

ULRR

Global software engineering education practice continuum special issue of the ACM transactions on computing education

Item Type	Article
Authors	Clear, Tony;Beecham, Sarah
Citation	ACM Transactions on Computing Education;19 (2) article 7
Publisher	Association for Computing Machinery
Download date	2026-04-15 14:14:14
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/8396

Global Software Engineering Education Practice Continuum Special Issue of the ACM Transactions on Computing Education

TONY CLEAR, Auckland University of Technology, New Zealand

SARAH BEECHAM, Lero—The Irish Software Research Centre, University of Limerick, Ireland

We are pleased to introduce this Special Issue on Global Software Engineering Education published by the ACM Transactions on Computing Education (TOCE) that focuses on educational practices to prepare students for a global workplace. This issue comes at a time when universities are recognizing the need to provide courses that address the challenges of distributed development and presents research that will facilitate course leaders currently running, or embarking on, Global Software Engineering Education (GSE-Ed).

ACM Reference format:

Tony Clear and Sarah Beecham. 2018. Global Software Engineering Education Practice Continuum Special Issue of the ACM Transactions on Computing Education. *ACM Trans. Comput. Educ.* 19, 2, Article 7 (January 2019), 8 pages.

<https://doi.org/10.1145/3294011>

1 INTRODUCTION

This Special Issue was sparked by recent interest in GSE-Ed evident at a workshop held in August 2016 at the International Conference on Global Software Engineering (<http://www.icgse.org>), which, in turn, was triggered by an ITiCSE Working Group Systematic Literature Review report of the field [1]. GSE-Ed research looks at how university-based software engineering courses can meet the needs of industry. The studies in this area suggest that conventional approaches to teaching SE are increasingly outdated and lack authenticity. There is an urgent need for well-trained global software engineers, since the majority of development jobs now involve some level of globalization, where individuals from different cultures, geographies, and time zones must collaborate, share tasks, and share knowledge. In order to remain competitive and increase productivity, software organizations are practicing Global Software Engineering (GSE). GSE promises organizations certain advantages, such as access to a larger skills base, new markets, and reduced labor costs. Yet, GSE projects often fail to realize hoped-for advantages, and research shows that this is largely due to Global Distance (where teams suffer overheads associated with operating across temporal, geographic, and cultural distances)—a symptom of distributed teamwork. Organizations are attempting to improve their productivity by reducing global distance, improving individual motivation, and strengthening team collaboration through careful task allocation [2], cultural and interaction training [3], and scaling agile methods [4]. Preparing tomorrow's engineers for the

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero—The Irish Software Research Centre (www.lero.ie).

Authors' addresses: T. Clear; email: tony.clear@aut.ac.nz; S. Beecham; email: sarah.beecham@lero.ie.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

© 2018 Copyright held by the owner/author(s).

