

## 1 Research Summary

I named the area of my research *Digital Archeology of Software* [1]: a study of software developers' culture and behavior through the recovery, documentation, and analysis of digital remains. These digital traces reflect various projections of collective and individual activity. The fundamental method of *digital archeology* is, therefore, the reconstruction and quantification of the behavior of an individual, a team, or an organization from these projections. I refer to it as *organizational tomography*. It allows a live, non-intrusive, fine-grained, and practical way to observe, analyze, and support large groups such as software development teams [2], entire companies [3, 4], or an entire population of open source projects [5, 6]. Two decades ago, software development was the only domain with accessible detailed records of people's activities retrievable from version control and problem tracking systems. The research in the area has experienced a dramatic growth as, for example, indicated by the recognition of my early work [2], conferences such as Mining Software Repositories and Mining Software Archives, and adoption of these novel measurement and decision support methods by industry (see "Impact" Section for more details). Presently, many other forms of human endeavor are recorded in unprecedented detail leading to the explosion of interest in ways to analyze these digital projections [7]. This new approach to understanding human activity can be both enhanced by and also advance the *organizational tomography* and open new opportunities to benefit mankind (see "Vision" Section for more details).

## 2 Research Topics

### Transfer of work

Software development practice is experiencing a radical change driven by the open source movement, the business needs to move development to low-cost locations, the aging and renewal of core developers in legacy products, recruiting in fast growing Internet companies, and the turmoil of the economy causing unprecedented turnover in software projects.

To address these challenges, my latest research investigates the transfer of work [1] and the associated phenomena related to organizational change [3] and the growth of software project competencies [8]. More specifically, I discovered that the relationship between the social and technical competencies was associated with the fraction of new participants who become long term contributors of a software project [9]. It provides a tantalizing opportunity to study how the initial environment a person encounters when joining a

team or an organization affects motivation and long-term behavior.

Given the diversity of software projects and the creative nature of software development, I sought to find out how much the context of a project may affect the outcomes by conducting multi-company studies [10, 11, 12] which took into account not only technical, but also social [13, 12] aspects of software development. Surprisingly, a software project with identical requirements may cost an order of magnitude more and require correspondingly more effort simply because of the differences in the nature of company's customer base [10]. At the same time, many phenomena related to how developers make mistakes leading to software defects are similar in diverse projects and companies [3, 12, 11].

### Distributed development

Combining insights from software development models and software change data I constructed tools optimally to distribute work [14], estimate and present developer expertise [15], and improve the quality of software [16, 17]. I have packaged these tools in the *SoftChange* system and deployed it in Lucent's and Avaya's projects. The measurement and analyses from these tools are used to determine if projects meet their quality targets [17], to guide strategic company decisions [4], and to demonstrate that the reliability of products meets the requirements of existing and prospective customers [17, 18].

### Large-scale phenomena

Assembling large, interconnected data sets from multiple projects and data sources allows to answer entire classes of questions about large-scale behavior that could not be addressed in other ways. I created methods for discovering, acquiring, integrating, and analyzing these heterogeneous types of data [19, 20]. A complete collection of data within a company [4] or an entire collection of open source code [5], gives ability to determine the source code origins and authorship via Universal Version History [21, 22] or the spread of innovation via code reuse [6]. The authorship and code origin analyses are used in Avaya to identify open source code, thus addressing key source code licensing issues. More importantly, such analysis opens the possibility to investigate creativity and innovation in software development: an intangible asset that is both crucial to business success and, at the same time, tends to be the first casualty in any cost-cutting, offshoring, or outsourcing scenario.

## Cost and quality

A decade ago, I introduced a methodology that uses widely available repositories of automatically recorded project data in change management and problem tracking systems to model and analyze software development. By performing statistical analysis of software changes I quantified the most influential drivers of cost [23, 24], interval [25], quality [16] and differences between development patterns in commercial and open source software development [26]. These methods are now widely used throughout the software engineering community to predict quality, cost, and schedule.

