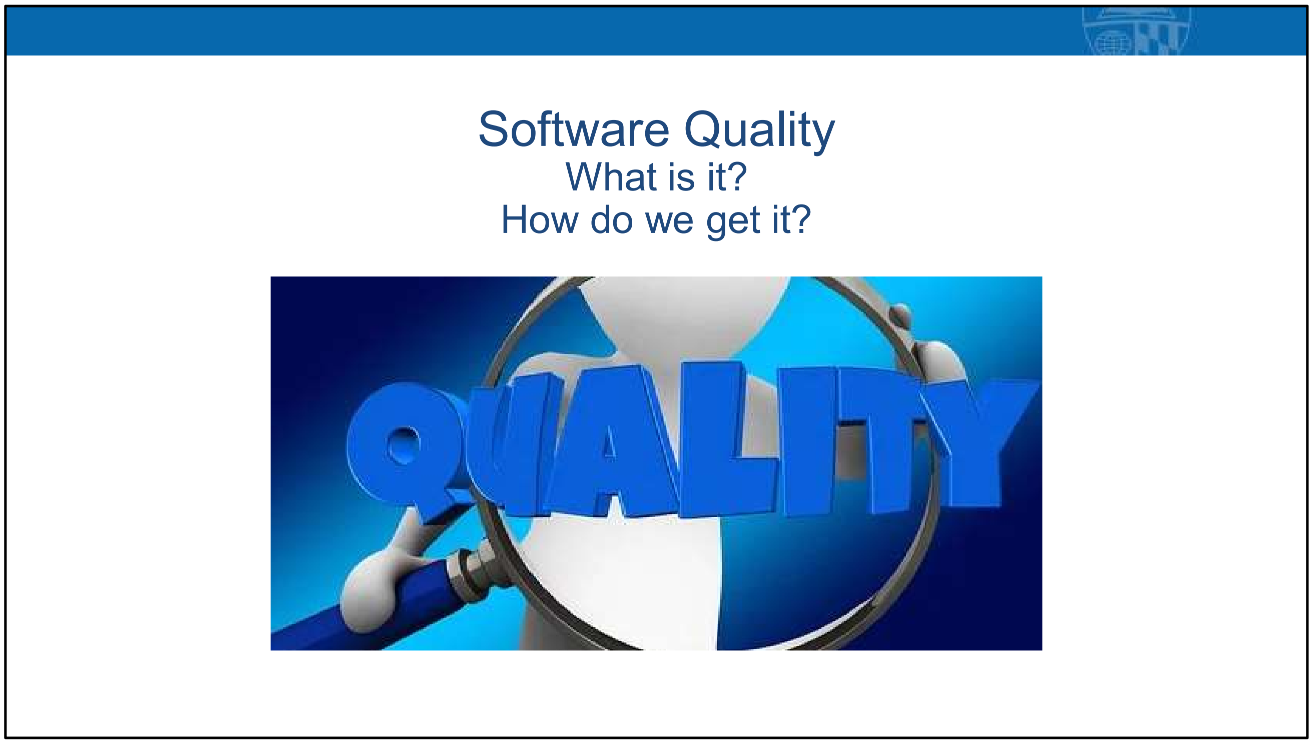
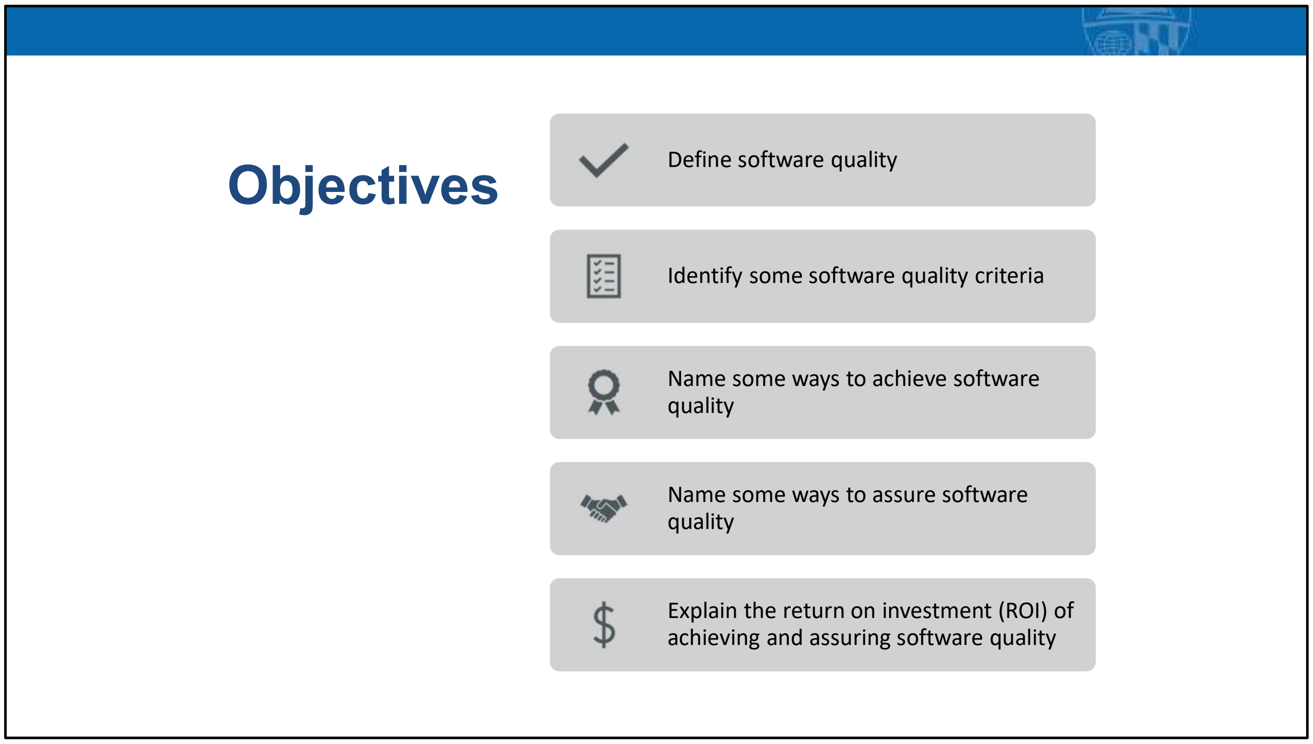


