Software Productivity Research In High Performance Computing

Susan Squires
Sun Microsystems Inc.
Michael L. Van De Vanter
Sun Microsystems Inc.
Lawrence G. Votta
Sun Microsystems Inc.

1. Introduction

The challenge of utilizing supercomputers effectively at ever increasing scale is not being met,[1] a phenomenon perceived within the high performance computing (HPC) community as a crisis of "productivity." Acknowledging that narrow focus on peak machine performance numbers has not served HPC goals well in the past, and acknowledging that the "productivity" of a computing system is not a well-understood phenomenon, the Defense Advanced Research Project Agency (DARPA) created the High Productivity Computing Systems (HPCS) program:[2]

- Industry vendors were challenged to develop a new generation of supercomputers that are dramatically (10 times!) more *productive*, not just faster; and
- A community of vendor teams and non-vendor research institutions were challenged to develop an understanding of supercomputer productivity that will serve to guide future supercomputer development and to support productivity-based evaluation of computing systems.

The HPCS Productivity Team at Sun Microsystems responded with two commitments:

1. Embrace the broadest possible view of productivity, including not only machine characteristics but also human tasks, skills, motivations, organizations, and culture, just to name a few; and
2. Put the investigation of these phenomena on the soundest scientific basis possible, drawing on well-established research methodologies from relevant fields, many of which are unfamiliar within the HPC community.

Team members brought expertise from multiple research fields with a specific focus on a sound working knowledge of concepts and methods appropriate to investigating human behavior. Socio-cultural concepts such as culture, ethnography, and social network analysis are not typically well understood in the computing community, sometimes leading to re-invention of methods already developed and validated in the social sciences. The first author is a social science professional with expertise in established practice. Other research-level expertise in the team included physics (both experimental and computational), software development (both technologies and human factors), and empirical software engineering. Given the breadth of the challenge and the small team size, a first principle was that every project demands careful – and quick – determination of appropriate outcomes, project constraints, and research methods. Conclusions must be founded in data and backed by justification for the design, execution, and application. (See also Kitchenham et. al. for general guidelines on conducting empirical research.)[3]

Social scientists have developed numerous methods that are both verifiable and reproducible in many contexts. However, the sheer number of methodological options makes it crucial that each project begin with clear research goals in order to identify the most effective combinations of concepts, research designs, information sources, and methods.

[1] Post, D.E., Votta, L.G. "Computational Science Requires a New Paradigm," *Physics Today*, 58(1):35-41.

[2] Defense Advanced Research Project Agency (DARPA) Information Processing Technology Office, High Productivity Computing Systems (HPCS) Program - http://www.darpa.mil/ipto/programs/hpcs/

[3] Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D. El Emam, K., Rosenberg, J. "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Transactions on Software Engineering*, 28:8, August 2002, pp. 721-734.

The research design presented in this paper is a three-stage framework, based on the scientific method, that allows the team to draw on multiple research disciplines appropriate to the phenomena under investigation. The framework is grounded in empirical data, validated by multiple approaches ("triangulation"), and applied to the practicing HPC professionals who actually perform the work being studied.

Research findings described in this paper must be understood in the context of the framework, as described in Section 2: definitions of the stages, methods used to collect and analyze information within each stage, and the relationships among the stages. Sections 3, 4, and 5 discuss how the framework was applied to studies of the HPC software development community. Although the productivity research program is still in progress, significant findings have already contributed to the community's understanding of HPC software development productivity.

2. A Three-stage Research Design

The research design is summarized in Figure 1. The stages are necessarily sequential; each provides a foundation for research methods in the next.

