theoretical computer science and algorithm design and offer ways to mitigate some of the imminent challenges. These can be seen as promising examples for a new class of algorithms whose design not only takes into account what is computationally possible, but also by what is socially desirable and ethically and legally justifiable. However, if and how algorithm design can be informed by theories about the mechanisms behind (adversary) collective phenomena like the ones mentioned above is still an open question. Future research is thus likely to benefit from a closer integration of existing (computational) models of human behavior.

Regaining Control about Cyber-Physical Systems How digital technologies affect the life of citizens and the development of whole societies has recently become the focus of a lively public debate. What has been less discussed is that the digitization of society also implies that technical systems are increasingly

[1] cf. the Computer Professionals for Social Responsibility (CPSR) founded in 1983

[2] http://www.fatml.org/

affected by human behavior and societal phenomena which are impossible to predict at design time, and difficult to anticipate at run time. Examples can be found in cyber-physical systems and smart infrastructures like power grids, communication networks, information or mobility systems, which increasingly incorporate autonomous components that monitor and adapt to the behavior of users. This not only leads to a feedback cycle between technical systems and human behavior, but it also threatens models which treat humans as an externality rather than as an integral component of technical systems. Changing this requires us to tackle questions like: How do incentive and pricing mechanisms in smart infrastructures, e.g. for energy, mobility or communication, influence collective user behavior, and how does this behavior feed back on the infrastructure? How will human drivers in traffic infrastructures respond to adaptive traffic control schemes, and how will this very response influence their performance? Will human drivers react differently to (fleets of) self-driving cars, and can we incorporate this human reaction into the design of autonomous vehicles?

Answering these questions requires extensive competencies in the modeling of human behavior and social systems which is far outside the scope of traditional computer science curricula. Nevertheless, the urgency to better integrate technical and social aspects in the modeling of systems is being acknowledged by a growing community of researchers that call for multi-level modeling techniques.Promising examples can be found in works showing that the feedback between dynamic pricing mechanisms and consumer behavior can possibly trigger blackouts in smart grids [7]. To prevent such detrimental phenomena, we will need to incorporate models for human behavior into smart grid technologies [8]. As an example, it has argued how formal calculus allows to operationalize social aspects in socially-intelligent infrastructures [9, 10].

Regaining Control about Software Engineering With more than five decades of experience, software engineering is one of the oldest disciplines in computer science. Pioneers in this field provided us with powerful programming languages, software design principles, methods to test, analyze and validate code, as well as tools that support developers in their daily work. But despite these *technological and procedural advances*, software engineering still poses a substantial challenge: Credible reports indicate that a substantial fraction of software projects either run over time (and budget) or fail altogether [11]. Technical factors aside, the software engineering community has long ago acknowledged that human and social aspects crucially influence the genesis of software systems [12]. Understanding development processes thus requires us to answer questions like: How are coordination and communication structures of software developers related to project success or code quality? How do motivational and psychological factors, such as emotions, influence software development? Does the social status of developers influence how their software artifacts are perceived? And how resilient is a development team against the departure of central team members?

A growing number of quantitative studies at the interface between software engineering and social science has taken on the challenge to answer these questions. Examples can be found in major software engineering venues like *Mining Software Repositories*, *Empirical Software Engineering* or the *International Conference on Software Engineering*. A key aspect of these works is that they apply state-of-the-art data mining and pattern recognition techniques to large-scale data sets that capture rich, multifaceted traces of the collaboration and coordination between developers [13, 14]. This has allowed researchers to show how social factors influence the productivity of software development teams [15, 16] or how developer emotions influence software development processes [17, 18]. We are convinced that a wider adoption of such data-driven techniques, and their integration with social science theories, will generate actionable insights into the social dynamics at work in development teams, helping us to improve both the process and the outcome of collaborative software engineering.

Will Computer Science become a Social Science? The fact that computer scientists are currently not well-trained for the challenges that we have outlined calls for a more systematic integration of computer science with the social sciences. A promising development in this direction is the emergence of *computational social science*, which combines theories and methodologies from the social sciences, including social psychology and behavioral economics, with computer science approaches like large-scale simulations, data mining, and machine learning. The growing availability of large data sets of humans and their interactions, both from offline and online domains, is a major driving force of this interdisciplinary research community [19]. Pioneering works have shown how large-scale analyses can be used to detect subtle patterns in social behavior which can either corroborate or invalidate theories from the social sciences.