## User interfaces, visualization, optimization

I began developing the methods that became digital archeology in my earlier work in which I investigated user interfaces [27, 28] and the visualization and modeling of complex phenomena such as the spread of diseases [29, 30], displays of a large number of images [31], and the response of the brain to visual or cognitive stimuli [32, 33]. My work on algorithms involves Bayesian methods of optimization for non-convex functions [34], boosting methods to optimize parameters of discrete optimization heuristics [35], clustering of high-dimensional datasets [36], isotonic regression [37], and the estimation of a covariance function from irregularly-shaped spatial aggregates [30].

## 3 Impact

My work affected research and practice. Many domains, especially in industry, where measurement was impractical before, benefited the most. Some of my early work [2] inspired research in domains not related to software engineering, for example, innovation [38], computer-supported collaborative work [39], management science [40], and others.

It also led to the creation of several sub-fields within software engineering. The Conference on Mining Software Repositories, Mining Software Archives, and Workshop on Recommendation Systems for Software Engineering are venues for several of these sub-fields, where research building on techniques I developed is regularly presented.

Furthermore, the techniques I pioneered have reshaped empirical software engineering and are used in a large fraction of publications in the main software engineering conferences, such as International Conference on Software Engineering and Foundation of Software Engineering.

The impact of my work on practice was even more profound, because the information crucial for decision making could finally be obtained without incurring prohibitive costs and delay of manual collection practices [41], thus leading to

better and radically different decisions and practices as outlined below. The impact was not contained within a single company, but affected the entire industry, including companies such as Microsoft, IBM, AT&T, and Alcatel/Lucent.

I developed techniques for precise estimation of productivity gains for a variety of tools and software development methods. This led to their adoption and to changes in development practices [24, 23, 42, 43, 11].

The investigation of quality issues in software patches resulted in the discovery that the desire to increase the revenue by including new features in patches made it inevitable that such patches would fail. This led, among other things, to the abandonment of this apparently lucrative, but in reality harmful practice [44, 16].

The discovery that the defect density (the development view of quality) is anti-correlated with the fraction of customers reporting software issues (the customer view of quality) resulted in more effective quality improvement methods that take the customer perspective into account. These measures and approaches are used within Avaya and shared with key Avaya customers [17].

Many other findings have dramatically changed the practice, for example, I developed an *in-vivo* system availability measurement [18] based on the alarms and problem reports from installed systems quantified the actual availability for different types of systems and customers. Such estimates were not available before and they demonstrated that only a handful of the most complex and largest installations reach the “five-nines” availability boundary, while the remaining installations operate far above these levels. The findings of performance issues related to transfer of code [1] and the project expertise [8] led to radical changes in the way work transfers were handled, in improvement of the hiring practices at offshore locations, and other substantial changes. The quantification of software activities and quality constitute the basis for the annual company-wide State of Software Report [4] that drives company decisions on how to improve software development in Avaya.

## 4 Vision

Software development is not an isolated activity: apart from dealing with the technology, individuals, and teams, it also needs to satisfy existing needs or to generate new needs and it is operating in an environment affected by the global economic and political dynamics. These larger forces shape and direct software development in many ways, e.g., outsourcing and offshoring. The issues facing software development, therefore, have to be resolved within that broader context. At the opposite extreme are aspects of software development related to individuals, their creative activities, and their learn-

ing process. As the economic and political forces change the environment in ways that stress the limits of human cognitive abilities, for example, the ability of a newcomer to become competent in a large software project [8], the outcomes critically depend on the ingenuity of software creators.

My research, therefore, will continue to select and solve critical issues at different scales and in various contexts, painting a unified picture of software development and enabling progress at this critical junction of society and technology. To achieve that, I will continue existing collaborations and will forge new ones in the areas where less progress has been made so far.

**At the largest scale** I will gain insights into the behavior of the entire open source and corporate software development. I will create a repository of most open source, corporate, and government software project data, create valid measures that describe individual and organizational behavior (such as the spread of innovation through reuse [6]), and create tools to address critical issues faced at this level. I have already obtained the collection of most publicly available source code version history data [5], and I have started a collaboration with Peking University in China and Queens University in Canada to collect other sources of data such as problem tracking systems and mailing lists. From the corporate perspective I will create a multi-company repository similar to the one I created for Avaya [4] to study software and organizational phenomena in a global setting [12, 11].

