
Empirical Software Engineering, 3, 7–8 (1998)

Editorial

Sharing the Wealth: Accumulating and Sharing Lessons Learned in Empirical Software Engineering Research

I have recently had the pleasure of working with a couple of new colleagues who are just now starting their careers. While I have personally gained a great deal from these interactions, I have only now started to appreciate the little nuggets of wisdom that I have somehow accumulated over the years. Little generic things like how to aportion credit among several authors, what makes a good and useful review; how to write a good call for papers; etc. But of course, there are also nuggets more specifically oriented towards empirical software engineering research. For example, how to motivate students to participate in an experiment; ways to define and collect data so that if the initial hypothesis doesn't hold up, you still have data you can do something with; and the pitfalls (so you can plan for them) of doing a controlled study.

Now these aren't the kinds of things you can write papers about (or at least papers that get published). In many cases they aren't significant enough, or general enough or original enough to make it through a rigorous refereeing process. But at the same time, these are the sorts of issues that can make or break an experiment. After reflecting on this, I came to the realization that there are dozens if not hundreds of other researchers who have worked in this field as long as I have if not longer. Together, we probably have centuries of collected wisdom and experiences.

It just so happened that as I was reflecting on these issues, I was also finishing up a project to design and implement a Web-based Lessons Learned Repository (WLLR) for a medium-sized hardware/software company into which their project managers and software engineers could deposit (and retrieve!) nuggets of wisdom acquired in the process of developing their products.

The goals behind the software engineering WLLR was to make sure that any given mistake (and there are always plenty of them) would happen only once, and no one would spend time re-inventing solutions to common problems. The WLLR was intended to augment the company's current "project post-mortem process" which was carried out religiously by every project, but whose final reports were filed away, never to be seen again (both I and the manager funding the project were surprised when we started to turn up post mortem reports from projects completed in 1982). These two goals implied it had to be convenient for engineers who discovered things that worked (or didn't work) to add them to the repository. But it also meant that when an engineer had an idea or a problem it would be easy to retrieve applicable lessons for the repository without lots of unrelated items clouding the issue.

Apart from technical issues like HTML input forms and cgi-bin scripts, we also had a number of design issues to deal with. We had to develop a set of classifications dealing with phase of the lifecycle, specific software engineering artifact affected and potential consumer

of this information. Each of these classifications had to be reviewed and discussed with numerous program managers and software engineers as we tried to resolve differences in terminology, process and methodology. We endeavored to ensure that we would capture enough information to retrieve a lesson only when appropriate, while at the same time making sure the originator of the information wouldn't end up spending half a day choosing among categories. And of course there are always policy issues. Who owned the repository (meaning whose budget had to pay for maintaining it)? Could certain lessons be filtered out? How long should a given lesson stay active? Could good ideas from magazines and journals be added, or did a lesson have to be experienced first?

It certainly doesn't take a rocket scientist to move from a WLLR for lessons learned about the process of developing software to a WLLR for lessons learned about studying the process of developing software. A web based Lessons Learned Repository for Empirical Software Engineering Research which would allow researchers the world over to contribute nuggets of wisdom might very well accelerate the state of both research and practice. It could help ensure that researchers don't make the same mistakes that others have made. When a solution to a common problem is identified, others could benefit from it and spend the time they would otherwise spend on addressing it to actually doing research. In short, the result might be more productive research efforts and less aborted projects.

Don't get me wrong. This wouldn't take the place of peer reviewed journals. The sorts of nuggets of wisdom I have in mind would be blatantly inappropriate for publication. No referee in their right mind would bless a paper whose only contribution was to point out that sometimes students get up and leave in the middle of an experiment to relieve themselves playing heck with your carefully constructed controls. Likewise, the kind of presentation I have in mind would similarly be inappropriate for publication—no one wants to read 5 pages of text to learn a two line lesson: "subjects don't respond well to threats of physical violence," for instance. On the other hand, it might be useful for researchers to be reminded of the limits of human endurance so they aren't nonplused when halfway through their four hour experiment, two-thirds of the subjects revolt, laying waste to months of work.

My goal in writing this editorial is to drum up a little interest in this idea. From experience (hey—here's our first lesson!), I know that this kind of an effort can take a long time to be implemented and even longer to get people contributing their wisdom. I also know that if you ask a dozen people what they want out of something like this, you'll get a dozen answers. In order to get the community firmly on board a Lessons Learned Repository for Empirical Software Engineering Research, it will require a fair amount of participation and input (here's our second lesson—don't go out and fire up one of your graduate students to do this as an M.S. project, put it on the web and expect people to start contributing—they won't!). So if you're interested, send me mail (warren@cs.pdx.edu). If I get enough interest, I'll create a mailing list. If I get lots more, we can put together a newsgroup. If I get swamped with e-mail I'll organize a workshop.

That's enough for this issue. See you on the web.

Warren Harrison