Software Quality

What is it?
How do we get it?

This is the first video on software quality. We will be discussing the definition of quality and ways in which we can achieve it in our software


Objectives

- ✓ Define software quality
- Identify some software quality criteria
- Name some ways to achieve software quality
- Name some ways to assure software quality
- $ Explain the return on investment (ROI) of achieving and assuring software quality

More specifically, here are the things we expect you should be able to do once we are done here.

You should be able to define software quality.

You should be able to identify some of the criteria that we are looking for in quality products. These are similar to the software engineering goals we discussed in the first module of this course.

You will be able to name some ways in which we can achieve software quality. You will also be able to name some ways to assure ourselves and others that we are building a quality product.

Lastly, you will be able to explain why it is a good investment to do the things to achieve and assure quality.

What is Quality?

- One definition: Meets requirements
- Better definition: Satisfies the needs of the customer

The traditional definition of quality is “meets requirements.”

This was useful for situations in which the requirements were stated in terms of things that could be measured. There would typically be quality inspectors at various points in the assembly line. If the inspectors noticed that things were beginning to get out of spec, they could shut the line down so they could figure out what was going wrong.

Consider the manufacturing of an automobile or a desktop computer. These types of products are made up of lots of subsystems and parts, each of which must adhere to a specification. The parts can be manufactured by several different vendors, so adherence to the specification is important.

Similarly, most software is made up of subsystems and components, each of which must interoperate with the others. So, specification of the interface and behavior of the components and subsystems is important. These specs are not measurable because they are not physical, but they can be inspected and tested for conformance. We will revisit this later in the course when we cover software reuse.

