COMP110 - Research Journal - Computing

1507729

November 27, 2016

1 When does a physical system compute? When does it matter?

The paper as written by Horsman, Stepney, Wagner, Kendon [1]. Supposes that by gleaming insight from other mathematical research it is possible to answer questions in the field of computer science. They propose a framework for testing if a computer is in the process of computing, though the applications of this seems rather limited I say this as in my opinion we don't work with systems to know when they compute just to know what they are computing and the results. They propose that the benefit of their research is the a unified definition for if an entity is a computer and to challenge whether or not the user is using the computer or allowing the computer to help them compute the answer themselves.

2 Experimental Investigations of the Utility of Detailed Flowcharts in Programming, are they still relevant?

The following paper by Shneiderman, Mayer, McKay and Heller [2] seeks to challenge the computational norms suggest initially by Goldstein and von Neumann [3] in 1947 about

the use of Flowcharts and the actual benefits they produce. They start by outlining their research experiments before beginning to harvest quantifiable data. They propose that through the use of experiments they can confirm if there is actually an advantage from the use of a flowchart in comprehension and debugging. However I find that their research is actually biased in how they propose the experiments they attempt to use multiple teams with a variety of skill sets, ideas and understandings. They use students who are just learning and assign them into teams allowing them as long as they need to complete the tasks. These freshly created teams could potentially have issues in how they function and problem solve leading to issues with how they complete tasks, the stages of team development as suggested by Tuckman [4] are a prime example of this. I would think a more appropriate way of approaching the task would be after a screening process to ensure that all participants show a normalized standard of understanding and competency they are then presented with a coding task individually, and upon leaving the task their results are processed to prevent cross contamination of the experiment between subjects. The fact they allow as many roaming variables as they do prevents the results of their experiment to be considered valid or reasonable. Finally they then go on to say that the results are inconclusive and that further research needs to be carried out.

3 A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space

The paper by Gilbert, Daniel, Johnson and Keerthi [5] looks into a fast method for checking the distances between points of irregular shapes to confirm if they have collided and if they are within a close proximity to each other. There is a huge amount of mathematics within this paper, to back up the suggested ideas however it's beyond my current ability to understand fully, it seems to suggest through the use of multiple

vectors, it's possible to check for distance across any number of irregular shapes.

4 Go To Statement Considered Harmful

The letter as written by Dijkstra [6], is all about the disadvantages of the Go To statement in higher level code. He puts forward the point that if a program is structured correctly it shouldn't need to use of a Go To statement and that the only place that such a statement should be used is in plain machine code. His point is logical and it does make sense from a maintainability perceptive as it makes it much harder to follow the codes processes. It's a sound notion that if a program is structured correctly it shouldn't need to jump back on itself. It is a stretch to say it is harmful, in a program needs to be adapted or have future iterations built from it, then yes it is harmful however if the code is for one use and isn't going to be seen again, then if it works it's not harmful just sloppy.

5 Conclusion

The following papers all speak on different topics, ranging from theoretical and abstract to physical. The first paper speaking on when a physical system computes, which even with the application of maths seems like more of a philosophical debate than that of a computer science question. If there is one paper that stands out more than any of the others it would be this one, it seems to have the least practical application in my opinion and doesn't seem to link back in anyway to the other papers.

The second paper on flowcharts has obvious practical applications, it questions the actual benefits gained through the use of flowcharts within the programming workflow. Even though their results were inconclusive it's still an interesting notion which attempts to subvert the common thinking of the time. It could be said that because their work was inconclusive as to which was more useful that in it's self proves the point of that

flowcharts aren't as useful as they appear. I would still say that the results they produced were incorrect as their method for testing allowed to many rogue variables.

The next paper again has practical applications as mentioned at the start of their work "In ROBOTICS and other fields, such as computer-aided design and computer graphics." It's very clear the practical applications for a fast algorithm to detect the distances between irregular objects, and the applications of this are numerous especially within the games industry but also beyond that.

Finally the paper on Go To statements seems to more of a rant though it is an interesting notion if nothing else, it seems a little dramatic to refer to the Go To statement as harmful it's really more of a pain than an outright harm to anything. I can't disagree with the deductions made as it seems only logical to have things structured so they don't require code to travel back on itself multiple times making it cleaner and easier to read as well as improving it ability to be maintained and expanded upon.

The majority of these papers seem to relate back to good industry practice is the only conclusion I can draw where they ask researchers to be vigilant and make sure they are using the computer to do the work and not using the computer to help you do the work. Next we looked at if flowcharts really do help to produce code in the industry and if they help producing code, paired with our example on a fast algorithm for checking detection on irregular shapes. and finally leading to the use of Go To statements and on structuring work so that you don't fall into the bad practice of their use.

References

[1] C. Horsman, S. Stepney, R. C. Wagner, and V. Kendon, "When does a physical system compute?" in *Proc. R. Soc. A*, vol. 470, no. 2169. The Royal Society, 2014, pp. 1–25.

[2] B. Shneiderman, R. Mayer, D. McKay, and P. Heller, "Experimental investigations

of the utility of detailed flowcharts in programming," *Communications of the ACM*, vol. 20, no. 6, pp. 373–381, 1977.

[3] H. Goldstein, "Planning and coding problems for an electronic computing instrument, part ii, vol i. rep. prepared for the us army ordinance dept., 1947. reprinted in von neumann, j," *Collected Works*, vol. 5, pp. 80–151.

[4] B. W. Tuckman, "Developmental sequence in small groups." *Psychological bulletin*, vol. 63, no. 6, p. 384, 1965.

[5] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[6] E. W. Dijkstra, "Letters to the editor: go to statement considered harmful," *Communications of the ACM*, vol. 11, no. 3, pp. 147–148, 1968.