**At the intermediate scale** I will continue to study related groups of software projects (ecosystems) in the open source and commercial environments. A key open question is how do ecosystems come into being? One hypothesis is that a successful project will grow until the core team could no longer cope with the increased amount of work. This would cause a split into sub-teams and sub-projects which would ultimately populate the ecosystem. Another critical question is to what extent the context determines the best practices of software development [10, 45]. For example, as a result of acquisitions, Avaya has multiple products that serve identical business purposes and each uses its own development and support practices: to what extent these practices are determined by the company, the technical structure of the product, or the customer requirements?

**At the smallest scale** I will continue the study of individual developers, why they are motivated, and how they learn. In particular, I will quantify the motivation of developers. One aspect of motivation may be reflected by the time a developer stays in the project [9]. However, the measurement based on observed activities and resulting artifacts may not

suffice to understand some dimensions of developers' motivation. I will fill these gaps by direct measurements of their cognitive responses to software development scenarios using, for example, Functional Magnetic Resonance Imaging (fMRI).

As noted above, software engineering involves people, teams, organizations, culture, and society. My research, therefore, will both learn and contribute to these domains because most of the findings are likely to be universal: answer the fundamental questions of who we are, where we come from, and how should we live.

## 5 Teaching Summary

Research can not exist without teaching. Results, no matter how detailed the description may be, would mean nothing without somebody skilled in the area to interpret what they mean, how they can be enhanced, and how to apply them to benefit the society. Also, the pleasure of a significant discovery can only be eclipsed by the pleasure of learning about a student's new and significant achievement. My current work of estimating mentor-follower relationships [1] and the dimensions and growth of developer competencies [8] provide insights and lessons on how the teaching and mentorship could be improved in a research environment as well. The practice and motivation are the two inseparable cornerstones for gaining competence: the motivation ensures more practice and the practice leads to expertise gain and motivates further practice at the next competence area or level. My approach is to allow students to spend sufficient time on practice and to select tasks appropriate for their skill to ensure strong motivational feedback once they succeed.

### Mentoring

I have worked with a number of students, postdoctoral fellows, or colleagues whom I attracted to *digital archeology*. In particular, Prof. Minghui Zhou's six-month visit at Avaya Labs Research led to the discovery of Inverse Conway Law [45]: the teams taking over a legacy software product spontaneously organize their work according to the structure of the work used by the original product creation team. Patrick Wagstrom (for whom I was a PhD thesis committee member) investigated associations between the commercial and volunteer developer participation in open source projects [46]. Paul Li [47] and Hung-Fu Chang [22, 21] were PhD students and summer interns at Avaya Labs Research and Prof. Alberto Espinosa [48] was finishing his PhD thesis at Bell Labs. I also supervised summer interns Christopher E. Weaver from University of Wisconsin (now Assistant Professor at the University of Oklahoma) and Jian Shen from

Stanford University. Long-term collaborations ensued with Todd Graves [49, 50, 51, 52] (now at Los Alamos National Laboratory) who started as a National Institute for Statistical Sciences postdoctoral fellow at Bell Labs, and with Prof. Harvey Siy [53, 23, 42, 54, 55] who started as a PhD. student at Bell Labs.

### Formal courses

I do not see many differences among disciplines in terms of basic principles that students need to be taught. Each discipline uses different terminology and within each discipline a variety of fashionable practices with their own buzzword-laden vocabularies are appearing and disappearing over time. Upon closer look, the same principles of rational inquiry, of properly collecting data and inferring conclusions, of creating models of reality capturing aspects salient to the issue at hand are hiding behind the menagerie of terms. Each discipline simply adapts these basic principles to its premises, to the types of problems it addresses, and, in the process, often invents a new vocabulary. In teaching I focus on conveying not just the terminology of a discipline or a practice, but also the endurance and the discipline-invariance of the fundamental principles.

I enjoyed teaching a number of graduate- and undergraduate-level courses listed below. In addition to the core software engineering, empirical, and practice-focused courses I would consider teaching areas of software engineering and computer science that currently lack a firm empirical basis. Teaching such a course would clarify how to strengthen the role of the empirical approach in these domains.