As such, the ubiquitous adoption of computing systems not only complicates the design of technical systems. Making human and social aspects accessible to measurement, modeling and thus specification can also help us to better design and control socio-technical systems whose study has so far mostly been based on anecdotal evidence or small-scale case studies [20]. However, for this we must go beyond a mere phenomenological perspective, focusing instead on the *mechanisms* underlying these phenomena. Data-driven modeling approaches which, e.g., model dynamical processes in social networks or utilize agent-based models to study collective phenomena, are needed to reproduce social behavior and thus open up new ways to forecast and control systems dynamics. To address these challenges, computer science must engage in a cross-fertilization with sociologists, political scientists and psychologists. Failing to reach out to these communities would have consequences not only for computer science, but for society as a whole. As such, the question whether computer scientists will be seen as part of the problem or as part of the solution is still open. The choice is ours.

References

[1] David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009.

[2] Helen Nissenbaum. Values in the design of computer systems. *Computers and Society*, 28(1):38–39, 1998.

[3] Sara Hajian and Josep Domingo-Ferrer. A methodology for direct and indirect discrimination prevention in data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 25(7):1445–1459, 2013.

[4] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: A mechanism for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

[5] Bernard Chazelle. Computing: The security of knowing nothing. *Nature*, 446(7139):992–993, 2007.

[6] Cynthia Dwork. *Differential Privacy*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[7] Sebastian M. Krause, Stefan Börries, and Stefan Bornholdt. Econophysics of adaptive power markets: When a market does not dampen fluctuations but amplifies them. *Phys. Rev. E*, 92:012815, Jul 2015.

[8] Peter Palensky and Dietmar Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE transactions on industrial informatics*, 7(3):381–388, 2011.

[9] Andrew J. I. Jones, Alexander Artikis, and Jeremy Pitt. The design of intelligent socio-technical systems. *Artif. Intell. Rev.*, 39(1):5–20, 2013.

[10] Jeremy V Pitt, Dídac Busquets, Ada Diaconescu, Andrzej Nowak, Agnieszka Rychwalska, and Magdalena Roszczynska-Kurasinska. Algorithmic self-governance and the design of socio-technical systems. In *ECSI*, pages 262–273. Citeseer, 2014.

[11] Narciso Cerpa and June M. Verner. Why did your project fail? *Commun. ACM*, 52(12):130–134, December 2009.

[12] Melvin E Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[13] Georgios Gousios and Diomidis Spinellis. GHTorrent: GitHub's data from a firehose. In Michael W. Godfrey and Jim Whitehead, editors, *MSR '12: Proceedings of the 9th Working Conference on Mining Software Repositories*, pages 12–21. IEEE, June 2012.

[14] Marcelo Cataldo, Ingo Scholtes, and Giuseppe Valetto. A complex networks perspective on collaborative software engineering. *Advances in Complex Systems*, 17(7-8), 2014.

[15] Ingo Scholtes, Pavlin Mavrodiev, and Frank Schweitzer. From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in open source software projects. *Empirical Software Engineering*, 21(2):642–683, 2015.

[16] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. Developer onboarding in github: The role of prior social links and language experience. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 817–828, New York, NY, USA, 2015. ACM.

[17] Emitza Guzman, David Azócar, and Yang Li. Sentiment analysis of commit comments in github: An empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 352–355, New York, NY, USA, 2014. ACM.

[18] David Garcia, Marcelo Serrano Zanetti, and Frank Schweitzer. The role of emotions in contributors activity: A case study of the gentoo community. In *Proceedings of the International Conference on Social Computing and Its Applications*, pages 410–417, 2013.

[19] Markus Strohmaier and Claudia Wagner. Computational social science for the world wide web. *IEEE Intelligent Systems*, 29(5):84–88, 2014.

[20] Alessandro Vespignani. Predicting the behavior of techno-social systems. *Science*, 325(5939):425–428, 2009.