# Jacob Blazina

# CS 325

# Dr. Salter

# November 5, 2021

# Summary for “Just Say No to Poorly Designed Software” by Charles Hannon.

Charles Hannon conjectures that software developers should consider end users when designing information technology systems. Hannon describes flaws in multiple information systems that may have been prevented with proper consultation of faculty members. Using examples such as Google and Blackboard, the author surmised that successful companies design their software with a focus on user experience. The author presents a solution to poorly designed software; decision makers must turn away software vendors that do not engage with the end users.

Hannon claims that software designers do not consult with faculty because decision makers have differing success criteria than the end users. Goals related to maintainability, reliability, and cost take precedent over user satisfaction. Decision makers are wary of contracting with young companies due to the risk that their software won’t be supported in the future. This risk aversion drives administrators to select mature vendors, even if these vendors’ products lack critical features for the end users. The author postulates that software companies hesitate to field alternate design methodologies due to market pressures. Hannon proposes that this paradigm shift must be driven by end users.

To enact this change, Hannon suggests that educators must teach students to be critical of poorly designed software. He concludes that it is time to engage with decision makers and software designers to create information systems that are user focused and flexible.

**References**

Hannon, C. (2004, October 22). Just Say No to Poorly Designed Software. Chronicle of Higher Education, p. B16. Retrieved from http://ezp.lib.cwu.edu/login?url=https://search.ebscohost.com/login.aspx? direct=true&db=a9h&AN=14905357&site=ehost-live

**Discussion Paragraph**

**How does this article apply to the field of computer science?**

A majority of things related to Computer Science have a direct impact and/or interface directly with humans. Creating a frontend application, an API for bank transfers, or an algorithm for more efficient warehouse storage all directly or indirectly have a human element involved. Any Computer Scientist must always remember WHY they are writing code, and WHO they are writing it for. I’m currently a software development manager, and from my experience can say that easily 90% of all feedback/suggestions/complaints I get have to do directly with User Experience. Very few people care about technical details, feature velocity, or feature volume. Most people are very happy to use the most base and simple software, as long as it is intuitive and user friendly.