Tutorial 4 Comp319 Software Engineering

For this and other tutorials work in groups of 2 or more. Completing the work in these tutorials will help you prepare for the examination. Remember the some parts of the examination, to gain full marks for this subject, it will be expected for you to quote from relevant research papers in the area.

Task 1 Describe pair programming

Two programmers work together to complete the programming task, one programmer writes the code while the other programmer, reviews the code, makes suggestions and comments on the approach. The two programmers swap places on a regular basis. Pair programming provides the benefit of reducing risk, because the change of a design error or serious flaw passing 2 programmers working on the task is a lot lower than 1 programmer. A programmer working in a pair is a lot less likely to implement a “hack” or shortcut which is structurally unsound. The observer has the time to review the code while it is being produced, therefore ensuring higher levels of quality. The concept of collective code ownership is important here, it means that responsibility for code that is spread between developers.

Task 2 What is your opinion on the usefulness of pair-programming, can you think of 2 benefits and 2 drawbacks to this approach.

Benefits Improved correctness, better coding quality.

Drawbacks Problems with people working together, increased costs.

Task 3 Read the papers on pair programming available on the tinyurl.com site.

(tinyurl.com/comp319) then go to tutorials/pair_programming

Summarise the findings of the different reports.

Looking at the research does not show conclusive benefits for pair programming, we can compare studies of Arisholm, Gallis and Sjøberg and Laurie Williams of the University of Utah.

The Williams et al. 2000 study showed that using pair programming there was a decrease in time to develop between 15% and 30% but this did require an increase in effort (programmer hours) between 15 to 60% with an increased in correctness of 15%.

Arisholm, Gallis and Sjøberg carried out research to determine how effective pair programming was in different contexts, for example with complex and relatively simple problems and with different combinations of staff skilling. They used a fixed set of problems and

split the developers into a pairing group and a single programming group.

They found that for most tasks the time taken was not significantly different when using a pair programming on average 84%, the amount of time was reduced but not by a large amount on average a reduction of 8%, however if the pairs were junior the increase in effort was much larger and the 111% and the time taken was larger as well. There was however a positive outcome in terms of correct solutions found overall of 7% and 73% for juniors.

One of the most interesting studies was a meta-analysis of many studies carried out by Jo E. Hannay et al. This showed small reductions in time (depending on the task complexity, less complex tasks have an improved time greater than complex tasks. They also showed that pair programming produces higher quality results for more complex problems. They also discovered a research bias in favour of pair-programming by some of the leading researchers in the area.

In general pair-programming results are somewhat mixed it seems to be able to deliver code slightly faster, of higher quality but at considerable cost. However the overall cost of the product for its whole life-cycle should be taken into account so quality improvements earlier on in development could result in savings later on, these are hard to quantify with short term studies and a more longer term study may produce more conclusive results.

Task 4 Given your findings from Task 3, what conclusions can be drawn about pair-programming. What context would it be useful? In what situations for example

is correctness more important than software production cost effectiveness, Hint think of the Therac example from previous tutorials.

For certain safety critical systems where correctness is a lot more important than cost, then pair

programming may definitely provide a lot of benefit. This is because the increase in production cost

is relatively small compared with the financial and reputational risk of getting the software wrong.