Reflection

Improving research through knowledge transfer

Andre Oboler & Simon Lock
Computing Department
Lancaster University
Lancaster, UK
oboler@comp.lancs.ac.uk, lock@comp.lancs.ac.uk

Ian Sommerville
School of Computer Science
St Andrews University
St Andrews, UK
ifs@dcs.st-and.ac.uk

***Abstract*— It is through our mental models of the world that we understand it. Advances in science are nothing more than improvements to the model. This paper presents the development and refinement of our model of the research process as we seek to understand and improvement the process through three generations of case studies. We conclude by introducing an approach to help manage and plan research projects.**

Process imrpovement, knowledge transfer, research environment, process modeling, software engineering education

I. INTRODUCTION

This paper presents the results of three years of case studies focusing on researchers in a RAISER [1] setting and examining ways of improving productivity for now, whilst also capturing more knowledge to improve productivity in the future. Past work examined the use of Software Engineering by Computer Science Researchers [2] and created the RAISER/RESET Software Development Life Cycle [1] to meet the needs of the research environment. The RAISER/RESET approach split the long-term work into Research (carried out by researchers under a RAISER methodology) and Development (to be carried out by professionals engineers attached to an academic institution under RESET guidelines). The RAISER phase aims to increase productive for the research as well as increasing the amount of knowledge generated as a result of the work. The Development phase organises this data and reengineer software code to make it easier for other researchers to extend the existing work.

In this work we develop an approach that meets RAISER guidelines and experimentally tests it. Our hypothesis is that this approach can facilitate an improved research process i.e. allow researchers to work in a more systematic way, avoid potential pit falls, and improve knowledge transfer between themselves and others, and between their development role and reporting of output. All this requires that the approach has a low enough burden to encourage adoption. The null hypothesis is that the default unplanned approach is equally good and the only approach researchers find acceptable i.e. researchers see no benefit in the approach and it is either not adopted at all or found to be a burden with higher cost than value.

Our work uses MSc students at Lancaster University who undertook research projects between 2004 and 2006. As very early stage researchers, MSc students were seen as more likely to try new approaches. As students with hard deadlines and project that only last about 5 months they were also seen as being very discriminating when it came to their own cost / benefit analysis of potential tools. Successful adoption of a tool is itself a validation of a tool having greater benefit than cost. Our case studies also involved surveys, interviews, observation and analysis of students' final products. After each year analysis was conducted and our model and methodology adjusted to provide an improved experience for the following years set of students and greater over all clarity about the RAISER enabled research process. Other factors such as literature and discussion with colleagues also influenced the development of the model between cycles. Our latest model and tool to facilitate its implementation (both presented here) may allow others to improve the way they assist research students, or indeed allow expert researchers to document and further improve their own approach.

We begin this paper by outlining the experimental design employed. Background on our case study based experiment is provided. We discuss the development of our model of the improved research process and how the results of our experiment have provided the rational behind a focus on knowledge transfer and critical reflection as a key part of the improvement process. The process model, a key tool to facilitate the improved model is introduced. We conclude with a brief discussion on the cost and benefit of our approach and its viability for real world adoption.

II. EXPERIMENT DESIGN

Case study design requires that goals be established before data is collected. It has been stated there is a lack of clarity on the underlying principles in the software development process, on the effect of various methodologies on the process, and on what constitutes a better product [3]. Software development forms a key part of the work the researchers we focus on undertake. By adopting a definition of research it becomes possible to focus on the needs of a software development process and how interventions on the process can be usefully assessed. The OECD definition of research is "creative work undertaken on a systematic basis in order to increase the stock of knowledge" [4]. The underlying principle is to allow researchers the freedom to creatively explore while still ensuring there is a systematic basis. A better product is a greater increase in knowledge. Our goal is to improve the generation of knowledge (by transferring more mental

0-7695-2703-5/06/$20.00 (c) IEEE

information into knowledge in comments and the thesis) and promote and support both the principle of creative freedom and of systematic approach.