Measuring Globally Distributed Software Development, Pacific Rim Summer School, Beijing, China, 2010, <http://www.cpathi18n.org>

Digital Archeology, Mining Software Repositories Summer School, Kingston, ON, 2010. <http://msrcanada.org/school/>

Predicting Risk of Software Changes, Mining Software Repositories Summer School, Kingston, ON, 2010.

Empirical Estimates of Software Availability, Mining Software Repositories Summer School, Kingston, ON, 2010.

Chunking Code, Mining Software Repositories Summer School, Kingston, ON, 2010.

Domain-specific defect models, Tsinghua University, Beijing, 2009.

How to run empirical studies using project repositories?, 4th International Advanced School of Empirical Software Engineering, Rio de Janeiro, Brazil 2006.

“Statistical Graphics,” 1994, CMU.

“Spatial Statistics,” 1993, CMU.

“Probability Theory and Random Processes,” 1993 and 1994, CMU.

“Probability and Applied Statistics for Management and Social Sciences,” 1993 and 1994, CMU.

“Probability and Applied Statistics for Physical Sciences,” 1991, CMU.

“Queueing Theory,” 1989 and 1990, CMU.

### References

- [1] Audris Mockus. Succession: Measuring transfer of code and developer productivity. In *2009 International Conference on Software Engineering*, Vancouver, CA, May 12–22 2009. ACM Press.
- [2] A. Mockus, R. F. Fielding, and J. Herbsleb. A case study of open source development: The apache server. In *22nd International Conference on Software Engineering*, pages 263–272, Limerick, Ireland, June 4–11 2000.
- [3] Audris Mockus. Organizational volatility and its effects on software defects. In *ACM SIGSOFT / FSE*, pages 117–126, Santa Fe, New Mexico, November 7–11 2010.
- [4] Randy Hackbarth, Audris Mockus, John Palframan, and David Weiss. Assessing the state of software in a large enterprise. *Journal of Empirical Software Engineering*, 10(3):219–249, 2010.
- [5] Audris Mockus. Amassing and indexing a large sample of version control systems: towards the census of public source code history. In *6th IEEE Working Conference on Mining Software Repositories*, May 16–17 2009.
- [6] Audris Mockus. Large-scale code reuse in open source software. In *ICSE’07 Intl. Workshop on Emerging Trends in FLOSS Research and Development*, Minneapolis, Minnesota, May 21 2007.
- [7] Nathan Eagle, Michael Macy, and Rob Claxton. Network diversity and economic development. *Science*, 328(5981):1029–1031, May 2010.
- [8] Minghui Zhou and Audris Mockus. Developer fluency: Achieving true mastery in software projects. In *ACM SIGSOFT / FSE*, pages 137–146, Santa Fe, New Mexico, November 7–11 2010.