1946-6226/2018/01-ART7

<https://doi.org/10.1145/3294011>

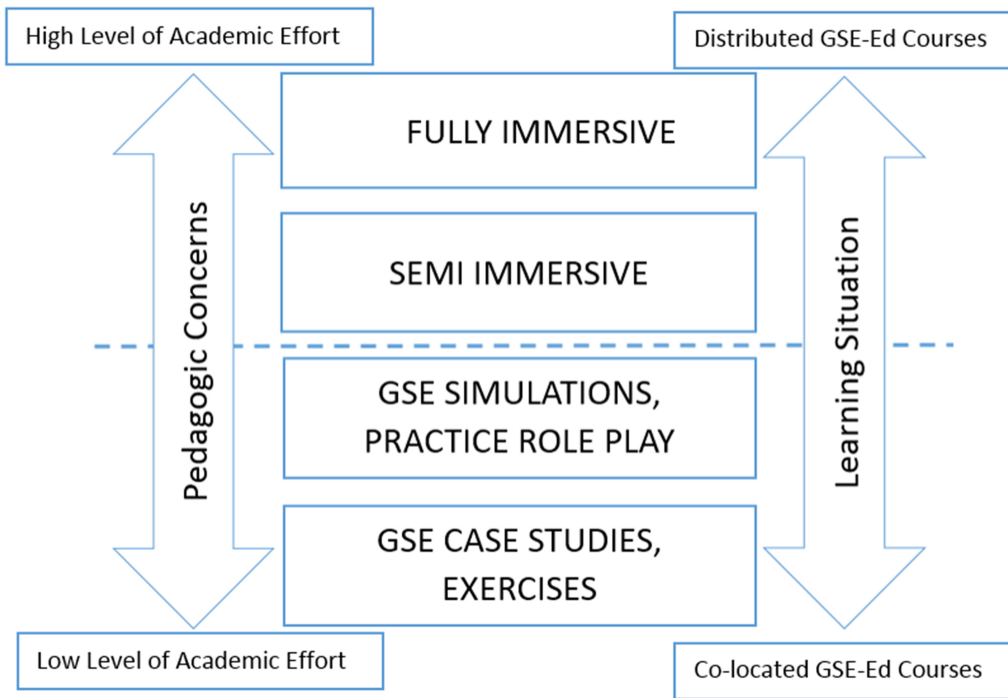


Fig. 1. A continuum model of global software engineering education.

complexities of GSE is therefore imperative and requires more than producing engineers who can develop code.

The article selection process started in 2016, when we solicited abstracts to review for suitability for the special issue. We received 28 abstract submissions from which 15 were invited to submit full articles, and 12 submissions were eventually received. Following two cycles of double-blind peer review by up to four reviewers, we finally selected eight articles for publication in this bumper special issue. We were pleased with the care taken by the reviewers, who we thank for their time, and to the authors in their responses, and the variety and quality of the articles we present to you here. Through the collection of articles that follow, we aim to provide sound examples of courses and initiatives for GSE-Ed and identify new directions for future research and education.

2 GLOBAL SOFTWARE ENGINEERING CONTINUUM

In framing the set of articles in this special issue, we sought to classify them in some meaningful way. Drawing on the debate on relative merits of three candidate approaches to the teaching of GSE-Ed [5] and on the Work Integrated Learning continuum outlined in Clear et al. [6], we developed the framework above as a continuum model of Global Software Engineering Education.

The continuum model shown in Figure 1 depicts a progression starting with lower instructor effort co-located courses and components, which aim to provide GSE-Ed experiences through *case studies, exercises, role-play* and *simulations*, in the classroom. Moving up the continuum we see half-way houses of *semi-immersive* experiences provided through Open Source Software Development team projects. At the top of the continuum sits the model most demanding on instructors and with the highest degree of distribution, where we see fully distributed courses delivered through a collaboration among several institutions. This special issue adopts this framework as an organizing vehicle and presents the articles in descending order from *fully immersive* to *simulations* and finally classroom *exercises*.

3 A CONTINUUM OF EIGHT GSE-ED SPECIAL ISSUE ARTICLES

3.1 Fully Immersive GSE Courses

The first of our articles fits the description of a *fully immersive* course, where students develop software in projects that span different institutions, countries and time zones. Ivana Bosnic et al.'s impressive longitudinal study places the student at the center of the investigation and traces their progress over time to the point where we learn about the value of such courses to those students who now work as professional software engineers in large and small organizations. The inclusion of "[Assessing the Impact of the Distributed Software Development Course on the Careers of Young Software Engineers](#)" in our special issue presents a rare opportunity to understand what students think about these immersive courses from a real-world professional perspective. We learn about the relevance of the course, and what the course offers over and above other software engineering courses (which presumably are of the standard co-located project team variety). The results are encouraging in addressing several known GSE problems to include strategic, cultural, communication, knowledge management, project and process management, and technical issues.

To understand the impact of the distributed software development (DSD) course, 79 former DSD course students (mostly now employed in the Croatian IT industry—in companies representing a range of sizes and levels of globalization)—completed a survey in which they were asked a range of questions related to the relevance of the DSD course. Bosnic and colleagues present many compelling and motivating results. The benefits that emerge include the ability to demonstrate new skills to prospective employers, and giving students an edge and extra confidence during an interview. Also, soft skills learned as a by-product were also important. An unexpected finding was that the course had a social impact, in which working with students from other universities led to long-lasting friendships and even a start-up. We encourage readers to read this article to find out more about what can be learned from conducting these *fully immersive* distributed software development courses, both from a project management, technical, and the all-important soft skills viewpoint. Developing an engineering way of thinking, problem-solving, and wide theoretical technical knowledge were deemed the most useful knowledge/skills learned.

