How Should Software Reliability Engineering (SRE) Be Taught?

Mario Garzia
Microsoft
mariogar@microsoft.com

John Hudepohl
Will Snipes
Nortel
Hudepohl@nortel.com, wbsnipes@ieee.org

Michael Lyu
Chinese University of Hong Kong
lyu@cse.cuhk.edu.hk

John Musa
Independent Consultant
j.musa@ieee.org

Carol Smidts
University of Maryland
csmidts@glue.umd.edu

Laurie Williams
North Carolina State University
williams@csc.ncsu.edu

Abstract

This article on teaching software reliability engineering (SRE) represents a consensus of views of experienced software reliability engineering leaders from diverse backgrounds but with ties to education: directors of software reliability and software reliability training in industry, a consultant who teaches SRE practice to industry, and university professors. The first topic covered is how to attract participants to SRE courses. We then analyze the job-related educational needs of current and future (those now university students) software practitioners, SRE practitioners, researchers, and nonsoftware professionals. Special needs relating to backgrounds, limited proficiency in the course language, and work conflicts are outlined. We discuss how the needs presented should influence course content and structure, teaching methods, and teaching materials. Finally, we cover our experiences with distance learning and its special needs. Some of this article applies to any course and is not SRE-specific.

1. Attracting participants

Unfortunately, there has been some difficulty in attracting participants to SRE courses in both universities and in practitioner environments. A cursory review of the general press and the technical literature indicates that potential participants are generally aware of the impact of software failures on our society. For example, there has been a continuing series of articles on this topic in ACM's *Software Engineering Notes*. However, it appears that there is much less awareness of the existence of SRE as one of the solutions to this problem. This would indicate a need for overview feature articles in the press and in general computing magazines. The latter should include not just those of the professional societies, but also those with a commercial flavor. These articles need to convey the true importance and value of SRE, and need to present case studies that emphasize its benefits. They need to catch the eye of readers, by showing how this practice can benefit them and their company.

"In addition to a lack of awareness, there appears to be a perception (not necessarily backed by facts) among some university students that SRE courses are not very interesting and are not "cool." This is not universal; SRE is among the most popular courses at some universities, which indicates a need to find out why this difference exists. It is not totally clear what makes a new practice attractive from a fashion viewpoint, but maybe we should not just dismiss this factor. Perhaps we should look at what areas are attractive (for example, agile methods) and why. One possible problem is that SRE is more intangible. There seems to be a significant difference in level of "coolness" between the relatively concrete and hence definitely learnable methods that are used to create code and the vaguer and hence difficult to learn methods that are being used to assess code. Perhaps we need to do a better job of highlighting the "cool" problems that SRE can solve and their importance to industry. And we need to make sure we are approaching the entire pool of candidates for these courses: not just the computer science area, but also statistics, reliability, and other engineering disciplines.

We should emphasize course activities that are appealing to participants: working cooperatively as teams in workshops, practically applying what they have learned, and chances to ask questions and present their own experiences and ideas. Cooperative work builds important social and communication skills that are not traditional engineering subjects, but are increasingly vital to successful careers. Also, in this age of globalization and international competition, many participants are concerned about their careers: they need to see how SRE is very business-oriented and thus closely related to increasing their competitivity.

2. Job-related needs of course participants

We can group the needs of course participants in four broad categories: SRE practitioners, software practitioners, nonsoftware professionals, and researchers. The needs are largely the same for people currently working and students in training. SRE practitioners must understand SRE practice in detail so that they can direct and guide its application as resident experts. Software practitioners need to understand enough about SRE to apply it intelligently in developing and testing software. What a software practitioner needs to know may vary from organization to organization, depending on how software development work is divided in the organization. Nonsoftware professionals need to understand enough about SRE to understand how it affects their work, and even to apply it to a limited extent. Researchers must understand not only the practice but also the theory on which it is based so that they can advance the field.

Just to give a very rough picture, the education of software practitioners and nonsoftware professionals requires perhaps two days and the education of SRE practitioners requires perhaps several weeks. The education of SRE researchers usually requires many months, plus years of active supervised research. These numbers include both formal coursework and practice. Thus, it is likely that not all needs can be met in one course, although it would be highly desirable for as many as possible to share a common basic SRE course. Since the number of SRE practitioners will be considerably smaller than the numbers of software practitioners and nonsoftware professionals, it may be practical to educate this smaller category primarily through reading and guided experience, once they have taken a basic course. Although researchers can start with a common basic SRE course, it will probably be desirable to develop a theory-based course for them if their numbers warrant (such a course is beyond the scope of this article).