- [9] Minghui Zhou and Audris Mockus. Does the initial environment impact the future of developers? In *ICSE 2011*, Honolulu, Hawaii, May 21–28 2011. accepted.
- [10] Bente C.D. Anda, Dag I.K. Sjøberg, and Audris Mockus. Variability and reproducibility in software engineering: A study of four companies that developed the same system. *IEEE Transactions on Software Engineering*, 35(3), May/June 2009.
- [11] Audris Mockus, Nachiappan Nagappan, and T Dinh-Trong, Trung. Test coverage and post-verification defects: A multiple case study. In *International Conference on Empirical Software Engineering and Measurement*, Lake Buena Vista, Florida USA, October 2009. ACM.
- [12] Marcelo Cataldo, Audris Mockus, Jeffrey A. Roberts, and James D. Herbsleb. Software dependencies, the structure of work dependencies and their impact on failures. *IEEE Transactions on Software Engineering*, 2009.
- [13] James Herbsleb and Audris Mockus. Formulation and preliminary test of an empirical theory of coordination in software engineering. In *2003 International Conference on Foundations of Software Engineering*, Helsinki, Finland, October 2003. ACM Press.
- [14] Audris Mockus and David M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.
- [15] Audris Mockus and James Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *2002 International Conference on Software Engineering*, pages 503–512, Orlando, Florida, May 19–25 2002. ACM Press.
- [16] Audris Mockus and David M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2):169–180, April–June 2000.
- [17] Audris Mockus and David Weiss. Interval quality: Relating customer-perceived quality to process quality. In *2008 International Conference on Software Engineering*, pages 733–740, Leipzig, Germany, May 10–18 2008. ACM Press.
- [18] Audris Mockus. Empirical estimates of software availability of deployed systems. In *2006 International Symposium on Empirical Software Engineering*, pages 222–231, Rio de Janeiro, Brazil, September 21–22 2006. ACM Press.
- [19] Audris Mockus. Software support tools and experimental work. In V Basili and et al, editors, *Empirical Software Engineering Issues: Critical Assessments and Future Directions*, volume LNCS 4336, pages 91–99. Springer, 2007.
- [20] Audris Mockus. Missing data in software engineering. In J. Singer et al., editor, *Guide to Advanced Empirical Software Engineering*, pages 185–200. Springer-Verlag, 2008.
- [21] Hung-Fu Chang and Audris Mockus. Evaluation of source code copy detection methods on FreeBSD. In *5th Working Conference on Mining Software Repositories*. ACM Press, May 10–11 2008.
- [22] Hung-Fu Chang and Audris Mockus. Constructing universal version history. In *ICSE'06 Workshop on Mining Software Repositories*, pages 76–79, Shanghai, China, May 22–23 2006.
- [23] D. Atkins, A. Mockus, and H. Siy. Measuring technology effects on software change cost. *Bell Labs Technical Journal*, 5(2):7–18, April–June 2000.
- [24] D. Atkins, T. Ball, T. Graves, and A. Mockus. Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Transactions on Software Engineering*, 28(7):625–637, July 2002.
- [25] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: Distance and speed. In *23rd International Conference on Software Engineering*, pages 81–90, Toronto, Canada, May 12–19 2001.
- [26] Audris Mockus, Roy T. Fielding, and James Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):1–38, July 2002.
- [27] Stacie Hibino and Audris Mockus. handiMessenger: Awareness-enhanced universal communication for mobile users. In Fabio Paternò, editor, *Mobile Human-Computer Interaction, 4th International Symposium, Mobile HCI 2002, Pisa, Italy, September 18–20, 2002, Proceedings*, volume 2411 of *Lecture Notes in Computer Science*, pages 170–183, Pisa, Italy, September, 18–20 2002. Springer.
- [28] S.L. Hibino, T. Graves, and A. Mockus. A web based approach to interactive visualization in context. In *Advanced Visual Interfaces*, pages 181–188, Palermo, Italy, May 23–26 2000.