Another strength of this first article is that the authors share their extensive experience of running distributed courses over 15 consecutive years. They observe, for example, a change in the organizations that their students join, which many years ago started with 25% of students joining distributed organizations, to now, after graduating, where practically all students end up working for global companies however small. This suggests that GSE-Ed courses are becoming even more important, and hence, the kind of advice offered in this special issue can help motivate all software engineering undergraduate courses to change and update their current offerings.

The second in our list of articles, "[Assessing Students' IT Professional Values in a Global Project Setting](#)," by Frezza et al., is set in the context of a *fully immersive* course, *IT in Society*, a collaboration between Uppsala University in Sweden, and Gannon University and Rose Hulman Institute of Technology in the United States. Working with a hospital in Uppsala as their client, students form a whole cohort working jointly on an Open Ended Group Project [7] in which, given a considerable scope for action, they ideate and frame the objective with their client and then work as a group toward a resolution. In this article, Stephen Frezza and colleagues investigate how students and alumni of the course perceive professional values in the quest of developing an instrument that can assess the extent to which these have been developed in the course. So while the context is *fully immersive*, the focus is on developing professional values as a consequence of exercising *professional practices*. This is important work for those of us who proudly state the graduate attributes that our students are meant to possess on completion of their degrees. Often these attributes are expressed in terms of broad competences such as "learning how to learn" [8]. Competences have

been expressed in Ref. [9] as comprising “knowledge + skills + dispositions”. Of these, “dispositions” are the most problematic, representing a mindset and tendency toward action. So to what extent they can be effectively assessed remains a somewhat open question, cf. Ref. [10].

In a carefully designed process, the authors draw on relevant literature, which has bearing on Global IT professional values, to develop a set of statements to deconstruct and crisply reflect specific values.

Developing a set of Thurstone scales, they build a set of actionable value statements which represent behavioral aspects of an agreed set of values, developed and refined by an expert team of 12 colleagues. This led to an instrument with the aim of assessing student valuing of global IT professional values. The validated instrument was emailed in a survey to both current students and alumni of the course, with some 26 responses. Student responses were broadly in agreement with the experts’ views suggesting an alignment of perspective on IT Professional values. In a comparison of their own values with those of the team, there were some interesting differences (e.g., “forgives teammates for technical and cultural mistakes”), suggesting the modeling of these values by the team gave students insight into their own values. The authors acknowledge that this work was exploratory, yet the instrument appears “inexpensive and reusable, and consequently shows promise for use in the development and use of values-centered student learning outcomes.” As an article investigating approaches to assessment of the affective as opposed to the more traditional cognitive domain, this novel work charts an intriguing path toward the assessment of global IT values.

In article three, “[Managing Diversity in Distributed Software Development Education—A Longitudinal Case Study](#),” Ivana Bosnic and colleagues discuss the diverse range of issues to be addressed in conducting *fully immersive* GSE courses. They draw on 14 years’ experience in running a distributed course across three European Universities, combining Action Research and drawing upon longitudinal case study data. The results are presented based on a three-layer model—the institutional layer, the teaching layer, and the project layer—each of which has its own diverse dimensions, which may become amplified across sites, and which are captured in a crisp set of key takeaways. While concrete and practical strategies to address the issues identified are provided, the authors also acknowledge the multi-layered nature of the diverse challenges that will be encountered in a *fully immersive* GSE course. Some of these institutional differences such as enrollment systems, grading systems, and course volatility cannot be removed but need strategies to mitigate them. Likewise, the number of participating students and disproportionate team numbers across sites cannot be avoided. So, in many cases, a balancing act is required. A key conclusion is that within the teaching teams, continuous interaction is critical, as well as a willingness to be open to others’ opinions and ideas.

