

From Bridges and Rockets, Lessons for Software Systems

C. Michael Holloway
NASA Langley Research Center, Hampton, Virginia

Keywords: safety, high integrity systems, software engineering, accident analysis, history

Abstract

Although differences exist between building software systems and building physical structures such as bridges and rockets, enough similarities exist that software engineers can learn lessons from failures in traditional engineering disciplines. This paper draws lessons from two well-known failures--the collapse of the Tacoma Narrows Bridge in 1940 and the destruction of the space shuttle Challenger in 1986--and applies these lessons to software system development. The following specific applications are made: (1) the verification and validation of a software system should not be based on a single method, or a single style of methods; (2) the tendency to embrace the latest fad should be overcome; and (3) the introduction of software control into safety-critical systems should be done cautiously.

Introduction

Articles and books abound warning about the inadequacies of software development practices. Often, these inadequacies are attributed primarily to differences between software engineering and traditional engineering disciplines. Differences commonly cited include the following: the inherently discontinuous behavior of software as opposed to the inherently continuous behavior of physical systems, the fact that software does not wear out like physical components, and the relative youth of software engineering as compared to traditional disciplines.

Differences such as these exist, but do not justify the attitude that software is so different that nothing can be learned from traditional engineering disciplines. There is much that can be learned, as others have recognized. For example, in a 1994 article Nancy Leveson drew parallels between the early development of high-pressure steam engines and current software engineering. She wrote, "Risk induced by technological innovation existed long before computers; this is not the first time that humans have come up with an extremely useful new technology that is potentially dangerous. We can learn from the past before we repeat the same mistakes.

This paper is based on a similar premise, but uses a different approach. Instead of looking at the development of a particular technology, we look at two specific failures from two very different technologies: bridges and rockets. Studying failures was chosen because, as Henry Petroski has written, the lessons learned from failures "can do more to advance engineering knowledge than all the successful machines and structures in the world." Bridges and rockets were chosen for two reasons. First, building bridges is one of the oldest engineering activities, and building rockets is one of the youngest. Second, the collapse of the Tacoma Narrows Bridge in 1940 and the destruction of the space shuttle Challenger in 1986 are two of the most widely known engineering failures of this century.

The discussion of both failures will be necessarily brief and incomplete, and will contribute nothing new to the understanding of either. The paper's contribution is in the direct application of lessons from these failures to software engineering.

The structure of the remainder of the paper is simple. First, the Tacoma Narrows Bridge collapse is described, and four lessons from it are explained. Second, the Challenger accident is described; how this accident reinforces lessons from Tacoma Narrows is explained; and one additional lesson is added. Third, applications of the lessons are made to software systems. Finally, brief concluding remarks are made.