1 Research Summary

The area of my research is *Digital Archeology* [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 the reconstruction and quantification of the behavior of an individual, a team, or an organization from these projections. *Digital Archeology* 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 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. Some of my early work [2] inspired research in domains not related to software engineering, for example, innovation [7], computer-supported collaborative work [8], management science [9], and others.

Combining insights from software development models and software change data I constructed tools optimally to distribute work [10], estimate and present developer expertise [11], and improve the quality of software [12, 13]. 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 [13], to guide strategic company decisions [4], and to demonstrate that the reliability of products meets the requirements of existing and prospective customers [13, 14].

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 [15]. This approach to understanding human activity can be both enhanced by and also advance *Digital Archeology* and open new opportunities to benefit mankind (see "Vision" Section for more details).

2 Research Topics

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 [16, 17], interval [18], quality [12] and differences between development patterns in commercial and open source software development [19]. These methods are now widely used throughout the software engineering community to predict quality, cost, and schedule.

The impact of my work dramatically affected practice because the information crucial for decision making could finally be obtained without incurring prohibitive costs and delay of manual collection practices [20], 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.

In particular, 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 [17, 16, 21, 22, 23].

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 [24, 12].

The discovery that 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 [13].

I introduced Customer Quality Metric [13] (the fraction of early customers affected by product issues) which is now used to evaluate and improve all major software products in Avaya.

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 [25, 26]. 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 [27, 28] 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 im-

portantly, 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.

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 research investigates the transfer of work [1] and the associated phenomena related to organizational change [3] and the growth of software project competencies [29]. 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 [30]. 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.

The findings of performance issues related to transfer of code [1] and the project expertise [29] led to radical changes in the way work transfers were handled, in improvement of the hiring practices at offshore locations, and other substantial changes.

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 [31, 23, 32] which took into account not only technical, but also social [33, 32] 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 [31]. At the same time, many phenomena related to how developers make mistakes leading to software defects are similar in diverse projects and companies [3, 32, 23].

User interfaces, visualization, optimization

I began developing the methods that became *Digital Archeology* in my earlier work in which I investigated user interfaces [34, 35] and the visualization and modeling of complex phenomena such as the spread of diseases [36, 37], displays of a large number of images [38], and the response of the brain to visual or cognitive stimuli [39, 40]. My work on algorithms involves Bayesian methods of optimization for non-convex functions [41], boosting methods to optimize parameters of discrete optimization heuristics [42], clustering of high-dimensional datasets [43], isotonic regression [44], and the estimation of a covariance function from irregularly-shaped spatial aggregates [37].

3 Vision

Software development is not an isolated activity: apart from dealing with the technology, individuals, and teams, it also has to satisfy existing needs or to generate new needs and to react to technical, economic, and political changes. These larger forces shape and direct software development in new ways as in, for example, outsourcing/offshoring, cloud computing, and mobile applications. The issues facing software development have to be resolved within this broader context.

In particular, the cloud computing opens new possibilities for Digital Archeology by concentrating immense collections of software development traces (GoogleCode has more than 300 thousand software projects and GitHub has more than three million repositories). Mobile computing and social networks enrich the traces with the explicit geographic and social components. These technologies also pose urgent research challenges of understanding and improving software development for cloud-based, mobile, and social applications.

Furthermore, the detailed records of individual and collective activity are now available for domains that are far removed from software development, such as IT services and games. Each domain reflects common aspects of human behavior related to individuals, their creative activities, and their learning process. As the economic, political, and cultural 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 [29], the outcomes critically depend on the ingenuity and motivation [45] of software creators.

My research will continue to select and solve critical issues at different scales and in various contexts, painting a unified picture of individual and collective behavior and enabling progress at the critical junction of society and technology. To achieve that, I will continue existing collaborations and will forge new ones.