We chose to use a multi case holistic study [5] as a platform for our experimentation. Based on the qualitative software engineering framework in the landmark paper by Basili, Selby and Hutchens [6] we designed our study in the blocked subject-project form. Our work is carried out in vivo and makes use of experts (including the MSc Students' supervisors). Our case study approach was designed according to Guidelines outlined in Kitchenham, L. Pickard and S. L. Pfleeger [7] who along with Basili [8] have classified our form of approach as a type of formal experimentation.

Our hypothesis is that the RAISER approach can facilitate an improved research process, this necessitates the effort expended being seen as beneficial to the researcher (rather than third parties), and that the researcher will encounter less problems or mitigate them. Our null hypothesis is an unplanned approach would be equally good or better. Observation of tool adoption, surveys (via e-mail to all student or participants) and interviews on initial and final impressions of the approach and tools, and a case study examination (using interviews, technical reviews, surveys and e-mail communication) of each project, the student's experience, and how they cope with or avoid problems form a basis for our analysis. We also examine students overall course work marks and project marks. While the number of students involved and variation due to other factors make highly statistically significant results unlikely, we believe we have enough participants to at least get an indication objectively supporting or refuting the qualitative results.

Glass [9] suggests the solution to improve software engineering is greater appreciation for "ad hoc" approaches. In computing, "ad hoc" is defined as "contrived purely for the purpose in hand rather than planned carefully in advance" [10]. The lack of planning is specific to the definition in a computing context. The Latin root of ad hoc means "to this", an approach can be planned "to this" specific development process without being made up on the fly. Our null hypothesis refers to unplanned approaches, i.e. ad hoc in the stricter computer science sense.

From our perspective the experiment takes place in an evolutionary paradigm. Basili [8] used "the study of improvements to methods being used in the development of software" as an example of an evolutionary paradigm. Basili also mentions revolutionary change citing as an example "the proposal of a new method or tool to perform software development in a new way" [8]. From the view point of the students both the experiment and the nature of doing research is a revolutionary development. To provide students with stability changes will take place between cycles rather than during the period when MSc students are active.

III. The Experiment

Over the course of three years we offered each cohort of MSc students the opportunity to take part in a trial designed to assist them with their MSc project. At the end of each year the assistance offered was to be reviewed and updated. This meant about 16 students took part in each version of the experiment. As the model of research was improved in an evolutionary manner some artefacts available from the first year unchanged have now been evaluated and in some cases used by a significant number of students. Other artefacts were developed as a result of repeated evolutionary improvement.

The trial involved giving students access to a restricted website containing tools, guidelines and other artefacts. Meetings and reviews were also offered. The opportunity to participate was present in a lecture which explained the RAISER/RESET approach and the goal of our research. In the first year students who signed up were randomly split into a control group (with website access but nothing else) and an experimental group (who were asked to also have meetings and given additional reminders about tools and when to use them). The first major evolutionary change to the experiment was the decision not to have a control group in following years. This was due to an observation that about half the student self select to participate and there is no significant difference between the control group and non participants. This suggested that tool availability is not enough and guidance and communication played a large part in enabling self-reflection. It is the self-reflection that enables researchers (like teachers) to improve.

We gathered results through initial surveys and exit surveys collected from as many MSc students as possible (participants and non participants). For participants we also conducted interviews combined with introductions to artefacts on the website. In years one and three there were also joint meetings with students and supervisors. All interviews were recorded. Students' final reports and code were collected and examined. After doing this for each year level we revised our model.

IV. Model Development

In the first year process descriptors (e.g. tools, guides, recommendations) were created and placed on the website. Students chose which tools they used. Examples of tools were dOxygen (an open-source software package), a getting started document (a guide) and a research journal (a template). This is presented in Figure 1.

In the second year this evolved with process descriptors being introduced not as plug-able components, but as more abstract models for students to consider and adapt to their own style and needs. The move from tools to a process focus and the use of process descriptors is based on a very influential paper by Osterweil [11] describing how "software process are software too". He explains that "a process is a vehicle for doing a job, [while] a process description is a specification of how the job is to be done". He states that these descriptors can be at a low level (e.g. our tools) or a high level. Taking the high level approach we arrived at the idea of a processor descriptor template which models the entire research process and provides options at each stage. Changes for the second year focused on helper descriptors, such as the creation of an installation and settings guide for dOxygen, a technical review preparation checklist and sample input and output to support the coding guidelines and allow students to quickly assess its cost and value. The second year model is shown in Figure 2.