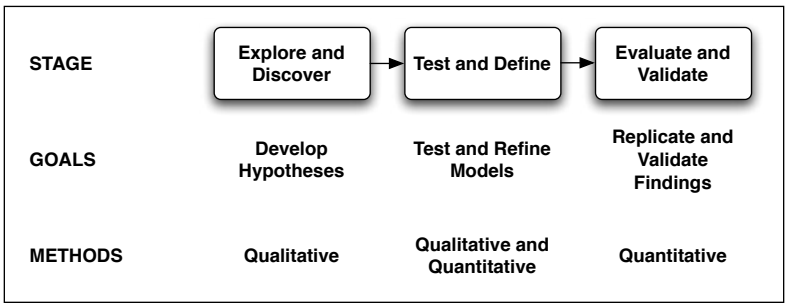

Figure 1. Research framework.

This framework is broadly analogous to realizations of the scientific method used in many disciplines, although the correspondence can be obscured somewhat by differences among the phenomena being studied and the methods appropriate to their study.

Stage 1: Explore & Discover. At the outset of an investigation researchers may not know the appropriate questions or issues to address, let alone have a coherent theory of the phenomena being studied. The first stage in the framework, based mainly on qualitative methods, is open-ended; it is designed to produce the insights necessary for hypothesis generation. Such insights can be considered an explicit "model" of the phenomena, analogous to the "paradigm" in which scientific inquiry is undertaken and through which experimental data are interpreted.

Stage 2: Test & Define. Rigor is added using methods that produce additional data surrounding theories generated in the first stage. This provides feedback on insights, supplies concrete data for model refinement, and leads to deeper understanding of what can be measured and how those measurements can be interpreted. This stage of "experimental design" is the necessary bridge between theory and experimentation.

Stage 3: Evaluate & Validate. Finally, more focused and mostly quantitative techniques are applied precisely in order to collect data, interpret the results, and validate the outcomes.

An important benefit of this framework is the reliable "roadmap" for moving from qualitative data collection and analysis, which are appropriate for discovery, to more quantitative data collection and analysis methods, which are appropriate to definitional and evaluative research.

The immediate challenge in a study of such breadth as software development is not so much in creating new methods, although researchers can be tempted to do so when investigating phenomena outside their area of training. A great many methods are available for studying human behaviors, the vast majority of which are known to graduate students who have taken class work in research design, advanced statistics, ethnographic interviewing, and content analysis.[4] The greater challenge is to determine exactly when, how, and why to use particular methods to draw out the implications of findings. This research framework makes it possible to select the appropriate qualitative and quantitative methods for data collection and analysis.

[4] Bernard, H. *Handbook of Methods in Cultural Anthropology*. Walnut Creek: Altamara Press. 1999.

Stage	Character	Methods
1. Explore & Discover	Open	Ethnography Case study Contextual observations Semi-structured Interviews Participation Document review Language patterning
2. Test & Define	Focused	Quasi-experimental studies Concept mapping Structured interviews Questionnaires Comparative studies Focus groups Semiotic analysis
3. Evaluate & Validate	Structured	Social network analysis Surveys Controlled experiments Product testing User-experience simulations Human testing Quality measurement

Table 1. Sample methods from the research toolbox.

Table 1 displays a sample of specific data collection and analysis methods appropriate to each stage, with the more qualitative methods appearing early and more quantitative methods appearing later. For example the case study with its open-ended qualitative observational and interview methods is most useful for early discovery research; on the other hand, surveys are appropriate in later stages when their design can be informed by data already collected. Data collection methods are just that: standardized techniques that enable researchers to gather valid and relevant information.

Qualitative and quantitative techniques are both useful, but only when used appropriately. For example, some methods are better to discern patterns, others to test specific ideas. Some methods are better to describe behavioral phenomena, others for opinions. Some methods excel at documenting how people interact with things, others at discerning how they interact amongst themselves. Some methods are better for discovering just how similar things or people are; others are better at teasing out causes from consequences. Qualitative data provide tremendous clues about people on a group level; examples of such "cultural" clues include descriptions of perceptions about how people use or classify objects, the nature of their personal interactions,

and opinions about the world around them. Quantitative data are also extremely useful, for example the number of lines of code (LOC) contained in an application or the number of full time personnel on a code project.