3.2 Semi-Immersive GSE Courses

Article four by Isomöttönen and colleagues asks the question “[Can Students Capitalize on Enabling Learning Environments?](#)” in their search for global employability. Isomöttönen et al.’s article focuses on creating “enabling” learning environments that prepare students for the global workplace. Also in common with Ivana Bosnic and colleagues’ article on course impact, appearing first in our Special Issue, Isomöttönen et al. have their lens firmly fixed on creating courses that give students the opportunity to think for themselves. In the context of our course structure continuum, we have termed this article as *semi-immersive* since the empirical study is designed around student-driven project work and the use of Open Data. The authors exercise a number of the dimensions of learning about GSE noted as missing in Clear et al. [1] (e.g., regulatory and intellectual property issues, student responsibility and decision making, software process, and management through self-reliance) and developing the well-rounded “independent” thinking student, which is

especially important when building software in virtual teams. Methodologically, this article provides a unique example of an article that adopts a critical stance by positioning the student experience and opportunities for real learning within the wider context of the distortions imposed by the steering mechanisms and regulatory context in which their course is situated.

Taking a qualitative approach in which a group of software engineering students were interviewed, Isomöttönen and colleagues warn of the dangers of too much regulation, in which education system-imposed and group-imposed regulations narrow student opportunities for learning and creativity. Students appear to embrace autonomy when offered, which was linked, not surprisingly, to motivation. Offering students, as a group, the opportunity to take care of their own learning was an important condition. These authors also advocate giving students complex and open-ended problems.

The article takes the reader through courses run at their institution, the University of Jyväskylä in Finland, in which they explore giving the students control over their learning using Open Data and project-based learning. Distilling the student responses, the authors present three thematic networks representing students' experiences of the education system: the first of which they term: "Search for meaningfulness in 'foundation' education." The second network presented describes students' experiences of constrained opportunities for learning, entitled "Opportunity narrowed;" and a third network centers on aspects that underpin positive "Transformations and success." Although drawing on the results of semi-structured interviews with 13 students, this article succeeds in shedding some light on the difficult and opaque question of what students gain from project-based learning in terms of learning outcomes, and how those opportunities for learning can be unwittingly narrowed. The resulting themes and networks can help other course leaders design similar novel interventions that attempt to improve and widen students' learning possibilities, and prepare them for the complexities of GSE.

The fifth article in the issue "[Exploring and Expanding GSE Education with Open Source Software Development](#)" by Deepti Mishra situates itself within the *semi-immersive* layer of our GSE Education continuum. It reports experiences from the Norwegian University of Science and Technology in adopting an Open Source Software Development (OSSD) approach as an alternative to their previous multi-site collaborative GSE course. The article provides an honest account of the challenges and differences between the two approaches, framing the comparison within the extended GSE taxonomy of Britto et al. [11]. The course aimed to teach GSE, with aspects of outsourcing (not always part of GSE courses) and OSSD. It combined practical work with OSSD projects and research seminars in which readings on key topics were critiqued and related to experiences in the course. The authors argue that OSSD enabled students to realistically experience many of the issues faced in a typical distributed GSE course, but with less demand on the instructor. The article adds to the debate on optimal approaches to teaching GSE coined by Beecham and colleagues [12]. Some key messages relate to the need to manage risks in an OSSD project to ensure that the project will expose students to a core set of GSE challenges. One proposed strategy is ensuring that a key member of the OSS project is available as a mentor to help with onboarding, so that students can readily come to grips with the important but non-trivial task of comprehending large and complex code bases [13], tool sets, and standards. Those complexities necessarily dictate an initial learning phase in an OSSD project. Further risks to be managed are lack of responsiveness from the OSSD team, so selecting a suitable project and a supportive community to work with are key considerations.

3.3 GSE Simulations and Practice Role-Play

In article six, Billingsley et al. report on a GSE-Ed related course they have been running for 6 years at their Australian university. With a primary initial aim of exercising *GSE Professional Practices*

through extending the studio model delivery to a distance version of the course, the research team explain how they emulate many GSE team practices in a local and time-restricted classroom environment. Two layers of collaboration are set up in which small teams of students work on different features of a common product. Students work intensively together on a given feature but must collaborate with other teams on the common product. This experience is further enriched as the course design requires students to use continuous integration, asynchronous communications channels, and work on complex software projects. The overhead of running such courses is kept to a minimum, as the students are based at the same university and self-organize inter- and intra-team collaborations to meet their goals. Course organizers therefore do not need to concern themselves with uneven team sizes, re-adjustment of assessment schemes, and different university structures, all of which is the case in the *fully immersive* experience we see in articles one, two, and three.