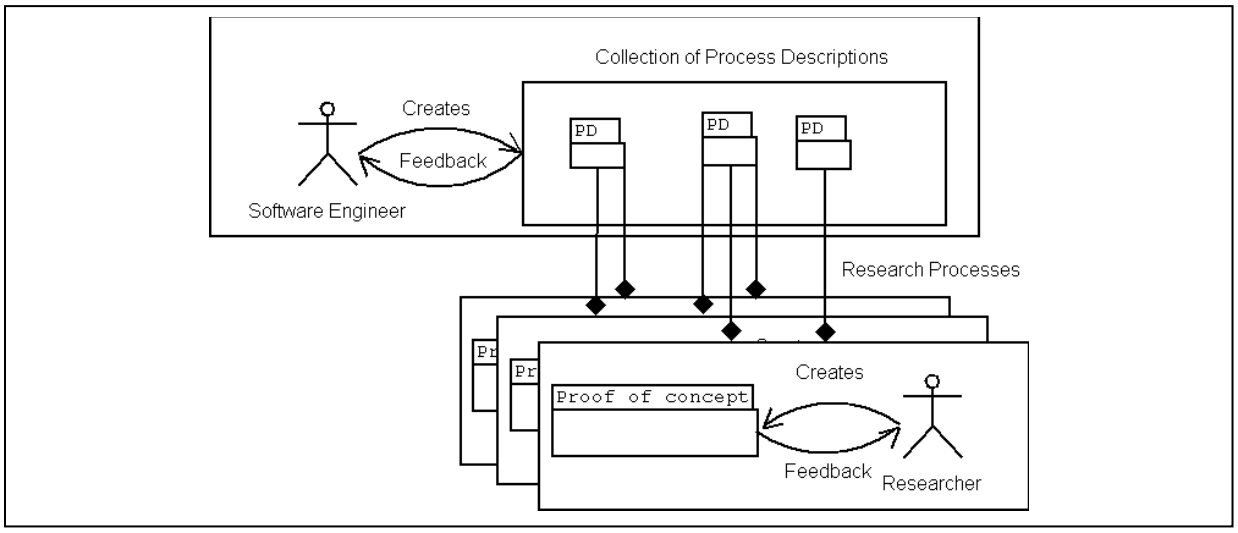

Figure 1. First Year model

After year two it became clear improving the research process had much in common with improving student learning. Teachers can facilitate student learning but they cannot learn for students. In Figure 3 the model of research includes a critically reflective role for researchers. During this reflection the researcher chooses existing process descriptors, but then works in an evolutionary way to review and change their own process and process descriptors. They also develop and adapt their process through communication with peers. In the third year we have introduced a tool to enable communication, self-reflection and planning in a lightweight and agile manner.

This development of the model allows us to adjust the underlying research process so that the role of process improvement becomes one of facilitating enhancement and an individual (but systematic) approach. This is a significant step forward particularly in the research setting. Glass (2002) claims "computer science academics looking for… the one true approach to build software systems" he juxtaposes this with software practitioners constant complaint that their project is different. Our approach now bridges this void. It enables systematic improvement, yet still ensures and in fact enhances creativity. Our key tools for facilitating this change are the process modelling tool described in the next section and the model present in Figure 3 which has been presented to students taking part in the third cycle.