Now, what about the requirements for the entire software system product? As we have seen, these requirements are usually written in the form of stories or use cases. We can certainly perform tests to ensure that the software does what we say it should do.

But maybe the requirements are wrong.

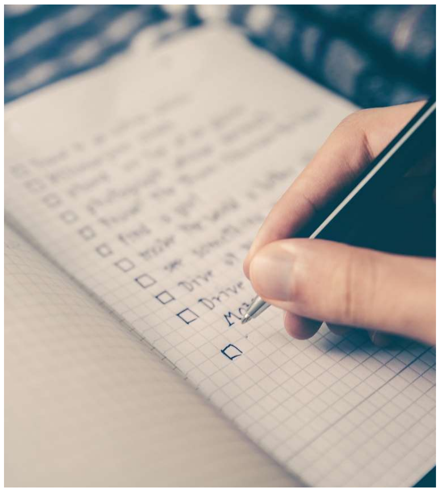
It is important to consider the goals of customers or other stakeholders before thinking about the software requirements. There should be traceability between the requirements and the goals of the stakeholders.

In that way, we will have a better chance of building a product that satisfies the needs of the customer or other stakeholders.

This is the reason we write Vision documents.

Some Software Quality Criteria

- Correct
- Efficient
- Flexible
- Robust
- Maintainable
- Reliable
- Reusable
- Testable
- Usable
- Understandable



Remember the first module of this course when we were discussing the software crisis and the goals of software engineering? This list is an expansion of the list of goals we had in that module.

You will recognize most of these as non-functional requirements. It is possible to write non-functional requirements in a measurable way, so it should be possible to write test cases that show that the software meets them.

Achieving Software Quality

- What can we do to build quality into our product?
  - Materials
  - Workers
  - Standards and conventions
  - Tools

Now we will address two categories of actions: Achieving and Assuring quality. They both start with A, so we need to distinguish between them.

Achieving quality means doing things that build a quality product. Assuring quality means convincing ourselves and others that we are building, or have built a quality product.

Here are some things we might do in order to achieve quality.

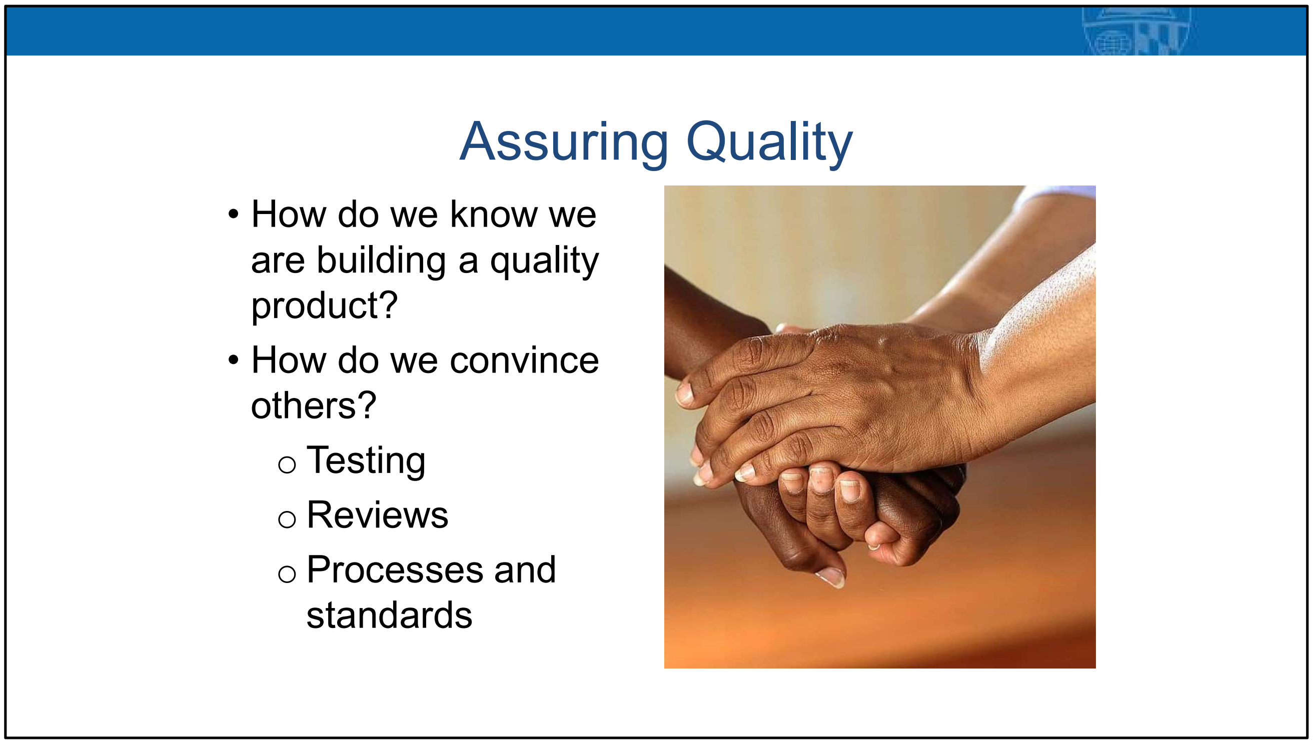
We ought to build our product out of high-quality materials, components or subsystems. We want to avoid any component that could be a single point of failure for the entire system.