When techniques are combined appropriately, based on sound methodology, it becomes possible to create and distinguish among the most important human processes within and across groups of people.

3. Discovery Research, Stage 1: The Use of the Case Study Method

Discovery is the most open-ended and the most time consuming of the three research stages. The goal is to uncover and understand the socio-cultural system that frames human action; that understanding must be consistent with the way local people understand it and it must be expressed in terms of the local (emic) categories people use to describe and categorize their own reality. Researchers collect and analyze verbal, observational and contextual information to characterize what people say and do in their natural environment. Consistencies and, more frequently, inconsistencies help identify unarticulated or unrecognized needs, gaps and adaptations often called "work-arounds" and "disconnects." Translating disconnects into the frame of reference of socio-cultural systems allows the researcher to identify and neutralize well-established assumptions. Such assumptions would otherwise be taken as given, which leads to stereotypic treatment and precludes understanding of the important and difficult issues.

3.1 HPCS Stage 1 Methods

The key to a successful discovery in a context such as HPC software development is the combination of two approaches: case studies and rapid ethnographic methods.

The *case study approach* is a well understood method for gathering initial information about a situation, as defined by Robert K. Yin: "an in-depth look at one or more specific incidents or examples."[5] The case study typically employs a set of qualitative, open-ended methodologies to explore a topic or problem domain and develop hypotheses. Methods may include data collection techniques such as Document Reviews, Observation, Collection of Contextual Artifacts, Self-Reporting, and Interviews. The breadth of data that can be collected provides foundational knowledge for developing hypotheses. The case study approach has been used in software engineering[6][7][8][9] and has played a significant role in the HPCS productivity research program.[10][11][12][13][14]

The objective of *rapid ethnographic* assessment in discovery research is typically to construct a socio-cultural model of the local living system.[15][16] All rapid ethnographic approaches share three important characteristics "(1) a system perspective, (2) triangulated data collection, and (3) iterative data collection and analysis".[15][4] The value of the case study and rapid ethnographic assessment approaches is growing; they have been used by the National Center for Atmospheric Research,[17] Department of Energy,[18][19][20] NASA,[21] and for describing technical change at the Department of Defense.[22]

The application of these approaches stresses open-ended interviews, site tours (contextual observation), participant observation, literature reviews, cultural history, and semiotic (content) analysis. An example from the HPCS research will demonstrate how discovery research gen-

[5] Yin, R.K. *Case Study Research: Design and Methods. Sage Publications*, Second Edition, 1994.

[6] Card, D.N., Church, V.E., Agresti, W.W., "An Empirical Study of Software Design Practices," *IEEE Transactions on Software Engineering*, 1986. 12(2): 264-271.

[7] Müller, M.M. and Tichy, W.F. "Case Study: Extreme Programming in a University Environment," In *Proceedings of 23rd International Conference on Software Engineering*. May 12-19, 2001. pp. 537-544.

[8] Perry, D.E., Sim, S.E., and Easterbrook, S.M. "Case Studies for Software Engineers," In *Proceedings of 26th International Conference on Software Engineering*, ICSE 2004. pp. 736-738.

[9] Seaman, C.B. and Basili, V.R. "An Empirical Study of Communication in Code Inspections," In *Proceedings of 19th International Conference on Software Engineering*. Boston, MA. May 17-23, 1997. p. 96-106.

[10] Carver, J., Hochstein, L., Kendall, R., Nakamura, T., Zelkowitz, M., Basili, V., Post, D. "Observations about Software Development for High End Computing," *CTWatch Quarterly*, Volume 2, Number 4A, November 2006 – http://www.ctwatch.org/quarterly/

[11] Kendall, R., Carver, J., Mark, A., Post, D., Squires, S., Shaffer, D. "Case Study of the Hawk Code Project," *Los Alamos National Laboratory Report* LA-UR-05-9011, 2005.