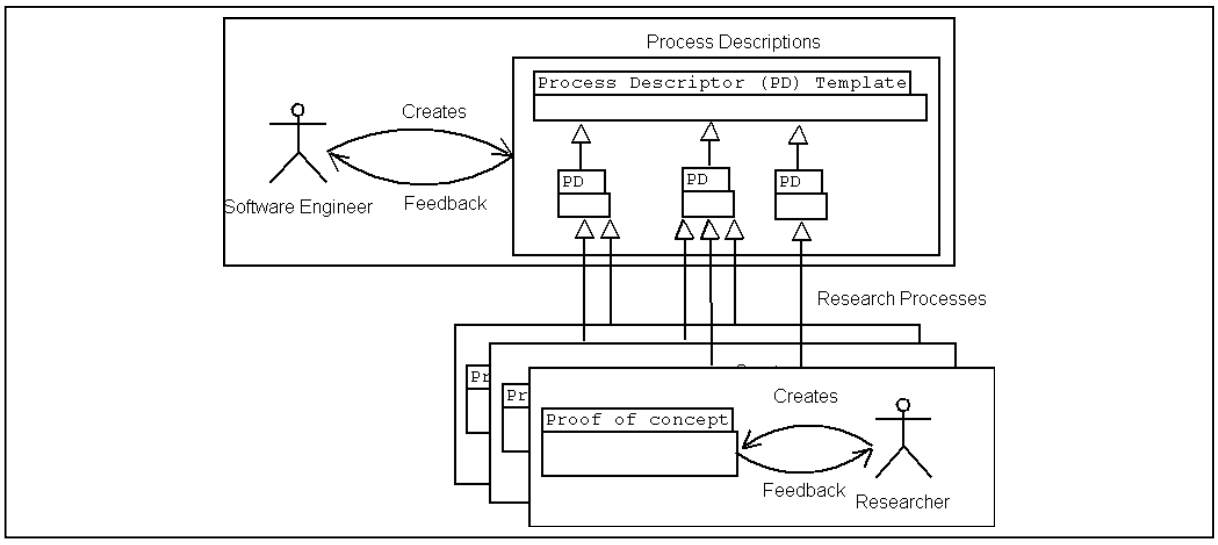

Figure 2. Second Year model

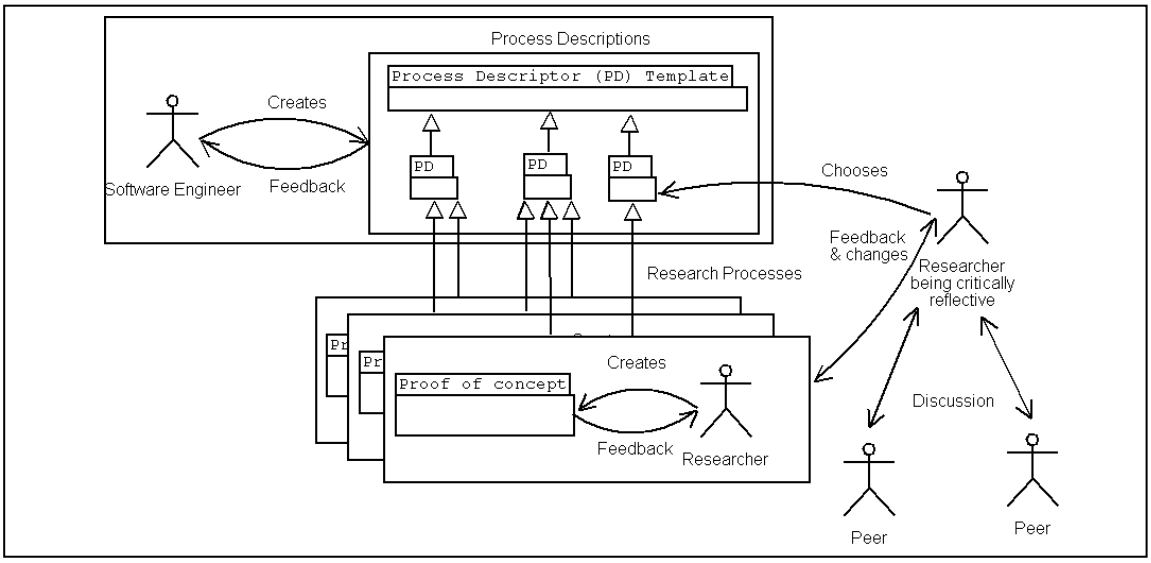

Figure 3. Third Year model

This development of the model allows us to adjust the underlying research process so that the role of process improvement becomes one of facilitating enhancement and an individual (but systematic) approach. This is a significant step forward particularly in the research setting. Glass [9] claims "computer science academics looking for… the one true approach to build software systems" he juxtaposes this with software practitioners constant complaint that their project is different. Our approach now bridges this void. It enables systematic improvement, yet still ensures and in fact enhances creativity. Our key tools for facilitating this change are the process modelling tool described in the next section and the model present in Figure 3 which has been presented to students taking part in the third cycle.

V. RESULTS

In addition to the development and improvement of the model we can present the following results on adoption, expected and perceived burden, user impressions, case study lessons, and finally statistical impact on results.

In each year about half the MSc students opted to participate. Students consistently felt the approach encouraged them to do more software engineering than they would otherwise do (an exception was a student with industry experience who found the approach helped him focus and reduce effort), yet feedback consistently indicted that participation took the same or less time than on software engineering than expected. Despite this limited effort expended, all participants in the second year felt they had benefited. In feedback one student noted how "it helps, as the project grows, to keep a clear vision of it" another said that "the tools you would be well advised to use are assembled for you. Advice on coding, backup, versioning issues etc are given without first having to ask the question". The direct link with knowledge transfer from project to report was picked up by a few students, one said they were helped by "code comments and the diary as a rough version of what I wrote in the final report". Another noted how it "helped [with] organizing the work". In the first year there was a degree of disappointment by those selected for the control group who felt they could have benefited more if they had rather been in the experimental group. The experimental group likewise felt they could have got more out of it if they'd put in more effort. These comments are one of the causes for the additional process deceptions in the second year that aimed to lower the introduction burden and allow students to more easily assess potential benefits. The students found this helpful and this was indicated in the survey where the