Overview of Software Challenges in CSE

Roscoe A. Bartlett, Oak Ridge National Laboratory

10/08/2013

Introduction

The Computation Science and Engineering (CSE) community is facing a significant set of challenges on many fronts with respect to the development and manipulation of CSE software. The primary goal of this short paper is only to identify, categorize and give name to these various challenges to help frame the possible solutions and research directions.

SE Challenges in CSE

This overall problem is composed out of the aggregation of a number of challenges in different areas which include:

- **The challenge of expanding complexity in CSE algorithms**: CSE algorithms are becoming more complex with greater varieties of algorithms being produced by different organizations and a large amount of this technology must be integrated together for the greatest effect.
- **The challenge of expanding complexity in CSE/HPC computer architectures**: Exascale architectures will be less reliable and require more specialized algorithmic and programming techniques.
- **The challenge of CSE developer software knowledge and skills**: There is a deep and fundamental deficit of basic software knowledge and skills in the CSE community that hinders even basic research. This applies at every level from hands-on developers to decision makers at the highest level and in between.
- **The challenge of programming languages and supporting tools for CSE**: The current generation of programming languages, supporting tools, and approaches do not well support the current CSE community and will not adequately address the growing complexity in algorithms & architectures.
- **The challenge of CSE software quality, verification and testing**: The current CSE community is not creating codes with sufficient verification and validation foundations to provide credible results (and this challenge can only be addressed if the other challenges are also adequately addressed).
- **The challenge of CSE software life-cycle and integration models**: The current CSE community lacks sufficient knowledge, skills, and discipline in software engineering life-cycle and software integration processes to address the growing complexity in multi-institution algorithms and architectures work.
- **The challenge of CSE software project management**: Most CSE projects are organized by sponsors that don't understand the realities of high-risk technically challenging software development which leads to incorrect perceptions of success and failure that damages every aspect of the CSE community.

In the end, all of the efforts in CSE are ultimately directed towards the goal of creating a new generation of CSE codes that will be able to more accurately predict the behavior of more complex phenomena at finer fidelity with greater accuracy and credibility. More complex algorithms of a greater variety are constantly being developed in order to overcome some existing bottleneck in performance or capability. The best work in algorithms and architectures must all be integrated together into single CSE codes in order to be able to meet the full potential of CSE in creating predictive CSE tools. These capabilities will be critical in order to produce credible results that can then be used to make critical decisions in climate, energy, and the many other areas that will critically impact coming generations. Any one single organization can not afford to hire the expertise to create and maintain software for the leading methods in all of these areas and therefore multi-institution collaboration and reuse will be necessary.

Background and related work

Some work has been done to study the software engineering (SE) issues in CSE and to characterize the SE challenges. One set of authors warned of a coming crisis in computational science [10] and that computational science demands a new paradigm [11]. These and some related papers were based on a number of case studies of projects in the CSE community [5, 4, 3]. These papers defined and described three types of challenges faced by CSE a) *the performance challenge* (producing high-performance, computers), b) *the programming challenge* (programming for complex computers), and c) *the prediction challenge* (developing truly predictive complex application codes). The main focus of these papers was on the prediction challenge where the authors argued for better methods and more effort and discipline in basic verification and validation (V&V) and to some lesser extent argued that software engineering practices in CSE projects needed to be improved to meet the prediction challenge. However, one can argue that without addressing the other challenges listed above, the CSE community will never be able to adequately address the prediction challenge. Other work has also been done to characterize the makeup of the current CSE community [5, 4, 3, 8]. These papers describe the large "communication gap" that exists between the CSE and SE communities [6]. Many of these papers also characterize a large productivity problem and an expertise gap [6]. This is consistent with with the challenges listed above. Also, a number of workshops have have been organized bringing together individuals from the SE and CSE communities, for example [2].

Summary and Research Opportunities

The CSE community needs to recognize and address the various SE related challenges facing the CSE community outlined in this short paper. Recognition of these challanges can then help to organize and prioritize research into possible solutions. The most general approaches should likely be to carefully apply modern Lean/Agile SE technical and management methods to CSE projects ([9, 7, 12, 1]. Core Agile practices include continuous integration (CI), test-driven development (TDD), (continuous) design improvement, collective code ownership, and coding standards.

References

[1] K. Beck. *Extreme Programming (Second Edition).* Addison Wesley, 2005.

[2] J Carver. Report from the second international workshop on software engineering for computational science and engineering (se-cse 09). *Computing in Science Engineering*, PP(99):1 –1, 2009.

[3] J. Carver, L. Hochsein, R. Kendall, T. Nakamura, M. Zelkowitz, V. Basili, and D. Post. Observations about software development for high end computing. Technical report, Univ. of Marlyland, 2005.

[4] J. Carver, R. Kendall, S. Squires, and D. Post. Software development environments for scientific and engineering software: A series of case studies. *ICSE07*, 2007.

[5] Stuart Faulk, Eugene Loh, Michael L. Van De Vanter, Susan Squires, and Lawrence G. Votta. Scientific computing's productivity gridlock: How software engineering can help. *Computing in Science Engineering*, 11(6):30 –39, nov.-dec. 2009.

[6] Stuart Faulk, Eugene Loh, Michael L. Van De Vanter, Susan Squires, and Lawrence G. Votta. Scientific computing's productivity gridlock: How software engineering can help. *Computing in Science Engineering*, 11(6):30 –39, nov.-dec. 2009.

[7] R. Martin. *Agile Software Development (Principles, Patterns, and Practices).* Prentice Hall, 2003.

[8] Zeeya Merali. Why scientific computing does not compute. *Nature*, 467:775 –777, 2010.

[9] M. Poppendieck and T. Poppendieck. *Implementing Lean Software Development.* Addison Wesley, 2007.

[10] D. Post. The coming crisis in computational science. Technical report LA-UR-04-0388, Las Alamos Laboratories, 2004.

[11] D. Post and L. Votta. Computational science demands and new paradigm. *Physics Today*, 58(1):35–41, 2005.

[12] K. Schwaber and M. Beedle. *Agile Software Development with Scrum.* Prentice Hall, 2002.