- [29] W.F. Eddy and A. Mockus. An example of the estimation and display of a smoothly varying function of time and space - the incidence of mumps disease. *Journal of the American Society for Information Science*, 45(9):686–693, 1994.
- [30] Audris Mockus. Estimating dependencies from spatial averages. *Journal of Computational and Graphical Statistics*, 7(4):501–513, 12 1998.
- [31] W.F. Eddy and A. Mockus. An interactive icon index: Images of the outer planets. *Journal of Computational and Graphical Statistics*, 5(1):100–111, 1996.
- [32] A. Mockus, W.F. Eddy, S.Y. Chang, and K.R. Thulborn. Software for the visualization of fMRI data. In *Proceedings of the International Society for Magnetic Resonance in Medicine Fourth Scientific Meeting and Exhibition*, page 1774, 1996.
- [33] W.F. Eddy, M. Fitzgerald, C. Genovese, N. Lazar, A. Mockus, and Welling J. The challenge of functional magnetic resonance imaging. *Journal of Computational and Graphical Statistics*, 8(3):545–558, September 1999.
- [34] A. Mockus and L. Mockus. Designing software for global optimization. *Informatica*, 1(1):71–88, 1990.
- [35] A. Mockus, J. Mockus, and L. Mockus. Bayesian heuristic approach (BHA) and applications to discrete optimization. *Fields Institute Communications*, 18:153–165, 1998.
- [36] W.F. Eddy, A. Mockus, and S. Oue. Approximate single linkage cluster analysis of large data sets in spaces of high dimension. *Computational Statistics and Data Analysis*, 23:29–43, 1996.
- [37] M. Lavine and A. Mockus. A nonparametric Bayes method for isotonic regression. *Journal of Statistical Planning and Inference*, 46:235–248, 1995.
- [38] Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, July 2003.
- [39] Gerard Beenen, Kimberly Ling, Xiaoqing Wang, Klarissa Chang, Dan Frankowski, Paul Resnick, and Robert E. Kraut. Using social psychology to motivate contributions to online communities. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004.
- [40] Alan MacCormack, John Rusnak, and Carliss Baldwin. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Journal Management Science*, 52(7), 2006.
- [41] R. Grady and E. Caswell. *Software metrics*. Prentice-Hall, Englewood Cliff, 1987.
- [42] D. L. Atkins, A. Mockus, and H. P. Siy. *Value Based Software Engineering*, chapter Quantifying the Value of New Technologies for Software Development, pages 327–344. Springer Verlag Berlin Heidelberg, 2006.
- [43] Birgit Geppert, Audris Mockus, and Frank Rößler. Refactoring for changeability: A way to go? In *Metrics 2005: 11th International Symposium on Software Metrics*, Como, September 2005. IEEE CS Press.
- [44] A. Mockus, H. Siy, T. Sundresh, J. Chen, and TL Graves. Role of change size, complexity, and developer expertise in predicting the quality of a software update. Technical Report 10009677-000324-01TM, Lucent Technologies, 2000.
- [45] Minghui Zhou, Audris Mockus, and David Weiss. Learning in offshored and legacy software projects: How product structure shapes organization. In *ICSE Workshop on Socio-Technical Congruence*, Vancouver, Canada, May 19 2009.
- [46] Patric Wagstrom, James Herbsleb, Robert Kraut, and Audris Mockus. The impact of commercial organizations on volunteer participation in an online community. In *Academy of Management Annual Meeting*, Montreal, CA, August 6-10 2010.
- [47] Audris Mockus, Ping Zhang, and Paul Li. Drivers for customer perceived software quality. In *ICSE 2005*, pages 225–233, St Louis, Missouri, May 2005. ACM Press.
- [48] J. A. Espinosa, R.E. Kraut, S. A. Slaughter, J. F. Lerch, J. D. Herbsleb, and A. Mockus. Shared mental models and coordination in large-scale, distributed software. In *Proc. International Conference in Information Systems*, New Orleans, LA, December 16–19 2001.
- [49] Todd L. Graves and Audris Mockus. Inferring change effort from configuration management data. In *Metrics 98: Fifth International Symposium on Software Metrics*, pages 267–273, Bethesda, Maryland, November 1998.
- [50] Stephen G. Eick, Todd L. Graves, Alan F. Karr, J. S. Marron, and Audris Mockus. Does code decay? assessing the evidence from change management data.

*IEEE Transactions on Software Engineering*, 27(1):1–12, 2001.

- [51] Nancy Staudenmayer, Todd Graves, and Audris Mockus. Adapting to a new environment: How a legacy software organization copes with volatility and change. In *Academy of Management Chicago 1999 Conference*, Chicago, Illinois, August 1999.
- [52] T. Graves and A. Mockus. Identifying productivity drivers by modeling work units using partial data. *Technometrics*, 43(2):168–179, May 2001.
- [53] Audris Mockus, Adam Porter, Harvey Siy, and Lawrence G. Votta. Understanding the sources of variation in software inspections. *ACM Transactions on Software Engineering and Methodology*, 7(1), January 1998.
- [54] Harvey Siy and Audris Mockus. Measuring domain engineering effects on software coding cost. In *Metrics 99: Sixth International Symposium on Software Metrics*, pages 304–311, Boca Raton, Florida, November 4–6 1999.
- [55] HP Siy, A Mockus, JD Herbsleb, M Krishnan, and GT Tucker. Making the software factory work: Lessons from a decade of experience. In *Metrics 2001: Seventh International Symposium on Software Metrics*, pages 317–327, London, England, April 4-6 2001.