The study reported here “[Taking a Studio Course in Distributed Software Engineering from a Large Local Cohort to a Small Global Cohort](#)” takes an action research approach, where over a number of years, a group of researchers and practitioners have tried to combine theory, action, reflection, and practice to improve software engineering studio education. The authors find this approach as particularly appropriate for their course that uses studio pedagogies, which are themselves grounded in theories of reflective practice and situated learning. The authors explain how the original (co-located) course has been adapted to fit the needs of an online, geographically distributed course, delivered at a different university. This transfer allowed Billingsley and colleagues to evaluate the distributed nature of the course, and compare the behavior of those students working remotely with those who met face to face occasionally. In all cases, students were forced to work remotely, with data from development logs indicating that much of the development occurred outside of synchronous class hours.

Although, in both example cases, much of the development is conducted in the same country (and therefore with students from a similar culture and time zone), the distributed nature and extensive use of asynchronous communication channels evident from this course indicate that many GSE challenges are experienced. Transferring from a classroom-based studio course to an online course allowed the authors to identify which parts of the course needed to change to meet the needs of a geographically distributed course. As such, the authors claim that the resulting course (detailed in their study) is extremely flexible, will suit large and small classes, and will work as part of a local or a global course.

Article seven, “[Evaluating GSD-Aware: A Serious Game for Discovering Global Software Development Challenges](#)” by Vizcaino and colleagues, fits within the *GSE Simulation* layer of our continuum. The article argues for the merits of simulation as an approach to learning GSE (or Global Software Development), echoing the views of Noll and Scacchi in Beecham et al. [12]. The authors “propose the use of a serious game called GSD-Aware, with which students can ‘suffer’ some of the typical challenges of GSD by interacting with avatars and by using several means of communication to solve a number of problems posed.” The article describes the game and an evaluation of its effectiveness in helping students develop awareness of GSD challenges. Although a relatively short (1 hour) intervention, the game did appear to have a positive impact on the GSE knowledge of a cohort of 40 students. They realized that issues such as lack of coordination, lack of trust between members of a team, cultural differences between team members, lack of communication, and differences in time zones had a far greater influence than they had initially thought.

3.4 GSE Case Studies and Exercises

In the final article, “[Building LEGO Towers: An Exercise for Teaching the Challenges of Global Work](#),” Šablis and colleagues “present a small-scale exercise that uses LEGO bricks to teach skills

necessary for global work.” This article fits within the *GSE exercises* layer of our continuum, with the authors suggesting such an exercise (of 75 minutes duration) as a complement to a larger GSE course, or a partial substitute if such a course is not possible. The exercise involves the students building two identical LEGO towers, working “as a virtual team with members separated in different locations, starting with an incomplete set of instructions and an incomplete set of building blocks. The communication between remote team members is restricted to the simulated lean communication channels, such as written notes delivered by facilitators.” The authors argue that the exercise can usefully explore particularly troublesome GSE issues in depth, in a concrete and meaningful way. Unlike many of the other articles in the issue, which have adopted a macro-level view, this article outlines the results of a micro-level set of task-based experiments. After identifying a set of threshold concepts in GSE, the article provides a carefully designed and detailed analysis of varying experimental configurations of the exercise, and their outcomes. The results suggest that the exercise may provide an accessible and relatively low-effort means of teaching and evaluating selected, highly focused but important aspects of GSE knowledge.

4 FINAL WORDS ON GSE-ED RESEARCH

The eight articles included in our special issue reflect the breadth of research currently conducted on GSE-Ed course design in undergraduate programs. GSE principles are shown to be taught in a multitude of ways, and there are examples of quick and easy solutions that should motivate those thinking of embarking on this form of teaching for the first time. In addition, for those who have experience in teaching such courses, there is a raft of new ideas and guidelines to draw on. For example, those new to GSE-Ed could start with the quick to administer GSE-Ed *games and simulation* options offered in articles seven and eight. Alternatively, something a little more heavyweight might be more suitable like the studio course detailed in article six where students engaging in *GSE professional practices* develop large software systems that require cross team collaboration. Moving away from the classroom and for those wanting their students to experience building real systems and working remotely, the open source option is what we view as *semi-immersive*. Essays on *professional values*, approaches to their assessment, and how to prepare the student for the global workplace, in terms of lifelong learning and students’ disposition to learn are topics of debate in article four. Finally, for the *fully immersive* experience, in which the students experience the gamut of responsibilities and issues present in GSE, see articles one, two, and three. The effort of running the multi-site, multi-university courses across different countries and time zones are shown to reap rich rewards for both the student, the profession they enter, and the reputation of the university who runs the courses.