An SRE course must have content that is applicable to large software development projects, large and varied customer bases, and frequent revisions and updates. Many commercial software organizations distribute SRE functions among different software practitioner roles in the organization. Efficiency requires that each software practitioner focus on the practice assigned to that role. However, software practitioners also need to understand the overall landscape of the application of SRE to software development to be effective. The instructor must communicate the big picture of software reliability in simple terms and relate it to the software practitioner's sphere of influence over reliability outcomes.

Accuracy in setting reliability objectives and in measuring reliability achieved is required to assure customer satisfaction, as reliability can be a key software differentiator. Accuracy depends on several important factors: addressing the variety of customer operational profiles and reliability needs, assessing reliability prior to release with limited run times, and evaluating reliability against competing products. Efficiency in development is required to assure that the work can be done within available time, resources, and budget. We need to predict expected reliability early in development when code is not yet available and measure reliability achieved as the code is developed and tested.

Developing operational profiles is greatly complicated for mass-market products where the number of different types of customers can be very large. Flexible methods for developing the right set of operational profiles in this environment are critical to meeting customer reliability expectations.

A fast-changing development environment is another factor that one must take into account. Assessing software reliability prior to release is more complicated when there are frequent deployments. For example, web-based software deployments can be just weeks apart. In large and complex development projects, the multiplicity of components and dependencies means that the software is in almost constant flux. To address these situations, SRE tools and techniques are needed to estimate the failure intensity of the software, even though there is only time to experience just a small sample of failures, due to the short release cycles or constant changes and additions before deployment.

Once reliability is established, one must assess the results against the product's competitors. This provides information on areas for improvement, allows product customers to evaluate their options, and provides results for product marketing. SRE techniques for developing and assessing software against reliability benchmarks, as is done for performance, can be very valuable and serve as a standard.

The SRE practitioner designs the SRE process a project will use. The process must work within the structure of the software development and release process. In some cases, the process is related to the Capability Maturity Model Integration (CMMI) or one of its derivatives. SRE should influence production, release, and process improvement decisions. The SRE practitioner must first understand the roles of all software practitioners in the organization and what it is important for them to contribute to SRE. Thus, the SRE practitioner requires a thorough knowledge of basic SRE and practice in its application to projects.

By understanding the project organization's software development process, the SRE practitioner can better define a set of SRE program practices that influence reliability at key leverage points. For instance, in the software design phase of the life cycle, the SRE practitioner could define a practice for software Failure Modes and Effects Analysis (FMEA). The software verification phase could incorporate a practice for enhanced code inspections that both removes defects and provides data for reliability estimates. Automating the practices with suitable tools provides a structure to the process, a means to incorporate it into existing software practitioner responsibilities, and a more consistent source of reliability data. It is important that initial application of practices by the software practitioners be carefully monitored by the SRE practitioner to tune them to the needs of the product, and to make sure that the software practitioners have learned and are applying them properly. Hence, the education of the SRE practitioner must include not only the basic SRE course but also general knowledge of software development and testing practices plus practical experience.

University professors generally feel that professional software practitioners should understand the principles behind software reliability and testing. They feel that such knowledge will aid practitioners in building solid, testable code, in validating and verifying this code, and in engineering reliability into their projects. Industrial organizations want software practitioners to acquire skill in applying SRE inexpensively, and may differ from universities in sacrificing background to attain immediate proficiency in a limited area.

Practitioners feel that there is very little testing training in universities, with the result that all graduates want to be developers and very few, if any, want to be testers. This may be due to a preponderance of general courses in universities that emphasize design and programming techniques in various applications (e.g., OO, Java, CORBA, web services, etc.). Testing and SRE courses do exist, but there are far too few of them. It is most important that the emphasis change.

Researchers must acquire a thorough grounding in the theoretical underpinnings of SRE. They need to understand the diversity of existing approaches, when to apply which approach or model, and the likelihood of success of a particular approach. They must know the important open research areas and questions. They also need hands-on experience through a real-life project to acquire judgment in evaluating the importance of new questions and new results.

3. Special needs

Course participants may have either pure academic background or may be more practically oriented. Some of the participants may have no prior software engineering knowledge and require a short prerequisite course to address this issue.

In both university and industrial environments, some audiences have participants of limited proficiency in the course language. The extent of this problem is generally greater for public (open to all individuals) courses, courses at international conferences, and courses in nonanglophone countries (since English is the most common course language).

In industry, the most important environmental constraint for a basic SRE course is the high-pressure demands of work in a competitive environment, making semester length courses impractical. In fact, two intensive days is the practical limit. Registrations fall off drastically for longer courses, so one must organize the material to fit within this limit. In a university environment, learning can be spaced over time, typically a semester, and participants have the motivation of working for a degree.

