Thomas Mintun - 14087314
CS4500-Chakraborty

1A.

“The Role of Software in Spacecraft Accidents” is about the common factors in five spacecraft accidents caused by software. Changes to the aerospace systems are suggested by the author, Nancy G. Levinson, “although a comprehensive discussion of how to solve these very complex problems is beyond it’s (the paper’s) scope.” Levinson maintains a very serious tone describing five spacecraft accidents, flaws in safety culture of those missions, management and organizational factors of the crashes, and a technical deficiencies section which is the bulk of the paper. Levinson believes “the cultural and managerial flaws manifested themselves into the form of technical deficiencies.” She ends with conclusions, acknowledgements, and resources.

1B.

The paper summarized above was interesting and thorough, albeit long and somewhat technical. Levinson was correct to say that there are similarities in all the crashes and must be caused by an incorrect system that goes beyond technical issues, "and include reasons why those failures or design errors were made." She is saying that yes technical errors were made, but these technical errors are a result of something grander. As Levinson states in the first section on page 2 systemic factors are those related to the overall production system of which the technical device is only a part of. Most of these errors seem trivial, and the solution presents itself as obvious in hindsight. The errors seem a result of laziness, engineers not taking total responsibility, complacency, and misunderstanding the risks. More so, on the bottom of page 5 she states "accidents involving software are usually *system accidents* that result from dysfunctional interactions among components, not from individual component failure. All of these accidents **(**as well as almost all software-related accidents known to the author) resulted from the software doing something wrong rather than the computer hardware or software failing to operate at all. In fact, in most cases the software or hardware components operated according to their specifications, **but the combined behavior of the components led to disastrous system behavior.**" The bold emphasis (mine) highlights that size and complexity of the projects contributed more to the failures than the technicalities of individual components. Furthermore, on the top of page 6, "All accidents… displayed aspects of system accidents... caused by **interactive complexity and tight coupling."** Bold emphasis (mine) added again shows that the overall size of the projects was the main source of failure. Levinson goes on to describe how complex the systems are, and that the interactions between components cannot all be planned or guarded against. The problem being the huge size of the projects is ambiguous, but it is the common overarching theme in all five accidents. It is also the common thing that made failure of these missions possible.

For the most part Levinson is sound in her arguments and analysis. To be fair to the teams that produced these failures though, hindsight is always 20/20. Levinson takes a very serious tone that is appropriate for the paper. Her problem is she begins to contradict herself because she is speaking from an authoritarian point of view, and she does not present solutions to the problems she states were inexcusable. She stated it is beyond the scope of the paper to go into detail of how to fix the accidents. Levinson does not have the stress of a mission, working with a team, or the stress of a time deadline while writing this, and still cannot contribute novel ideas on how to fix the problems. This shows the scope of the problems, and how complex the systems she is criticizing are. In the second paragraph of page 16, "… it was not wise to change the software that had worked well on the Ariane 4 unless is was proven necessary to do so **was well founded: errors are often introduces when software is changed…"** Then on page 22 in Section 5.4 she contradicts herself again saying "The software test and simulation processes must reflect the environment accurately. Although implementing this principle is often **difficult or even impossible for spacecraft…"** Although the Ariane 5 was a costly crash to Levinson no human lives were lost. It is estimated the crash cost $370 million dollars, which really is not that much money to a country. No human deaths occurred in the Ariane 5 accident in 1996. Perhaps the team knew it was difficult or impossible to simulate and implement new software, so they sent the rocket knowing it may crash. They received data, and only 1 other major failure has happened in the Ariane 5 program in 97 total launches.

2A.

Kweku Ewusi-Mensah begins the article “Critical Issues in Abandoned Information Systems Development Projects” with the story of companies in different industries trying to come together to offer “the most advanced registration system in the combined industry of travel, lodging, and car rental.” The project was canceled after a lot of time and money was spent on it. He states several factors that he believes caused the cancelations of large scale information system projects. Describing IS projects in general is next and followed by the bulk of the article that is a section titled “Factors Contributing to Project Cancellation.” In this section he uses evidence from real world IS project failures to back up his factors that he believes causes the cancellations of IS projects. Finally, he concludes with advice on how to properly stop IS projects by minimizing damage to the companies and workers behind them before his actual conclusion section. He says executives should have a systemic examination of what went wrong and why on the last page, and share this information to “help diminish the frequency and of abandoned development projects and improve the practice of information systems development throughout history.”

2B.

At the bottom of the first page, Kweku states that “it is apparent that such cases are an industry wide problem,” talking about failures of large teams to work together on complex problems. Today in 2018, this has not changed. Software teams today will still have unforeseen problems and complexities arise due to the largeness of their projects. Stating IS projects are group activities (on page 2) means they are subject to communication, coordination, and interactions flaws. This certainly is still true today in 2018 as seen with spacecraft failures of SpaceX, Twitter banning users for good or bad reason, and Tesla’s self-driving cars crashing. Furthermore, he states “large size of the project, complexity of problem domain…” are risks. This is also still true today. In class, Professor Chakraborty has stressed the size and complexity of projects is the main cause of errors. Chakraborty has stated in class “you cannot predict how changing one thing will affect the rest of the variables because there are so many variables and infinite amounts of interactions. It is not a linear thing.” Lack of responsibility and complacency as seen in the articles the class read about spacecraft accidents certainly apply here, and some of those accidents were recent, so the claim this paper is outdated is refuted. All of the IS problems Kweku states are relevant to software teams today, most importantly: lack of responsibility from senior management (Kweku mentions senior management at EDC to have meetings on a project), workers not understanding what they are doing and why they are doing it, and projects going over budget as the Confirm project described on page 6. “IS project cancellatuin is a multi-dimensional and multifaceted issue with different interacting parts” is an undeniable thesis that still holds true in 2018.