[12] Kendall, R., Post, D., Squires, S., Halverson, C. "Case Study of the Condor Code Project," *Los Alamos National Laboratory Report* LA-UR-05-9291, 2005.

[13] Kendall, R., Post, D., Squires, S., Carver, J. "Case Study of the Eagle Code Project," *Los Alamos National Laboratory Report* LA-UR-06-1092.

[14] Post, D., Kendall, R., Whitney, "Case Study of the Falcon Code Project," *Proceedings Second International Workshop on Software Engineering for High Performance Computing System Applications*, St. Louis, 15 May 2005.

[15] Beebe, J. "Basic Concepts and Techniques of Rapid Appraisal," *Human Organization*, 54(1): 42-51. 1995.

[16] Trotter, R., Schensul, J. "Methods in Applied Anthropology," in *Handbook of Methods in Cultural Anthropology*, H. Russell Bernard (ed.), Walnut Creek: Altamara Press. 1999.

[17] Lahsen, M. "Seductive Simulations: Uncertainty Distributions around Climate Modeling," *Social Studies of Science* 36(6): 895-992. December 2005.

[18] Gusterson, H. *Nuclear Rites: A Weapons Laboratory at the End of the Cold War*, University of California Press: Berkeley. 1996.

[19] Gusterson, H. *People of the Bomb: Portraits of America's Nuclear Complex*, University of Minnesota Press: Minneapolis. 2004

[20] McNamara, L., Trucano, T. "So Why Do You Trust That Model? Some Thoughts on Modeling, Simulation, Social Science and Decision Making," *Advanced Concepts Group News and Views,* 8:2. Albuquerque, NM: Sandia National Laboratories. 2006.

[21] Shalin, V., Wales, R., "Shift Handovers in ISS Mission Control," in *Human Organizational Risk Management Workshop,* NASA-Ames April 25-27 2001.

[22] MacKenzie, D. "Missile Accuracy: A Case Study in the Social Processes of Technological Change," in *The Construction of Technological Systems*, Wiebe Bijker, Thomas Hughes and Trevor Pinch (Eds) MIT Press: Cambridge MA. 1987.

erates cultural insights; those cultural insights lead to better understanding and help develop hypotheses that can be tested in the subsequent research stage.

3.2 Example: The HPC "Expertise Gap"

Anecdotal evidence from DARPA and the HPCS Mission Partners suggested that an "expertise gap" lay at the heart of the crisis in HPC application development.[23] Case studies were conducted first to explore the expertise issue, starting with a detailed look at how professional HPC programmers and teams spend their time. Qualitative data collection methods included semi-structured interviews with individual HPC programmers, and contextual observations at sites in which HPC programmers work. Of course the raw data from these methods did not directly lead to the kind of insights that are the goal of this research stage. Combining and comparing the data, the team began to identify patterns across individuals and teams, plot bottlenecks and create models of HPC programmers, all based on information taken directly from the HPC professionals and the context of their work.

[23] Sarkar, V., Williams, C., Ebcioglu, K. "Application Development Productivity Challenges for High-End Computing," *First Workshop on Productivity and Performance in High-End Computing*, Madrid Spain, pp 14-18. February 2004.

From five case studies a pattern emerged in the area of expertise. In all cases at least one founding team member had been recruited for special knowledge of science, but in each case the scientist was not an HPC programmer and had little or no knowledge of FORTRAN or C++. The scientist's first required task was either to learn one of the programming languages or to build a working relationship with someone who did know it. In either case, the educational process took considerable time before the individual/pair could perform effectively. Project management was typically taken on by another person whose role was to "run interference" by keeping the sponsor happy and negotiating for time on a shared large machine. Teams in this context typically take about four to six years to get a working code. Success is commonly attributed to having the right mix of expertise. The team was successful only when they had the appropriate mix of knowledge, represented by four areas:

- Science
- Programming
- Scaling / Optimizing
- Management

Even having the range of knowledge is insufficient. Effective communication and collaboration among the experts can be very challenging and is crucial to project success.

