# How 'Go to statement considered harmful' influenced computing.

**COMP110 - Research Journal**

1706966

November 15, 2017

## 1 Introduction

This paper will be looking at the historical context and influences made by the paper Go to statement considered harmful [1]. It is important to note that [1] was originally penned in 1968, and the author of this research journal has minimal knowledge of how computing was during that time, or even in the present time period, at the start of this paper. However with the use of other research articles a clear insight will be shown into how [1] has changed the computing world at present.

## 2 Summary of chosen paper

The instant insight that is be taken from [1] is that Dijkstra's opinion on the use of go to statements in code base directly relates to the skill of the programmer. Such that if more go to statements are used in a piece of code, the lower the quality of the programmer writing it. This clearly shows Dijkstra's disapproving thoughts of the go to statement and he goes on to mention possible better ways to have your code bases work flow.

Ways mentioned are conditional clauses (if statements) and repetition clauses (while statements). Though it is mentioned that the later is unnecessary 'because we can express repetition with the aid of recursive procedures' [1, p.147] the topic is not touched on more and just continues with repetition clauses. These clauses mentioned are widely used throughout programming currently, though from how it is stated in [1] it comes across that, although these were available for programmers to use at the time, they may not have been as commonly used then as they are today.

Dijkstra goes on to loop back around to talk about the go to statement and how 'unbridled use of the go to statement has an immediate consequence that it becomes terribly hard to find a meaningful set of coordinates in which to describe the process progress' [1, p.147]. I believe this extract refers that if the go to statement is used without proper consideration it can lead to difficulty when trying to interpret how the program is running, and where it is in the process at any giving time, this is reconfirmed in [2, p.37].

Towards the very end of the article it is mentioned that in [3] that the go to statement has been proved to be unnecessary using a jump-less flow diagram, but Dijkstra goes on to say how this is not recommend as 'the resulting flow diagram cannot be expected to be more transparent than the original one' [1, p.148]. This points back to the article title that the go to statement should be considered harmful for programs, but not entirely unusable as they can lend themselves to improving program readability.

## 3 Influence regarding other papers

Many papers have gone on to cite [1] over the years, using it to confirm and give a basis for some of their claims. The use of this paper alone give credence to it's impact on software development. This section will look at how extracts have been used in other papers, and merge similar concepts, giving a better look at the specific parts that have moulded the way code base is used today.

Wendy Hui Kyong Chun references Dijkstra's work over multiple papers, in two of them the same part is used to convey similar messages. 'shorten the conceptual gap between static program and dynamic process, to make the correspondence between the program (spread out in text space) and the process (spread out in time) as trivial as possible.' [1, p.137] is used in both [2] and [4] to pull out the idea that 'go tos make it difficult for the source program to act as a legible source.' [4, p.303]. This matter gets expanded on leading into the idea that code is not purely about the end product but that the source itself is what truly matters. the argument goes on to say that debugging code is not the only reason for this but also critical analyses. I believe it could be taken deeper than this, with the readability and easily understandable code being given a much higher prestige in the age of open source software and group collaboration, it becomes more and more imperative that programmers strive for these traits in their work.

This paper [5] focusing on teaching programming to beginners uses Dijkstra's paper to argue that because go to statements have been marked as harmful 'why should the students learn a concept which is known to have downsides.' [5, p.94] although it is understandable where they are coming from, it should be noted that it wasn't concluded for go to statements to not be useful, just harmful when over used or used where another clause would have been better. Go to statements should be taught but done correctly and not as the main base of a language.

## References


[1] E. W. Dijkstra, "Go to statement considered harmful," *Communications of the ACM*, vol. 11, no. 3, pp. 147–148, 1968.

[2] W. H. K. Chun, "On software, or the persistence of visual knowledge," *grey room*, no. 18, pp. 26–51, 2005.

[3] C. Böhm and G. Jacopini, "Flow diagrams, turing machines and languages with only two formation rules," *Communications of the ACM*, vol. 9, no. 5, pp. 366–371, 1966.

[4] W. H. K. Chun, "On" sourcery," or code as fetish," *Configurations*, vol. 16, no. 3, pp. 299–324, 2008.

[5] F. Huch, "Learning programming with erlang," in *Proceedings of the 2007 SIGPLAN workshop on ERLANG Workshop*. ACM, 2007, pp. 93–99.