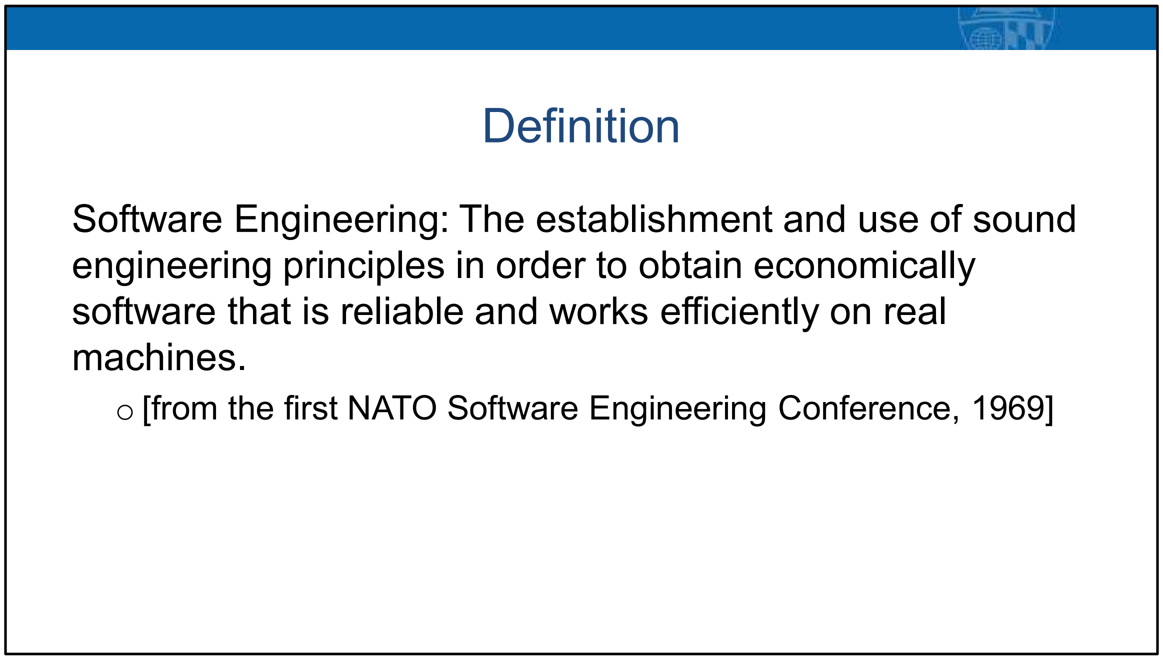
Origins

Now that we have seen something about WHAT software engineering is, let's now look at WHY we have software engineering. Let's look at the origins of software engineering.

The person generally credited with defining the term "software engineering" was Margaret Hamilton. In the mid 1960's while she was working with NASA on the Apollo project, she realized that there needed to be more than just programming involved in the creation of code. She said that what was needed was an engineering approach to the development of software.

In 1968 there was an international conference on Software Engineering sponsored by NATO, The North Atlantic Treaty Organization. The purpose of the conference was to formalize the definition of software engineering and to start to develop goals and principles of software engineering. We will see some of the results of that effort later in this module.

You might ask why the conference was sponsored by a military alliance. Well, if you think about it, the military and the space program were the largest consumers of computer hardware and software in the world. It was in their best interest to encourage the production of high-quality software.



Definition

Software Engineering: The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

- [from the first NATO Software Engineering Conference, 1969]

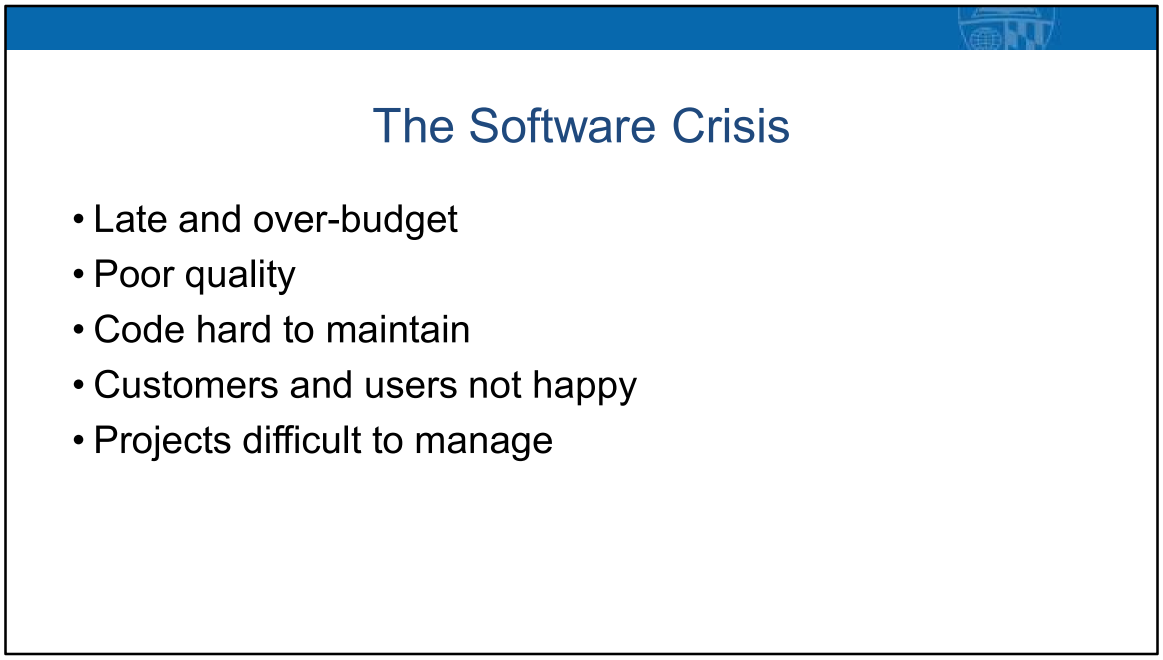
Here is the definition of software engineering that was agreed upon at the conference. The conference proceedings were published in 1969.