Underlying the ethnographic case study approach is the understanding that all people belong to one or more networks of interlocking social relationships in which members share a common or core set of beliefs, values and behaviors. Anthropologists and other trained ethnographers use various methods to uncover the core sets of beliefs, such as:

- Gathering individual (emic) perspectives from members of these socio-cultural groups;
- Examining the collected information to identify patterns of shared beliefs, behaviors, values and rules;
- Constructing group "mental models" from identified patterns to understand the meaning at the core of the system; and
- Interpreting how the members of a socio-cultural network use their mental models to construct and express appropriate shared behaviors, beliefs, and values, to provide a contextual frame of meanings for products, and services.

The case studies of HPC software development led to recognizable patterns that might explain why HPC expertise is so scarce. A hypothesis was developed postulating that domain specific expertise in at least four different areas is needed to use highly parallel machines. As machines get bigger and more complex, the pool of experts narrows. Very, very few people have complete skill sets. Team approaches are the best strategy at the moment, but this by itself does not appear to represent a long-term solution. The next research step was to craft a more focused study to test the hypothesis and to understand in more detail when and how the various areas of expertise were used; this takes place in the second stage of the research framework.

4. Definition Research, Stage 2: Combining Qualitative with Quantitative Measurement

Definition Research helps test an idea or hypothesis focusing the research on more detailed use, use features, and meaning associated with activities and helps define work models and workflow. The methods used during Definition Research differ from Discovery because of what has already been learned during Discovery: parameters of the topic, the concepts, and something about the individuals. Definition Research usually starts with a series of questions, based on some grounded hypothesis generated during discovery. For example, researchers investigating programming techniques can compare existing codes in context to make inferences about how proposed changes might change both the programming work and the results.

Definition research concentrates on details, so any interviews conducted in this stage are more structured than during Discovery. These interviews follow a set of well-understood rules; for example the interviewer builds rapport in the first segment and subsequently seeks deeper information. Details are summarized at the conclusion of the interview in order to confirm the data with the respondent. Verbal and written statements are validated through observed actions. In order to fully understand the specifics of the context, researchers listen for native language: words, terms, and descriptions.

4.1 HPCS Stage 2 Methods

An advantage of both Discovery and Definition Research is that relatively few cases are needed to discern relevant cultural patterns and to learn about shared understandings and behaviors in a group. This approach has powerful implications for quantifying human behavior patterns in ways never imagined a few years ago. Notre Dame mathematician Albert-Laszlo Barabasi recently wrote in *Science* that our grouping ability to collect data about human actions combined "with the sophisticated tool of network theory, . . . (provides) a glimpse of an unprecedented opportunity to quantify human dynamics."[24] Social network analysis can take millions of bits of data and construct reliable and predictive human patterns. But such an approach can begin with small data sets as well.

[24] Barabasi, A.L. "Network Theory – the Emergence of the Creative Enterprise," *Science*, 308(29): 639-650. April 2005.

Mathematician Duncan Watts points out that "the world that we live in is not at all random. We are very much constrained by our socioeconomic status, our geographical location, our background, our education and our profession, our interests and hobbies. All these things make our circle of acquaintances highly nonrandom."[25] Watts and fellow mathematician Steven Strogatz are among a growing number of researchers who have been examining highly structured social networks in order to understand and use mathematical formulations to predict membership connectiveness. Their work has been inspired by, and extends the work of, the theoretical mathematician Paul Erdös, who has been indirectly responsible for popularizing

[25] Watts, D., Strogatz, S. Interview in *Discover.* 1998.

the idea of six degrees of separation: the idea that only six other individuals link all humankind through their social networks to almost everybody else in the world.