Conflicts with work, urgent situations, meetings, etc. mean that, even with a two-day class, a large percentage of the participants (often 20% to 30%) will miss some class hours (often 10% to 25%). Teaching methods must adapt to this situation.

Industrial organizations expect participants to bring back and implement new ideas at work; in classes for industrial professionals, you need to increase the likelihood that this will happen. As we mention later, one way to maximize this and increase attendance and attention is for participants to work on specific problems from their projects as part of the class. Another is to offer ready-to-use tools and procedures.

4. Course content and structure

Course feedback strongly indicates that practitioners prefer to learn an organized, tested process rather than a collection of techniques and tools. References to SRE users, especially those who have written up their experiences, is important. The two-day course constraint noted in the previous section requires that you present a process that works for, say, 80% of but not all projects. Separate the material for special situations, and consider letting students access it on their own.

The basic course should cover the six principal activities of SRE: defining the product, implementing the operational profiles, engineering the “just right” reliability, preparing for test, executing test, and guiding test. Guiding test includes determining when to deliver software, and it presents the methods used for evaluating field reliability, so that feedback can be provided to the next release cycle. Defining the course based on individual activities provides a focus for specific roles, indicating the relative importance of different activities for those roles. An overview of all the activities should be given separately for management personnel. Deemphasize theory, as most practitioners have little interest in it, except to know of its existence.

It is important that the structure of the course be highly interactive, so that the course can accommodate to the different backgrounds and prospective roles of the participants. This approach also makes the course more attractive to prospective participants. Another requirement for the structure is the use of workshops. These are also attractive to potential participants. They also teach implicitly the social and communication skills that are indispensable in today’s development environment.

Experience with SRE courses in software development organizations has shown that scheduling attendance by product development groups is much more effective than by individuals. The workshops can then focus on the particular products and the techniques that are most relevant to them.

To help convince an organization to apply SRE in the workplace, the course should provide information on how to accomplish technology transfer. Conduct a final workshop that addresses how the participants will apply what they have learned on the job, setting up action items, persons responsible, and target completion dates. References to users of SRE who have published their experiences are helpful.

A course that is directed to future researchers will require background in statistics. However, this will in general not be necessary for future software or SRE practitioners.

Let’s consider two specific examples of course content, a course at North Carolina State University (NCSU) and an in-house only course at Nortel.

Since three of the six principal SRE activities are intimately connected with testing, NCSU combines SRE with a testing course. Course participants apply the testing and reliability theory learned in the class in several ways. First, the course takes place in the NCSU Laboratory for Collaborative System Development. Therefore, the twice-weekly classes are intermingled between lecture-style instruction and hands-on exercises on the computers. For example, instruction is provided on unit testing, coverage principles, and the JUnit testing framework. Then, a short exercise for developing JUnit with high coverage is completed.

Additionally, the students apply their software reliability and testing knowledge via a semester-long project. In five iterations, students develop the project. Code is synchronously developed with automated and “strategized” black and white box test cases and through manual test plans. The programs are written such that the operations are logged, enabling a retrospective analysis of the actual operational profile. In the final phase of the project, students choose a subset of their manual tests that they can run in one 70-minute class period, based upon an estimated operational profile. During two class periods, student teams swap completed projects and run their 70-minute test on other teams’ projects. Test failures are documented to provide feedback to student teams on defects that have escaped all testing.

Nortel breaks down SRE activities into reliability practices for software practitioners, using the guideline that each reliability practice should be trainable in two hours of instruction and hands-on training. Training for each practice is tailored to the role of the software practitioner and focuses on the information needed to perform the practice and how it contributes to the overall reliability of the software. A business case for each practice can show how it contributes to the product reliability and improves development and maintenance costs. With the brief period for training, some practices will require a mentoring period as the software practitioners begin to perform them in everyday work. A support system provides mentoring as needed to the trained software practitioners in the organization. The data collection aspects of each activity provide the SRE practitioner with performance results and data for updating reliability predictions. The practices support each other through their modeled influences on reliability and business outcomes.

The Nortel approach requires centralized planning for the application of SRE, and probably for other practices as well. This approach has many advantages, but the majority of companies take a decentralized approach of letting each project apply SRE as it sees fit, although they do base their application on the six principal activities of SRE described above.

5. Teaching methods

When a class is given for industry, do so locally but off-site if possible to discourage interruptions. Try to minimize class time missed due to work conflicts by consulting with the class at the start about any necessary adjustments in start, stop, lunch, and break times. However, since some missed time is unavoidable and since some participants may have difficulty with the course language, alternate means of receiving the course material are essential. It is important that all lecture material be based on a detailed set of slides. The level of detail must be such that someone who has missed class time can review the slides at any point and catch up to the class. A book that is carefully correlated with the course is essential here. The instructor should encourage participants to interrupt the class at any time to clarify issues of general interest. Suggest deferring individual issues (for example, from those who missed part of class) for one-on-one discussions at break periods.

