Software Production Data

Understanding software creation and evolution is a big data problem

"Software is a critical enabler of America's security, economy, and quality of life and, therefore, is a national priority."[1] Currently we have an inadequate, mostly anecdotal understanding of software production. We do not well-understand the mechanisms required to create, maintain, sustain, and evolve our software infrastructure, especially as it must respond to changing needs and technology over years of development and decades of operation. Developing the necessary evidence-based understanding to manage and improve software production will require us to meet the challenge of obtaining, sharing, and studying the complex, heterogeneous, richly interconnected data that is software production data. Such understanding will facilitate better, faster, more responsive software production decision-making based on evolving project data and a better, faster, cheaper, more responsive software production process. It will yield benefits throughout out the economy to organizations dependent on computer software for incorporating into their products or for their manufacture or as a critical enabler of scientific discovery and innovation.

Software production information process. Software production (both creation and evolution) can usefully be thought of as an information process, defined by Denning as a sequence of representations.[2] At any point in time, the union of the existing artifacts[3] of a software production project constitutes one representation in this information process. The creation of these artifacts, and changes to them over time, demarcate the sequence of representations that forms the software production information process. We can think of this as the state space of software production.

The open source Mozilla Core, an example of a relatively large software production project, illustrates the amount of data that one software production information process can generate. By 20 May 2011, Mozilla Core involved 1379 developers over 13 years creating and evolving 8.2M lines of uncommented code and 2.6M lines of comments.[4] Over the 13 years, developers made 205,779 commits to the source code repository. Considering just source code and comments up to that day, the software production data (that is, the information process) of Mozilla Core is a sequence of 205,779 representations. Each of these representations may encompass as many as 8.2M lines of uncommented code and 2.6M lines of comments.

[1] *Leadership Under Challenge*, p. 33, President's Council of Advisors on Science and Technology, 2007.

[2] P. Denning, *The Great Principles of Computing*, **American Scientist**, Sept.-Oct. 2010.

[3] Example artifacts are requirements, specifications, models, designs, source and executable code, source code comments, test plans and results, bug reports, commit messages, emails and other communications among developers.

[4] http://www.ohloh.net/p/mozilla.

“Developing an understanding of software production entails examining the decisions and assumptions that are reflected or recorded in artifacts.”[5] Artifacts other than source code are important to understanding the creation and evolution of decisions and the assumptions and rationale that support them. The source code is the result of decision making, but rarely contains an explicit description of decisions and assumptions, or a rationale for them. Artifacts constituting the representations of the software production information process may be individually complex and richly, often implicitly, interconnected. They may be formal (e.g., source and executable code, specifications, machine-readable models); informal, but structured (e.g., pseudo-code, requirements, graphical models, test plans); or unstructured natural language text (e.g., email, notes).

An example of a government-sponsored software production project is a "very large, multi-billion dollar software intensive Program of Record (POR) which developed a fully integrated System-of-Systems (SoS) over nearly a decade of activity." There were "multiple engineering teams which were comprised of nearly 1500 geographically dispersed contractor engineers and customer stakeholders." The architecture and design effort developed a hierarchy of models. Over a six-year period there were over 24,000 Trouble Reports for the SoS design effort alone. The number of changes that may be needed to address 24,000 Trouble Reports suggests a long information process, whose representations comprised "a large volume of artifacts," including trouble reports, requirements, architecture and design models, source code, contracts, schedules, budgets, communications supporting collaboration, meeting minutes, presentations, and notes.[6]

The amount and complexity of software production data can lead to information overload when project stakeholders try to use it to guide their decisions. “The nature of software [production] data is unique: it is heterogeneous, incomplete, evolving, and it deals in specifics that are typically not readily generalizable. While other fields that rely on data analysis have common data models for their domain, a software data model does not exist.... Today there exists only a superficial understanding of how software engineers and managers use data to make everyday decisions.”[7]

Obstacles to the study of software production data. Although there are examples of studies of software production data, there are many obstacles to the more thorough, scientific study the subject deserves.

- First, obtaining software production data is a particular challenge. While much open source software production data is available, it is not yet known how complete or accurate it is, nor how representative since the production practices

[5] J. Kirby, D. Weiss, R. Lutz, *Evidence-based Software Production*, Workshop on the Future of Software Engineering Research, 2010.

[6] *Model Driven Architecture and Design, Challenge Problem for the SPRUCE II Challenge Problem Population*, available from the SPRUCE Systems and Software Producibility Collaboration and Experimentation Environment, http://www.sprucecommunity.org.

