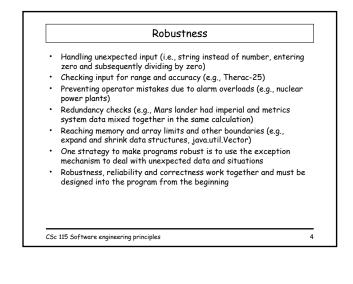
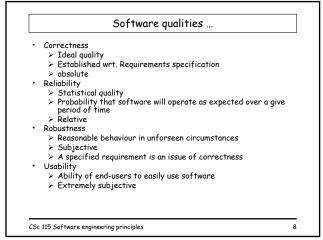


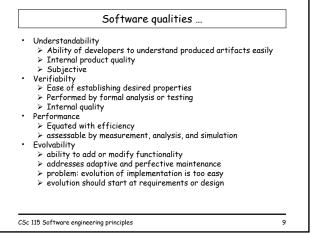
Software qualities
Software engineering is concerned with software qualities
Qualities (a.k.a. "ilities") are goals in the practice of software engineering
External qualities
visible to the user
reliability, efficiency, usability
Internal qualities
the concern of developers
they help developers achieve external qualities
verifiability, maintainability, extensibility, evolvability, adaptability
Product qualities
concern the developed artifacts
maintainability, understandability, performance
Process qualities
> deal with the development activity
products are developed through process
> maintainability, productivity, timeliness

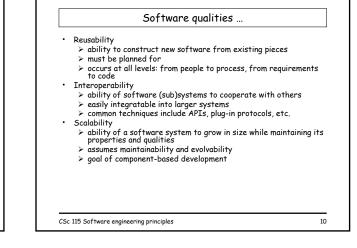


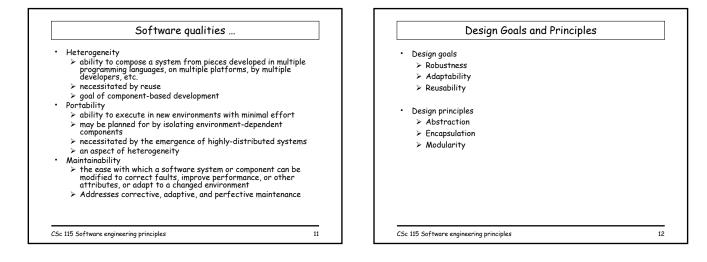
Adaptability	Laws of software evolution	
 The programs we are writing today might last for 30 years For programs to stay useful, they must adapt over time Different parts of the program evolve at different times and at a different pace > Database > User interface Programs are expected to run on many platforms > Windows > Mac > Unix/Linux > Solaris > Network-centric > Web-centric Laws of software evolution 	 First Law of Lehman [Leh80]: "Software which is used in a real-world environment must change or become less and less useful in that environment." Second Law of Lehman [Leh80]: "As an evolving program changes, its structure becomes more complex, unless active efforts are made to avoid this phenomenon." Third Law of Lehman [Leh80]: "Program evolution is self-regulating process. System attributes such as size, time between releases, and the number of reported errors are approximately invariant for each system release." 	
Sc 115 Software engineering principles 5	CSc 115 Software engineering principles	

Laws of software evolution	
 Fourth Law of Lehman [Leh80]: "Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development." Fifth Law of Lehman [Leh80]: 	
"Over the lifetime of a system, the incremental system change in each release is approximately constant."	
 What can we say about the complexity of the software systems developed over the past 40 years? Constant? 	.
> Increase?	
CSc 115 Software engineering principles 7	









Design principles: Abstraction

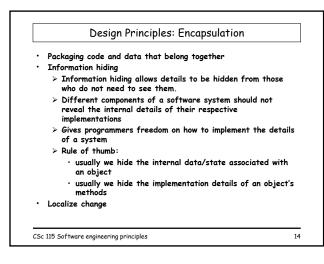
- To emphasize the important aspects and deemphasize immaterial aspects
- For example, a program is a string of bits, characters, tokens, syntax tree, classes, logical units, subsystems, application
- Levels of abstraction

 - Application
 Concepts, business rules, policies
 - ➤ Function Logical and functional specifications, non-functional requirements
 - > Structure
 - Data and control flow, dependency graphs
 - Structure and subsystem charts
 Software architectures
 Implementation

 - AST's, symbol tables, source text

CSc 115 Software engineering principles

13



Design principles: modularity

- An organizing structure in which different components of a software system are divided into separate functional units
- Separation of concerns
- The architecture of a house can be viewed as several interacting units
- \succ Electrical, heating and cooling, plumbing, structural subsystem The elements of susbsystems can be readily replaced if certain
 - standards are followed
 - Facilitates reuse and understanding
- Subsystems are organized into hierarchies of subsystems > Part-of hierarchies (i.e., packages, classes, fields, methods, statements, local variables)
 - > Is-a hierarchies (i.e., inheritance)

15