

So...let's talk about some different types of metrics and how they might be used.

There are a number of different types of metrics that organizations use. For our discussion I'm going to break them down into three categories: process metrics, product metrics, and quality metrics. Please note in advance that the categories are not mutually exclusive when it comes to quality metrics. Process metrics measure something associated with a process...like the total effort spent on each phase in a project life cycle, an estimate of time remaining for project completion, maybe average inspection rates (measured in, say, pages inspected per hour), and so forth. As you can see by these examples these are characteristics of various processes.

Product metrics measure something associated with a product...like product size (maybe in lines of code), the number of defects per thousand lines of code, or maybe the control flow complexity of a software component.

Quality metrics measure something that is associated with the quality characteristics of a process or product. Examples include measures like the percentage of milestones satisfied on time, the defect density of a delivered product, or the percentage of code exercised under test. They are indicators of general or specific product or process quality. The remainder of this lecture will focus on examples of quality metrics.

Specifying & Measuring Quality

Quality Requirement	Definition				
Interoperability	The extent of effort required to interface the product with other products.				
Maintainability	The extent of effort required to make changes to the product.				
Portability	The extent of effort required to transfer the product to other HW/SW environments.				
Usability	The extent of effort required to learn, operate, and interpret the product.				
Reliability	The extent to which the product consistently performs its intended functions.				
Reusability	The extent of effort required to reuse product components in other products.				
Testability	The extent of effort required to verify correct product operation and performance.				

Before you can measure software quality, you must be able to specify what software quality is...with respect to a specific software product. In the software engineering world there are some industry-wide practices that correlate with good software quality and that can be designed into software product components. Examples of such practices would be that a software component follow structured programming principles, that it is well-documented, etc.

There's also a whole category of non-functional requirements in software engineering that are often referred to as quality requirements. These requirements often end with "ility"...maintainability,

testability, and so forth. A definition of these frequently-specified quality requirements is illustrated in this table...and many leading-edge software organizations include at least a subset of them on most projects.

As you can see, almost all of these definitions begin with the phrase "the extent of"...so what does that mean? Simply put, it means how easy or difficult. For example, maintainability means how easy or difficult would it be to change the product. A product with good maintainability would be a lot easier to change than a product with poor maintainability...that is, it would take less effort.

Most people would agree that these seem like desirable characteristics for many software projects. The challenge for the software engineer is how to build those characteristics into the software product and demonstrate that the product does, in fact, satisfy those requirements.

Let's see how that might be done by using metrics.

6

Quality Requirement Visibility

QUAL RQT	RQTS	DSGN	CODE	TEST	OPNS	MAINT	TRANS
Interoperability					X		X
Maintainability						X	X
Portability							X
Usability				X	X	x	
Reliability				X	X	X	
Reusability						x	X
Testability				X		X	X

Conference on Software Development, 1990.

This table shows where in the overall product life cycle evidence of each of the quality requirements is traditionally able to be observed. We'll refer to this as the requirement's visibility. And by traditionally, I mean in projects where metrics are not used. The first four columns in the table correspond to the developmental phases for the product. Operations refers to the time period the product is in production mode, maintenance refers to maintenance mode where the product will undergo changes for whatever reasons and in which the requirements, design, code, and test activities would be repeated to varying extents, and transition refers to the situation in time when the product may need to be ported to a different hardware or software platform. The X's indicate where the visibility appears.

One thing that's pretty obvious is that...traditionally...visibility starts to occur very late in the product life cycle...during the testing phase at the earliest...and for many of the quality requirements, even later. In this situation, if a quality requirement is not met, there will be substantial rework required. And...we've already seen how costly that can be.

Let's compare this traditional approach to one that incorporates metrics. 7 This table illustrates where metrics can be used to provide early visibility and evidence as to whether a **Quality Requirement Visibility** specific quality requirement is, in fact, being built into QUAL RQT RQTS DSGN CODE TEST OPNS MAINT TRANS the software product. Note that we can get visibility 0 0 Maintainability much earlier in the product life cycle...in all cases by Portability O O X
 Usability
 O
 O
 X
 X
 X

 Reliability
 O
 O
 O
 X
 X
 X
the design and code phases...and in some cases as early on as the requirements phase.
 Reusability
 O
 O
 X
 X

 Testability
 O
 O
 O
 X
 X
O where metrics can be used X traditional visibility Now...let's see how this can be done. Adapted from J.M. Demasco, Software Quality Engineering Workshop, presented at National Conference on Software Development, 1990.