[7] *Future of Software Engineering Research*, p. 46, Software Design and Productivity Coordinating Group, draft in preparation.

for commercial and particularly Government software may differ considerably. For example, in contrast to typical open source and commercial practices, “major DoD development projects are structured in a way that often keeps development teams at arm's length from the key operational mission stakeholders and from overall project management.”[8]

- Second, obtaining and sharing commercial and Government software production data requires the ability to protect personal, proprietary, and other sensitive data.
- Third, the collection and long-term storage and management of software production data needed for long-term and longitudinal studies of software production pose significant challenges. Some particular issues are representation of the data, determining and maintaining its integrity and provenance, and sustaining analysis and research efforts well beyond the duration of the typical research grant.
- Fourth, the multiple forms of software production data, e.g., formal, semi-formal, and unstructured natural language text, pose a research challenge to its integrated analysis.[9]

Vision. The fruit of meeting these challenges will be an economy in which a deep, evidence-based understanding of software production drives decisions at all levels regarding the creation and evolution of the nation’s critical software infrastructure. The availability of a “wide range of analysis techniques and methodology for software [production]..., which stakeholders can use to make data-supported decisions”[10] will facilitate better, faster software production decision-making based on evolving project data and a faster, cheaper, more responsive software production process. An evidence-based understanding of national needs and opportunities will guide software production research and education. The nation’s software production workforce will benefit from the resulting software production education and training. This workforce will employ a software production infrastructure comprised of tools, methods, and techniques that empirical studies will have refined and will have shown to be more effective and productive than alternatives. A result will be the creation of valuable, productive (hence, well-paid) jobs. Moving them abroad would require the replication of this software production infrastructure and of the evidence-based training and education. Further, this superior software production infrastructure will have beneficial effects throughout the economy. It will help make more competitive the many industries dependent on computer software for incorporating into products and for their manufacture. It will benefit an important component of the nation’s infrastructure of complex, large-scale, long-lived software, “simulation-based engineering and science (SBE&S)[, which] is a critical

[8] *Critical Code: Software Producibility for Defense,* p. 35, National Research Council, 2010.

[9] *Designing a Digital Future*, President’s Council of Advisors on Science and Technology, 2010.

[10] *Future of Software Engineering Research,* p. 47, Software Design and Productivity Coordinating Group, draft in preparation.

enabler of scientific discovery and innovation."[11] "[U]se of SBE&S by U.S. manufacturers is critical to creating and keeping good, high-paying jobs, strengthening and growing the U.S. manufacturing base, and in addressing critical 21st century problems...."[12]

Scope. Consistent, standardized, comprehensive software production data from across the economy will enable evidence-based assessment of the state of the practice across the economy and across critical subsectors and industries, facilitating the evidence-based focusing of software production research and education. This software production data must include data from the creations and evolution of free and open source software (FOSS), commercial software, government-developed software, and contractor-developed government software. It must be available for study to researchers at universities and at industrial and government laboratories. Software production organizations and projects will be able to obtain evidence-based assessment of the values of various methods, tools, and techniques, and to measure the impact of lost, missing, or degraded software production knowledge, and the impact of the geographic distribution of projects.

Important gaps. Currently, there are no systematic efforts to collect and analyze software production data from across the economy nor from important sectors or industries.

Research questions. Some research questions that could lead to better understanding of software production and to its improvement are:

1. What are the areas of knowledge that are critical to software engineering and how should we measure our effectiveness in defining and using them?
2. How do we standardize the collection of software measurement data across different organizations?
3. What will be the incentives for different organizations to collect the same types of data and provide them for analysis and archiving?
4. Who will be the keeper of the data?
5. How will data be made available to researchers and practitioners who want to use it in different ways?
6. Who will sponsor the research and development needed to answer the preceding questions?[13]
7. What sort of knowledge is lost when team members or whole teams leave a program and what is the impact of losing that knowledge on software and evolution?
8. How do we evaluate the impacts of changes to software production practices?

[11] *Simulation-Based Engineering and Science for Discovery and Innovation*, National Science and Technology Council, draft.

[12] Ibid.

[13] The first six items on the list are quoted from J. Kirby, D. Weiss, R. Lutz, *Evidence-based Software Production*, Workshop on the Future of Software Engineering Research, 2010.

9. There are many methods, methodologies, and techniques proposed in the software engineering literature. What are their impacts on projects creating and evolving large-scale, long-lived software intensive systems?
10. What factors determine when it is worthwhile to capture and maintain software production decisions, assumptions, and rationale?

Some first steps toward answering the above research questions that would provide near-term value and move us toward a data-driven software production process that is faster, cheaper, and more responsive to changing needs and technologies are to:

1. Investigate techniques and tools for distinguishing decisions, assumptions, and rationale from among the mass of formal, structured, and unstructured data recorded by artifacts comprising a representation of a software production information process.
2. Investigate techniques and tools for identifying changing decisions, assumptions, and rationale across a sequence of representations of a software production information process.
3. Investigate techniques and tools for identifying connections among decisions, assumptions, and rationale and among their changes.
4. Investigate techniques and tools for developing software data models based upon study of the software production information process.
5. Develop alternative software data models based upon study of the software production information process.
6. Investigate techniques and tools for visualizing and communicating at scale decisions, assumptions, and rationale, their connections, and their changes across a sequence of representations of a software production information process.

We ask for vigorous Big Data SSG, NITRD, and OSTP support for establishing funding of the study of software production data, as described above. Without such research, flaws in our software engineering discipline will remain unknown and unfixed, opportunities for significant improvement will be missed, and making confident and correct predictions about critical properties of our software will remain a great challenge. Without such research, “America’s security, economy, and quality of life,” dependent on software as a “critical enabler,”[14] remain at risk to software production discoveries and advances made elsewhere. The sooner we start such studies, the better.

[14] *Leadership Under Challenge*, p. 33, President’s Council of Advisors on Science and Technology, 2007.