Although Watts and Strogatz are most interested in the interconnectiveness of social networks, many others such as Borgatti and Everett[26] (see also *Mathematical Social Sciences* and *Social Networks*) focus on the highly nonrandom nature of social networks. Their goal is to devise statistically reliable mathematical formulae that predict the number of individuals who need to be interviewed in order to capture the shared characteristics of social networks: shared beliefs, values and behaviors. The analysis of Handwerker and Wozniak suggests the surprising low number of seven,[27] although this number is reliable only when certain criteria are met:

1. The information gathered is about shared or core understandings within the social network. This is not about group variation.
2. The cognitive domain of the social network is internally consistent and ordered.
3. Information is gathered from key members of the social network: cultural experts.
4. Informants are interviewed independently of others in their social network. Informants must not be allowed to confound information by checking or comparing notes with other members.
5. There are no known divisions within the social network: no sub-groupings that might have distinct sets of core knowledge and behaviors.[27]

One way to determine if a social network is bounded is to analyze patterns that emerge from the data. Highly redundant information about membership of a group and their perceptions of cultural norms are strong evidence that there is consensus about who is in the group and what they consider appropriate. However, if any of the criteria are not met, then another individual must be interviewed. Once a consensus is identified, then it is recorded as a discovery and the researcher moves on. In our Discovery research we were able to identify these shared work patterns with a high level of confidence, making it possible to identify potential participants for the following research stage.

One of the HPCS goals during the Definition research stage was to understand the workflow of HPC programmers and to identify how and where current HPC development paradigms limit code development. For example, a hypothesis from the Discovery stage was that one limiting factor is the level and range of expertise needed. Definition research was designed to validate the hypothesis and, if validated, shed additional light on the development process: where specific expertise was deployed and where programmers encounter significant time and effort bottlenecks.

4.2 Example: HPC Workflow

Two methods were chosen to collect data: one quantitative, one qualitative. Quantitative information on programmer activity (time on task details) was collected using HackyStat, an in-process software engineering measurement and analysis tool.[28] HackyStat recorded hours of event traces from development tools (for example "open file," and "build") while HPC professionals developed code. Additional (qualitative) data was collected in the form of real-time, time-stamped journals written by professional code developers who agreed to record a personal narrative of their work.

[26] Borgatti, S., Everett, M.G. "Network Analysis of 2-mode Data," *Social Networks*, 19(3): 243-269. 1997.

[27] Handwerker, W.P., Wozniak, D. "Sampling Strategies for the Collection of Anthropological Data: An Extension of Boaz's Answer to Galton's Problem," *Current Anthropology*, 38(5): 869-875. 1997.

[28] Johnson, P., Paulding, M. "Understanding HPC Development through Automated Process and Product Measurement with HackyStat," *Second Workshop on Productivity and Performance in High-End Computing* (P-PHEC), San Francisco, Feb. 13, 2005.

By combining HackyStat telemetry data that measured activity with programmer journals, the team was able to corroborate, validate, and interpret results. For example, the significance of the expertise gaps and bottlenecks to the HPC productivity problem became apparent when studying individual professionals and their time usage; the journal entries represented real-time accounts of code development from the programmer's perspective. These patterns were used to define a typical HPC development workflow, identify where in the workflow the most effort is being expended, characterize the expertise profiles associated with workflow tasks, and draw conclusions about productivity bottlenecks and their root causes.

These results are summarized in Figure 2, which illustrates the typical workflow that developers go through in creating and optimizing HPC applications along with the skill sets required to perform each activity. A typical workflow includes understanding the problem that needs to be solved, formulating an initial computational solution, empirically evaluating the proposed solution through prototyping or experimentation, coding for sequential execution, evaluating the overall computational approach, then coding and optimizing the results for a parallel platform.

