

§1. Nature of Software System software Services for other software Operating systems and middleware

- Embedded software
 - Resides within a larger system and implements specific functions
- Satellites, defense systems, energy systems
- Application software
- Stand-alone program that solves a specific need

Dmitry G. Korzun, 2013

Dmitry G. Korzun, 2013

Text editors, media players

Types of Software Systems

Software-embedded systems

- Hybrid combination of hardware, software, and people
- Hardware performs principal actions, software is supporting
- Vehicles, computer-controlled manufacturing machinery

Software-intensive systems

Network of computers and users (any information system)

Dmitry G. Korzun, 2013

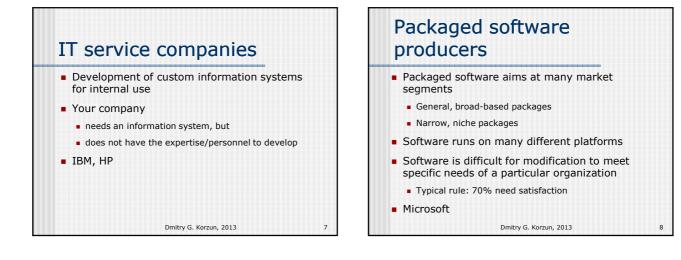
Dmitry G. Korzun, 2013

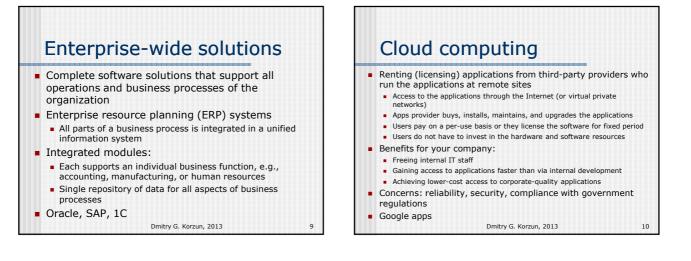
- Software on computers perform virtually all of the system functionality, in support of human operators
- WWW is an extreme case

Data-intensive computing systems

- Large-scale computing resources
- Supercomputers, parallel computing
- Weather analysis

Sources of Software Outsourcing Practice when one organization develops or runs a computer 1. Outsourcing application for another organization 2. IT service companies Extreme 1: a company develops and runs your application on its computers Packaged software producers You only supply input and take output 4. Vendors of enterprise-wide solution Payroll systems Extreme 2: a hired company runs your applications at your software site on your computers 5. Cloud computing Big business Cost-effective solution 6. Open source India, China, Latin America 7. In-House Nearshoring: no more than one time zone away





11

Open source

- Not just a final product but the source code itself
- Development by community, not by a particular company
- Money made with open source
 - Provision of maintenance and other services
 - One version is free, a featured version is commercial
- Examples:
 - Linux (~15% servers, ~2% desktops), mySQL, Firefox
 - SourceForge.net
 Dmitry G. Korzun, 2013

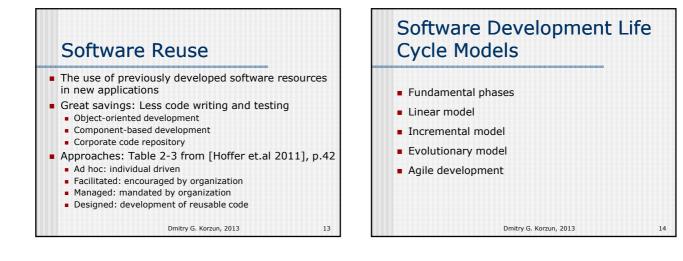
In-house

- Development of an information system from scratch for internal corporate use
- Now, a small fraction of efforts of internal IT departments

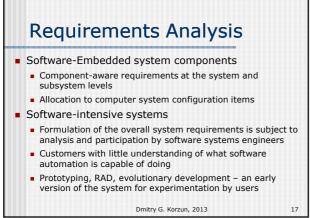
Dmitry G. Korzun, 2013

- Higher maintenance burden than other development options
- Hybrid solutions: in-house + purchased components

2







Requirements Generation Process

- Iterations of 4 critical steps:
 - Elicitation (language barrier between users and developers)
 - Analysis and Negotiation (necessity, consistency,
 - completeness, feasibility)Documentation
 - Validation (every requirement is consistent, coherent, unambiguous)
- Use cases (understanding of sequence of events and activities that need to be performed)
- Interface requirements (association of each input and output with requirements)

Dmitry G. Korzun, 2013

System Architecture

- Partition into relatively independent subsystems that may be designed, developed, produced, and tested as separate system building blocks
- Achievement of a high degree of "Modularity"
- Binding: loose and tight (grouping of closely related blocks)
- Coupling: tight and loose (interactions between blocks are minimized)

Dmitry G. Korzun, 2013

- Architecture modeling
 - Structured analysis and design
 - Object-oriented analysis and design

Structured Analysis and Design

- Functional flow block diagram (FFBD)
- Data flow diagram (DFD)
- Entity relation diagram (ERD)
- State transition diagram (STD)
- Data dictionary

19

21

Object-Oriented Analysis and Design

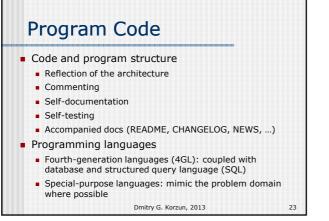
- Class encapsulates data and functions that operate on them
- Self-contained, robust, reusable program building blocks
- Arranging related classes into groups (subsystems or packages)
- Defining all of the relations/responsibilities within and among the groups

Dmitry G. Korzun, 2013

UML: unified modeling language



Dmitry G. Korzun, 2013

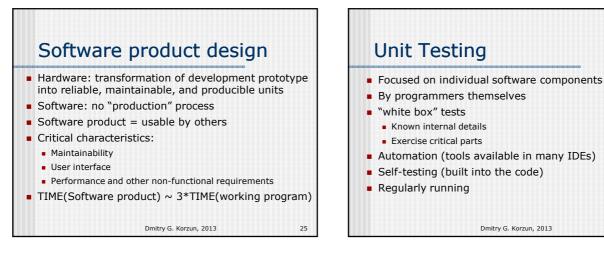


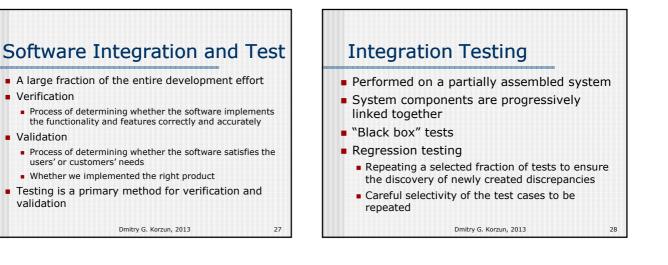


Verification

Validation

validation





System Testing Validation tests Alpha testing In a controlled environment at the developer's side Beta testing At a customer's side, without the developers Dmitry G. Korzun, 2013 29

Software Engineering Management

- CASE tools
- Requirements management tools
- Software metrics tools
- Integrated development support tools

Dmitry G. Korzun, 2013

30

Software Configuration Management (CM)

Difference between hardware and software

- Software abstractness and lack of well-defined components makes it difficult to understand
- Software has more interfaces; their penetration is deeper and hence is difficult to trace
- Any change may propagate deep into the system
- Any change may require retesting the whole system
- When a software system fails, it often breaks down abruptly
 The flexibility of software renders making a software change deceptively easy

Dmitry G. Korzun, 2013

31

Operation Description Description</