We hope that the ideas and guidelines presented in this special issue will provide a valuable resource for educators embarking on GSE-Ed courses and stimulate further research in this area, which is only now starting to build momentum and become identified as an educational field in its own right.

ACKNOWLEDGMENTS

We also thank the many reviewers who participated in this special issue; the final set of articles have benefitted from the constructive feedback.

REFERENCES

- [1] Tony Clear, Sarah Beecham, John Barr, Mats Daniels, Roger McDermott, Michael Oudshoorn, Airina Savickaite, and John Noll. 2015. Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review. In *ACM Proceedings of the Working Group Reports of the 2015 on Innovation & Technology in Computer Science Education Conference (ITiCSE'15)*.

- [2] Outi Sievi-Korte, Sarah Beecham, and Ita Richardson. 2019. Challenges and recommended practices for software architecting in global software development. *Information and Software Technology* 106 (2019), 234–253. <https://doi.org/10.1016/j.infsof.2018.10.008>
- [3] Miguel J. Monasor, Aurora Vizcaino, Mario Piattini, John Noll, and Sarah Beecham. 2014. Evaluation of a simulation platform for interaction training: A multi-phased methodology. In *Proceedings of the Frontiers in Education Conference (FIE'14)*. IEEE.
- [4] Mohammad Abdur Razzak, Ita Richardson, John Noll, Clodagh Nic Canna, and Sarah Beecham. 2018. Scaling agile across the global organization: An early stage industrial SAFe self-assessment. In *Proceedings of the 13th Conference on Global Software Engineering*. ACM.
- [5] Sarah Beecham, Tony Clear, John Barr, Mats Daniels, Michael Oudshoorn, and John Noll. 2017. Preparing tomorrow's software engineers for work in a global environment. *IEEE Software* 34, 1 (2017), 9–12.
- [6] Tony Clear, Gwyn Caxton, Simon Thompson, and Sally Fincher. 2011. Cooperative and work-integrated education in information technology. In *Integrated Handbook for Cooperative and Work-Integrated Education*, R. Coll and K. Zegwaard (Eds.). World Association for Cooperative Education Inc, Lowell, MA, 182–196.
- [7] Mats Daniels, Christine Faulkner, and Ian Newman. 2002. Open ended group projects, motivating students and preparing them for the “real world.” In *Proceedings 15th Conference on Software Engineering Education and Training (CSEE&T'02)*.
- [8] Tony Clear, Elin Parsjö, Åsa Cajander, Mats Daniels, Nanna Lagerqvist, and Roger McDermott. 2016. A framework for writing learning agreements. In *Proceedings of the 46th ASEE/IEEE Frontiers in Education Conference*, D. Trytten, H. Matusovich, and M. Castro (Eds.). IEEE.
- [9] Stephen Frezza et al. 2018. Modeling global competencies for computing education. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 348–349.
- [10] Tony Clear. 2017. Meeting employers expectations of DevOps roles: Can dispositions be taught? *ACM Inroads* 8, 2 (2017), 19–21.
- [11] Ricardo Britto, Claes Wohlin, and Emilia Mendes. 2016. An extended global software engineering taxonomy. *Journal of Software Engineering Research and Development* 4, 1 (2016), 3.
- [12] Sarah Beecham, Tony Clear, Daniela Damian, John Barr, John Noll, and Walt Scacchi. 2017. How best to teach global software engineering? Educators are divided. *IEEE Software* 34, 1 (2017), 16–19.
- [13] Tony Clear. 2005. Comprehending large code bases—The skills required for working in a “brown fields” environment. *SIGCSE Bulletin* 37, 2 (2005), 12–14.