sample dOxygen input and output was ranked the 6th most useful tool, compared to dOxygen itself which ranked 8th (out of 16 tools). The fact that dOxygen was used by the majority of the students and commented on very positively in most students feedback shows the value not only of the tool, but also of the process descriptor allowing fast evaluation. While the installation and basic setting guide to dOxygen was used by many students (in the second and third year) it ranked a poor 10th.

The difference between student course work grades (completed before they started their project) and their research project grades are shown in Table I, this is divded into participants and non participants for each of the first two years. In the first year participants on average improved marginally on their course work marks. In the second year the improvement by participants over coursework grades was higher at 5.33%. Both are variance of below 1 standard deviation from the participants mean, but the second years results are getting more noticeable. The third year's results are not yet available as the work is in progress, but with the methods having been improved further and feedback so far being even more positive than in the past it's hoped this will be reflected in the marks.

Short sessions for PhD students and established researchers that introduce the tools, followed by interviews, have indicated potential acceptance and an expectation of the usefulness of the approach beyond MSc setting. Subjects evaluated the tools purely on their merit and did not have

access to or knoweldge of the results or feedback from MSc students.

TABLE I. ANALYSIS OF GRADES

	Coursework Mean	Project Mean	Project improvement
Participant Year 1	62.73 (SD 6.46)	63.98 (SD 5.87)	1.25
Non Participant Year 1	62.41 (SD 4.75)	61.73 (SD 5.61)	-0.68
Mean difference	0.32	2.25	**1.93**
Participant Year 2	60.99 (SD 6.64)	66.32 (SD 8.86)	5.33
Non Participant Year 2	59.4 (SD 7.16)	58.51 (SD 10.4)	-0.53
Mean difference	1.95	7.81	**5.85**

VI. IMPROVING YOUR PROCESS

In our final set of case studies we have introduced a tool to assist student plan their research process, reflect on their choices and requirements, and improve communication with supervisors and peers.

Based on suggestions in Osterweil's paper on software processes being software too [11] we decided to encourage students to use UML to model (at a high level) their personal approach and plans for their research process. This is effectively reapplying Osterweil's ideas from 20 years ago while taking into account other advances in software engineering. Where we differed was a focus on high level rather than low level process design and a view that models would be more readable than code. Our solution however does not involve the student drawing any diagrams.