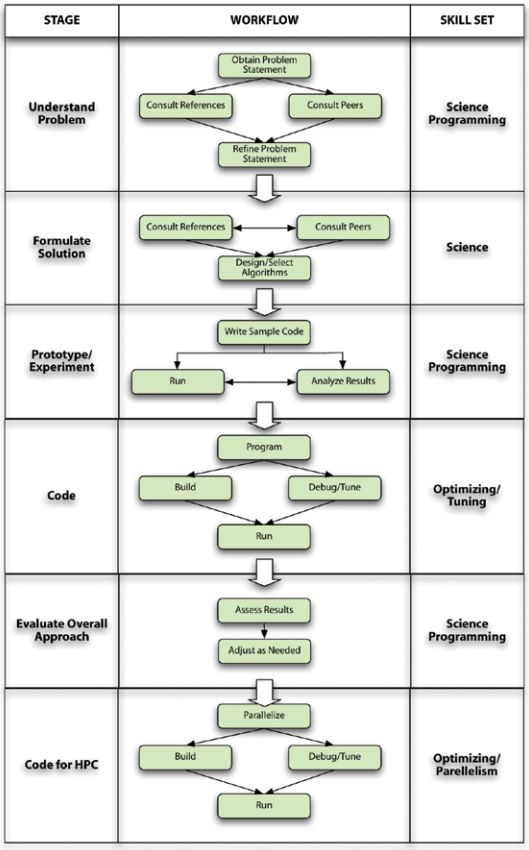

Figure 2. Work Model of HPC Programmers.

Activities that consume the greatest proportion of resources, effort, time, and expertise, within the overall programming effort were also identified:

- *Developing correct scientific programs:* activities associated with translating an understanding of the scientific problem that must be solved (e.g., a predictive weather model) into code.
- *Code optimization and tuning:* activities associated with refining a serial version of the code to ensure correctness and achieve desired levels of accuracy and efficiency.
- *Code parallelization and optimization:* activities associated with parallelizing the code and tuning to achieve high machine utilization and rapid execution.
- *Porting:* where a solution exists, this comprises the activities associated with translating the existing solution to a representation appropriate for a new computing platform.

Once the expertise problem was understood and the likely location that consumed the most time and effort for those experts was pinpointed, the question of how widely these findings could be applied was raised. Full validation required mapping the extent of the expertise gap, calling for a large quantitative survey of the sort suitable for the next stage of the research framework.

5. Evaluative Research, Stage 3: Developing Quantitative Models for validating HPCS Programmer Workflow

Like rapid ethnographic research, evaluative research has its own history that can be traced back to the middle of the 20th century. In 1967 Michael Scriven proposed that all evaluation could be broken down into two distinct types, formative and summative evaluation.[29] Formative evaluation validates and improves upon an idea or hypothesis. Summative evaluation answers the question, "to what extent?" Evaluative researchers use a toolbox of methods from many of the social sciences to validate a hypothesis or determine its extent in a population. Methods are typically quantitative. Again the HPCS research provides an example, although this phase of the program is in very early stages and the results are still preliminary.

[29] Scriven, M. "Beyond Formative and Summative Evaluation," In G. W. McLaughlin & D. C. Phillips (Eds.) *Evaluation and Education: At Quarter Century.* Chicago, IL: University of Chicago Press, pp. 19-64. 1991.

One of the patterns that emerged from case study research suggested that HPCS code teams were more concerned about programming correctness than performance. The workflow pinpointed the most likely places where a programmer would find the most difficulty. Up to this point the conventional wisdom had been accepted without question, namely that performance was the paramount concern of the programmer. To evaluate the extent of this pattern, which was uncovered in Stage 1 and 2, a survey was administered to HPC programmers at National Labs and in private institutions where large, highly parallel code is written. Quantitative statistical procedures were used to analyze the survey data. Table 3 provides the resulting response distribution when asked about the top issues facing the HPC programmer.