At the largest scale I will continue to derive insights into the behavior of the entire open source and corporate software development and of related socio-technical systems. I will create a repository of open source, corporate, and gov-

ernment 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 [32, 23]. Such global questions start from basic census: what is the distribution of code, people, and their activities and how does it depend on context? Higher level questions would include comparison of cultures (for example by comparing the creation of Wikipedias in different languages) and the spread of innovative practices and artifacts. The answers to such questions would provide a basis for engineering socio-technical systems with desired features, for example, longevity.

In particular, a regional "census" of technology companies in the New York City region would allow quantification and understanding of factors that affect innovation and growth in such regional ecosystems. Such understanding would be essential to quickly learn what is working, and what is needed to accelerate growth.

At the intermediate scale I will continue to study related groups of 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 can no longer cope with the increased amount of work. Another critical question is to what extent the context determines the best practices of software development [31, 46]. 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? While the project context may be a crucial factor that determines the most suitable tools and practices, experiments that could control for the context in real software projects are prohibitively expensive [31, 47]. When conducted, however, they bring surprises, e.g., only the overall systems size affects maintenance effort [48]. Natural experiments where different companies or open-source projects produce software with similar functionality would be possible using multi-company data described above.

At the smallest scale I will study individuals' micro-activity while navigating or editing artifacts and interacting with others. Such data can be obtained via plug-ins of the authoring environments, e.g., Eclipse, and from frameworks that support analysis of web activities, e.g., Google Analytics. Such micro-level data could dramatically increase the power of the higher level models to predict individual's actions and motivation. However, the measurement based on observed activities and resulting artifacts may not suffice to understand some dimensions of individual's behavior. I hope to fill these gaps by direct measurements of cognitive responses using, for example, Functional Magnetic Resonance Imaging (fMRI).

To solve the outlined challenges, it will be essential to refine and enhance *Digital Archeology* in several ways. Technical challenges include scaling to larger and more diverse data. Analytic challenges include piecing together these unstructured, dirty, and difficult-to-combine data. Novel analysis and visualization techniques will be needed to address these demanding tasks.

In summary, software development 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 for any human endeavor: the answers to the fundamental questions of who we are, where we come from, and how should we live.

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] 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.

[8] 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.

[9] 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.

[10] Audris Mockus and David M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.

[11] 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.

[12] Audris Mockus and David M. Weiss. Predicting risk of software changes. *Bell Labs Technical Journal*, 5(2):169–180, April–June 2000.

[13] 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.

[14] 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.

[15] Nathan Eagle, Michael Macy, and Rob Claxton. Network diversity and economic development. *Science*, 328(5981):1029–1031, May 2010.

[16] 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.

[17] 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.

[18] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: Distance and speed. In *23nd International Conference on Software Engineering*, pages 81–90, Toronto, Canada, May 12-19 2001.

[19] 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.

[20] R Grady and E Caswell. *Software metrics*. Prentice-Hall, Englewood Cliff, 1987.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Minghui Zhou and Audris Mockus. Does the initial environment impact the future of developers? In *ICSE 2011*, pages 271–280, Honolulu, Havaii, May 21–28 2011.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Audris Mockus. Estimating dependencies from spatial averages. *Journal of Computational and Graphical Statistics*, 7(4):501–513, 12 1998.

[38] 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.

[39] 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 Exhibitionn*, page 1774, 1996.

[40] 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.

[41] A. Mockus and L. Mockus. Designing software for global optimization. *Informatica*, 1(1):71–88, 1990.

[42] A. Mockus, J. Mockus, and L.Mockus. Bayesian heuristic approach (BHA) and applications to discrete optimization. *Fields Institute Communications*, 18:153–165, 1998.

[43] 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.

[44] M. Lavine and A. Mockus. A nonparametric Bayes method for isotonic regression. *Journal of Statistical Planning and Inference*, 46:235–248, 1995.

[45] Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in oss community. In *ICSE 2012*, pages 518–528, Zürich, Switzerland, 2012.

[46] 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.

[47] 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.

[48] Dag I.K. Sjøberg, Bente Anda, and Audris Mockus. Questioning software maintenance metrics: a comparative case study. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '12, pages 107–110, New York, NY, USA, 2012. ACM.