From the lessons of experience with existing process descriptors it is clear that class diagrams are more likely to be updated if they do not need to be manually drawn. It seemed advice to MSc students to use dOxygen to generate class diagrams would apply as well to the diagram of the process. Working on this assumption a model of the research process was created as a template using java code stubs. Based on the lessons from our model the file was created as a generic template and students encouraged not only to fill in the template but to adjust and rename the stage / tasks (the classes) as they saw fit.

The generic template has classes for: definition, literature review, methodology design, collecting results, analysing results, drawing conclusions, and creating final output. With in each class are a set of properties representing sub-tasks, questions, or decisions related to that stage. The class also has methods representing any tools to be used in this phase. The detailed comment for the tools includes any instructions or notes on getting and using the tool.

The tool can be using by a student to plan their approach, discuss it with their supervisor, reflecting and update the plan as the research develops. Being stored as just another code file allows it to be integrated with any IDE allowing the student to work seamlessly on their code and their process. Having a generated API for discussion, complete with class diagrams and modular in the same way as code allows not only better communication but a more systematic reflection in a way that is second nature to many computer scientists.

VII. CONCLUSION

From three years of case studies we see that more than tools are needed to improve the process of developing research software. Researchers need to plan and reflect on their own process, even if it does regularly chance, and experts need to provide opportunities for process enhancement. We have presented knowledge transfer and a lack of self-reflection as bottle necks in this process, and have proposed a solutions including a new model of the research process (which need to be communicated to researches) and a tool to facilitate with this.

At a more meta level, our experiment indicates that it is possible to have a development approach that is personally tailored yet systematic, that incorporates innovative approaches but encourages reuse and sharing of tools and knowledge. Through experimentation we have developed one such approach and the tools to facilitate it.

We do not suggest that our approach is perfect, but it does provide a framework for improving the computer science research experience and in particular the communication and reflection of researchers over their research process. We see future work adding new process descriptors both generically and in more topic specific ways. Our approach if adopted can assist not only new academics but we believe experts as well.

REFERENCES

[1] Oboler, A., D.M. Squire, and K.B. Korb, *Software Engineering for Computer Science Research - Facilitating Improved Research Outcomes.* International Journal of Computer and Information Science, 2004. **5**(1): p. 24-34.

[2] Oboler, A. *Examining the use of Software Engineering by Computer Science Researchers*. in *In Proceedings of Education Students' Third Regional Research Conference, Graduate School in Humanities University of Cape Town*. 2003. Cape Town, South Africa.

[3] Basili, V.R. and M.V. Zelkowitz. *The Software Engineering Laboratory: Objectives*. in *Proceedings of the fifteenth annual SIGCPR conference*. 1977.

[4] OECD, *The Measurement of Scientific and Technological Activities - Frascati Manual 2002 : Proposed Standard Practice for Surveys on Research and Experimental Development* 2002, Frascati, France: Organisation for Economic Co-operation and Development. 255

[5] Yin, R.K., *Case Study Research: Design and Methods 2nd Edition.* 1994, Beverly Hills, CA: Sage Publishing.

[6] Basili, V.R., R.W. Selby, and D.H. Hutchens, *Experimentation in Software Engineering*. IEEE Transactions in software engineering, 1986: p. 758-773.

[7] Kitchenham, B., L. Pickard, and S. Pfleeger, *Case Studies for Method and Tool Evaluation*, in *IEEE Software*. 1995. p. 52-62.

[8] Basili, V.R. *The role of experimentation in software engineering: past, current, and future*. in *Proceedings of the 18th international conference on Software engineering*. 1996.

[9] Glass, R.L., *Searching for the Holy Grail of Software Engineering.* Communications of the ACM, 2002. **45**(5).

[10] Howe, D., *The Free On-line Dictionary of Computing* 2005.

[11] Osterweil, L. *Software Processes are Software Too*. in *9th Int. Conf. on Software Engineering*. 1987: IEEE Press.