Workshops have proved to be very effective in reinforcing what has been learned. Although it is possible to create canned workshops, participants have shown a strong preference for applying SRE to their own projects. These have the most meaning for them. This applies to university settings as well; students usually have some project they are working on, even if it is in another course.

If most participants have limited proficiency in the course language, slow the course's pace and avoid humor and slang. Allow workshops in the participants' native language, but have all decisions written in your language on flip charts so you can guide the workshops.

In analyzing teaching experience, some lessons learned are SRE-specific. Participants usually find the engineering just right reliability and preparing for test activities the most difficult to learn. Teaching how to use the software reliability estimation program CASRE in a computer classroom is usually not worth the time, since there is a good user manual in the book Musa, *Software Reliability Engineering – Second Edition*, Author House, 2004. CASRE is easily learned from the manual, and the time can be put to better use. In application, the importance of carefully defining product, customers, and users is usually grossly underestimated. In developing operational profiles, the difficulty of determining occurrence rates for operations is usually overestimated.

6. Teaching materials

As noted in the previous section, detailed slides are essential, with copies provided to all participants. Reinforce the material presented by applying it in workshops related to the participants' projects.

A book that is closely coordinated with the course serves many purposes. Course participants can use it for review and reinforcement of what they have learned. It is very important in an industrial setting, where participants often miss parts of the course. The book is often needed during workshops as participants practice applying what they have learned. And it is a valuable familiar reference when they apply SRE on the job. A book is also very helpful to practitioners whose native language is not that of the course.

An appropriate book can include material that SRE practitioners need beyond the basic course. For example, it can be used to provide for treatment of special situations, since there is not time to cover them in the basic course proper. It can include FAQs collected from former course participants; they resolve the difficulties participants most commonly encounter. It can also provide theory for those who may be interested in it, but theory should be well separated from the core material.

The software reliability portion of the course taught at NCSU uses Musa, *Software Reliability Engineering - Second Edition*, Author House, 2004 (ISBN 1-4184-9387-2). Lecture slides coordinated with this book and other resources are posted on the OpenSeminar in Software Engineering (http://openseminar.org/se/courses/41/modules/206/index/screen.do).

The book is also used in many other universities worldwide as well as in a widely taught two-day intensive course by Musa in industry, both on site and by distance learning. A semester length course at the University of Maryland uses Musa, John D., et al, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 1990. (ISBN 007044093-X) and Lyu, Michael R., *The Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996 (ISBN 0-07-039400-8). At the Chinese University of Hong Kong the SRE component of an undergraduate level software engineering course and a graduate level SRE course use *The Handbook of Software Reliability Engineering*.

There is a resource center for SRE available at http://members.aol.com/JohnDMusa/. It contains or links to a great deal of material that is of interest to professors, practitioners, and researchers.

7. Distance learning

Some course participants cannot attend standard classes and require distance learning. Distance learning can be either scheduled in a virtual classroom setting or self-paced. The virtual classroom setting, often web-based, has the advantage of real time interaction with the instructor and the class. Complex questions can be better handled in such a situation and group workshops are feasible, although the latter do not work as well as in a live setting. Virtual classes may provide discipline for participants who need it.

On the other hand, self-paced courses are very advantageous for most industrial professionals since they can take them anywhere at any time; for example, during travel. You never miss a class or part of a class due to conflicting work demands.

Both require development of detailed slides, a means for participants to ask questions, and a means for participants to apply

SRE and receive critique and feedback from instructors. A book that is closely coordinated with the course is particularly important for distance learning participants. Although questions, critique, and feedback by telephone are possible, experience indicates that email is usually superior. This eliminates the problem of scheduling a time when both instructor and participant are available. It gives each party time to reflect in formulating focused, unambiguous, precise questions and answers.

Distance learning instructors need to give attention to the danger of participants becoming isolated, especially if there are classroom participants taking the same course.

8. Summary

The extensive experience of the authors with the teaching of SRE indicates that a course should be practically oriented and highly interactive. Participants require chances to ask questions and present their own experiences and ideas. Lecture material needs to be applied in workshops, with participants working cooperatively in teams. If the teams are product development groups and they are applying SRE to their own products (even if simplified), the course will be particularly effective. Workshops make it possible for the course to be sufficiently flexible to meet widely differing participant needs. Course interruptions are common, and teaching methods must adapt to this situation. As a result, requiring a good textbook that is carefully correlated with the course is absolutely essential.