We should use skilled workers to build our product. More experienced or highly trained workers may be a bit more expensive but in the long run they can save you (as we shall see in a couple of minutes).

They should follow good practices, standards and conventions. For programming, this might entail following coding conventions.

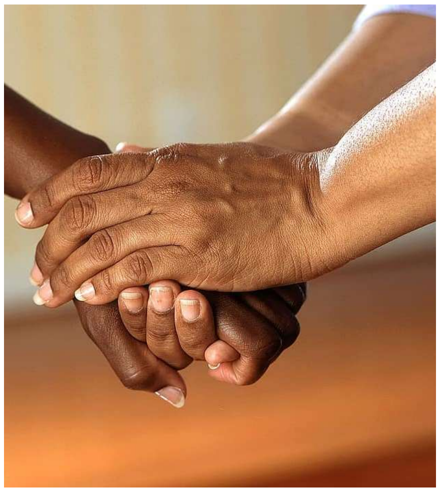
Using good tools can both make it easier to build our product and enable a higher quality production as well.

Note that these suggestions apply to building cars, computers, and houses, as well as they do to building software.



Assuring Quality

- How do we know we are building a quality product?
- How do we convince others?
  - Testing
  - Reviews
  - Processes and standards



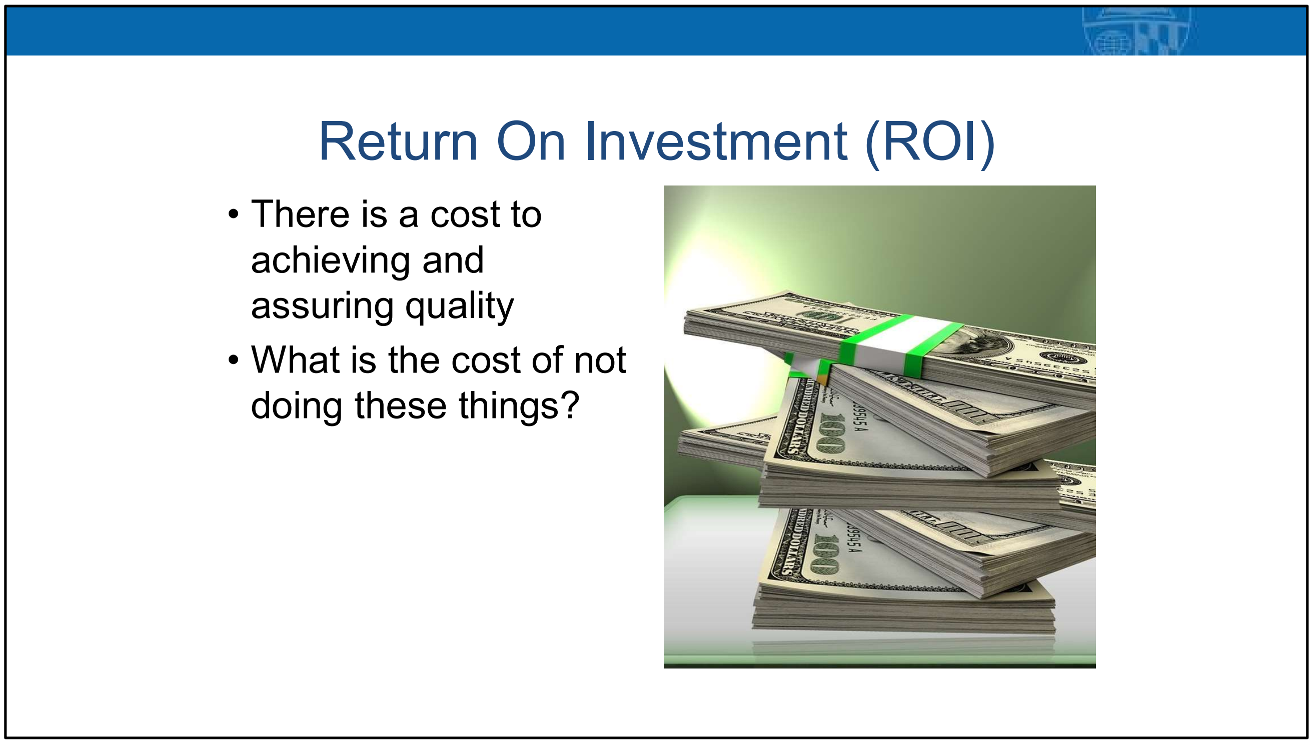
Assuring quality means convincing ourselves and others that we are building a quality product.

We will cover testing in a later module of this course. For now, let us agree that as we run tests, if we are encountering fewer and fewer defects, that this shows that quality is improving

What do we do before we have enough software in place to do testing? We can run inspections and reviews. We can review requirements, designs, code, and even testing scenarios. We should also perform reviews of our processes and standards.

Following standards and conventions and other processes not only can help us achieve quality, but the mere fact that we have good processes in place, and that they are being followed, gives us assurance that we are more likely to build a quality product.

Putting this another way, you could say, "If you've got quality and you know it… clap your hands" <clap> <clap>.



Return On Investment (ROI)

- There is a cost to achieving and assuring quality
- What is the cost of not doing these things?

