Pair programming COMP319 Notes

What is pair programming?

In pair programming, 2 programmers work together to complete the programming task, one programmer writes the code (the driver) while the other programmer, reviews the code (the reviewer). The reviewer 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. This means is a programmer is off sick, their partner can swap places and carry on with development.

Pair programming research findings

Looking at the research shows fairly inconclusive results for pair programming in terms of its benefits, 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%. The improvement in correctness would be important for projects with highly critical quality criteria such as safety critical applications for example in a healthcare environment or vehicle control.

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 solid conclusions.