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

This definition highlights several of the issues that software developers were having at the time. You an see that they knew they needed to apply well defined, repeatable processes and procedures. Building software economically was important. Reliability means that the software does what it is supposed to do. Efficiency has always been an important property of software.

The timing of the conference, at the end of the decade of the 1960s is a reflection of the fact that the size and complexity of software systems were beginning to get out of hand. The traditional approaches to building software were not sufficient. An engineering process is what was needed. At the beginning of the decade, computers were not large or powerful enough to allow very big programs, so many of the problems with software didn't become apparent until later, when larger computers allow us to build larger programs that began to exceed our intellectual control.

And the problem has been getting worse ever since.



The Software Crisis

- Late and over-budget
- Poor quality
- Code hard to maintain
- Customers and users not happy
- Projects difficult to manage

There's actually a term for the problems that software developers were facing by the end of the decade in the 60s, it's called the Software Crisis.

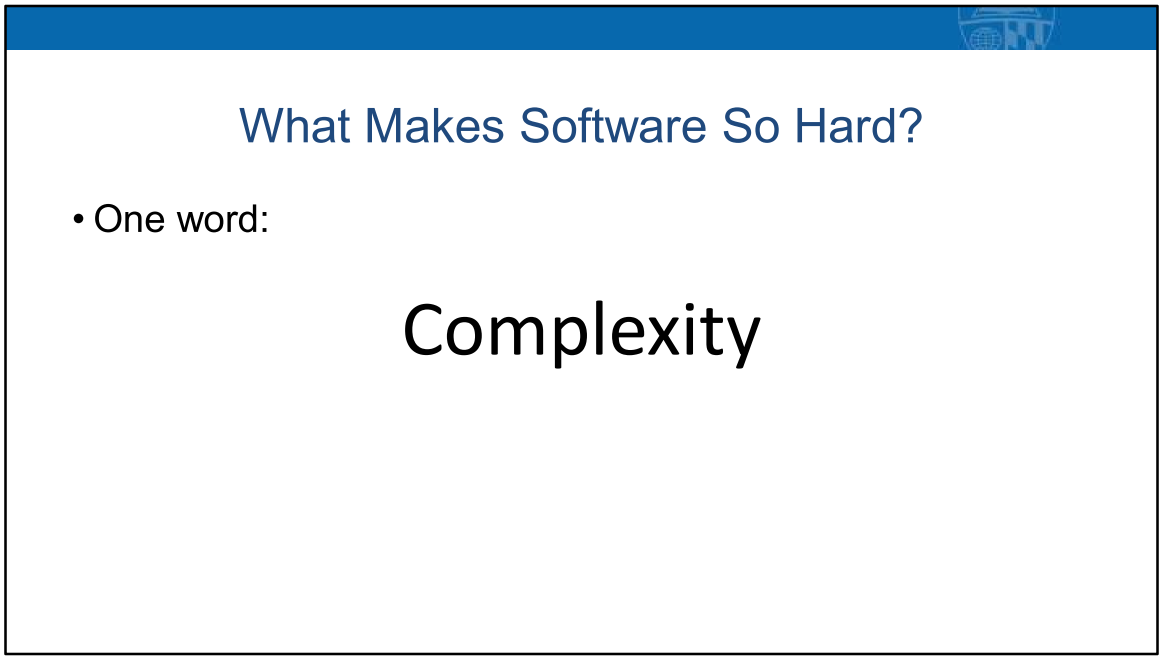
I think the central symptom here was the poor quality of software. We just didn't know how to write software correctly to avoid bugs. And as a result, the software was late, because we couldn't remove the bugs in time to get the software out the door.

As a result of our hurrying to deliver on time, we left a lot of bugs in our software.

We would try to get the software to work by hacking at it and hammering at it, and finally we'd get something to work and get the software out the door. But when you tried to make changes to the software, to either fix bugs, or to improve it, we tended to make the software worse than it was to begin with. If you wanted to add new features, a lot of times what you had to do is just rewrite the software from scratch

When we finally did get something to work, it wasn't really what the customer needed or wanted.

Projects were difficult to manage, because we tried to apply management techniques that had worked well for things like manufacturing, Software is just different. And we couldn't apply those same management techniques to the development of software. We had to learn how a new way to manage software projects.



What Makes Software So Hard?

- One word:

Complexity

The reasons for the software crisis could be summed up in one word, complexity.

What was needed was a set of tools, processes and procedures that would help us manage the growing complexity of our computer systems.

Good news: we got off to a good start in 1968 by beginning to think about software engineering.

Bad news: the problem keeps getting worse. The complexity of our software systems is outpacing our ability to manage it. We have to keep coming up with better and better ways to handle the complexity problem.

In fact, the term Software Crisis may not be the right term here. Some people have suggested that it would be better to call this the Software Chronic Affliction.

In the next video, we'll explore this a little bit more.