Now wait a minute. Isn't all this going to drive up my costs and take more time to develop my product?

Well, yes, there is a cost to doing the things we talked about. And yes, it does take time to perform them.

But what is the cost of not doing these things?

One of the obvious costs is scrap and rework. If you are manufacturing small parts such as computer chips and they start failing their tests, it is cheaper to just scrap the defective pieces. This is a cost. If we are building larger systems, such as TVs or computers, and they fail the tests, it is more cost effective to send the defective products be repaired. These refurbished products are usually sold as refurbished, or what Amazon calls renewed, at a discounted price. There is a cost to fix the product, combined with the loss of profit on the discounted price, but the cost penalty is not as high as what you would have if you scrapped the product.

If we continue to sell defective products, we will get low customer satisfaction ratings. This will result in a loss of business.

Some companies have even had to go out of business because of poor performance.

What if you get sued? Once you get the lawyers involved, that can get really expensive.

But wait, it gets worse. What if your defective product ends up doing something bad such as destroying property, injuring someone, or even killing them? You are now going to have even bigger legal problems.

Hopefully you will agree that the cost of doing things to achieve and assure quality is much less than the cost of not doing them.

Quality pays for itself.

Next

- Defects

Coming up in the next video is a discussion of defects.

One thing I think we can agree on is that defects are bad. Not having defects is good.

There’s more to it than that, but that’s the bottom line.