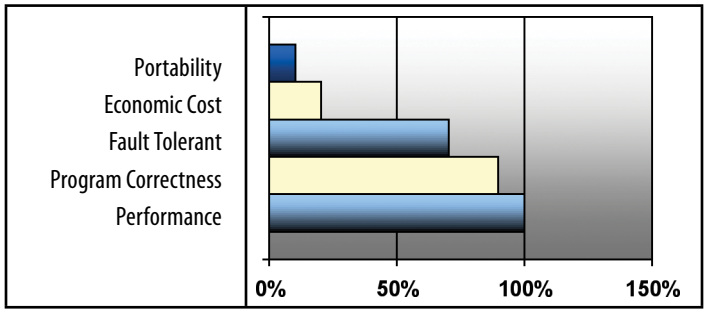

Table 2. Response Distribution of Top Issues.

The case studies in the first stage of this research had provided information about what programmers said and the empirical studies of programmers in the second stage supported the findings from the earlier research and validated the workflow of the programmer with the quantitative data. Not surprisingly, the survey data collected in the third stage of the research confirmed that performance is important in HPC code, but also confirmed that programming correctly is at least as important, as reported by 90% of those surveyed. The perceived value of performance is a strong shared value in the HPC community; however, the concern about correctness appears to be at least as strong although not verbalized as often.

This example highlights the need to follow the research stages from hypothesis development (Stage 1) and question generation (Stage 2) before attempts to quantify. Unfortunately there is a tendency to jump to quantitative research because of the supposed reliability of numbers. However, quantitative results are only as good as models of the phenomena that are the context for interpretation of the data. In the example, without having proceeded through the logical progression from hypothesis to validation, the significant concern for program correctness might have been overlooked. And, of course, the link between correctness and need for expertise is

clear. Such an oversight might have led to hardware and software design decisions that turn out to be counterproductive in the area of program correctness.

6. Conclusions

The three research stages and associated methods described in this paper have immense potential to increase understanding of software development, both in the HPC community and beyond. Research results to date include fundamental discoveries about productivity.[30] [31] [32] [33] These findings are grounded in empirically validated models that reflect the experience of practicing HPC professionals.

We are just beginning to understand how central to the efforts of HPCS research are the essential and intimate relationships among people, tools, and code: independent changes in each are unlikely to produce the dramatic 10x increase in software productivity that was envisioned by the founders of the DARPA HPCS program and which is desperately needed by the HPC community. Meeting that goal demands aligning those changes around a deep understanding of what makes software development productive: for machines, for individuals, for organizations, and for communities. As the technology historian, Kingery noted, "... No one denies the importance of things, but learning from them requires rather more attention than reading texts."[34]

Acknowledgments
We would like to thank our HPCS colleagues at Sun Microsystems and elsewhere in the HPC community for their helpful discussions and comments.

[30] Loh, E., Van De Vanter, M. L, Votta, L.G. "Can Software Engineering Solve the HPCS Problem?" in *Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications*, St. Louis, 15 May 2005.

[31] Squires, S., Tichy, W.F., Votta, L.G. "What Do Programmers of Parallel Machines Need? A Survey," *Second Workshop on Productivity and Performance in High-End Computing* (P-PHEC), San Francisco, Feb. 13, 2005.

[32] Squires, S., Van De Vanter, M. L, Votta, L.G. "Yes, There Is an 'Expertise Gap' in HPC Application Development," *Third Workshop on Productivity and Performance in High-End Computing* (P-PHEC), Austin, Feb. 12, 2006.

[33] Van De Vanter, M. L, Post, D.E, Zosel, M. "HPC Needs a Tool Strategy," in *Proceedings of Second International Workshop on Software Engineering for High Performance Computing System Applications*, St. Louis, 15 May 2005.

[34] Kingery, W. (Ed), Editor's Preface, *Learning from Things: Method and Theory of Material Culture Studies*. Washington, D.C.: Smithsonian Institution Press. 1996.