Object-Oriented and Classical Software Engineering

EMERGING TECHNOLOGIES

Overview (contd)

- Aspect-oriented technology
- Model-driven technology
- Component-based technology
- Service-oriented technology
- Comparison of service-oriented and component-based technology
- Social computing
- Web engineering
- Cloud technology
- Web 3.0

Overview

- Computer security
- Model checking
- Present and future

18.1 Aspect-Oriented Technology

- A *concern* of a software product is a specific set of behaviors of that product
  - Example:
    - » In a banking product, concerns include
      - A set of interest computations
      - The writing of information to the audit trail

- A *core concern* of a software product is a primary set of behaviors of that product
  - Example:
    - » In the banking product,
      - The set of interest computations is a core concern
      - The writing of information to the audit trail is essential, but is not a core concern

Aspect-Oriented Technology (contd)

- Separation of concerns is highly desirable
  - But not always achievable in practice
  - Example:
    - » In the banking product,
      - The set of interest computations can probably be isolated to a few modules
      - But virtually every banking operation has to write to the audit trail

Aspect-Oriented Technology (contd)

- A cross-cutting concern cuts across module boundaries
  - Example:
    - » In the banking product,
      - The audit trail
- Cross-cutting can have a deleterious effect on maintenance
  - Cross-cutting can lead to regression faults

Aspect-Oriented Technology (contd)

- Cross-cutting violates separation of concerns

Figure 18.1(a)

- A change to the audit trail mechanism requires all six pieces of audit trail code to be consistently changed

Aspect-Oriented Technology (contd)

- Aim of aspect-oriented programming (AOP):
  - Isolate such cross-cutting concerns in special modules called *aspects*
- Aspects contain *advice*
  - Code to be linked to specific places in the software
  - Example:
    - » In a banking product, advice includes
      - An audit trail routine
- A *pointcut* is a place in the code where the advice is to be executed

Aspect-Oriented Technology (contd)

- An aspect therefore consists of two pieces:
  - The advice, and
  - Its associated set of pointcuts

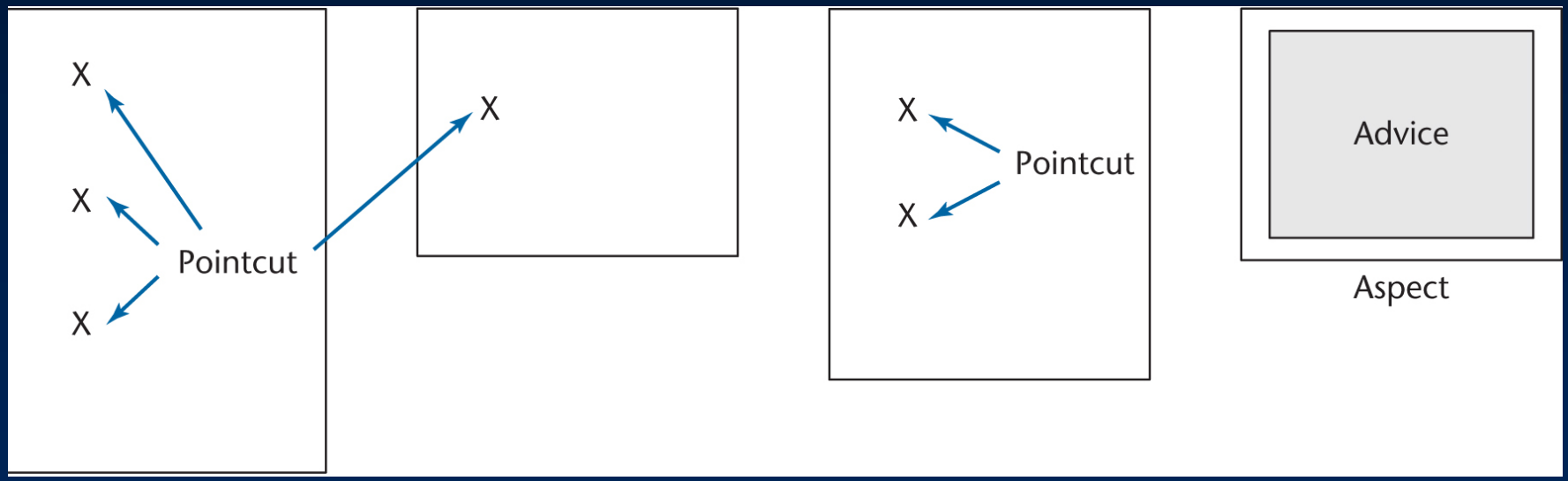

Figure 18.1(b)

- Now, a change to the audit trail mechanism is localized to the aspect

Aspect-Oriented Technology (contd)

- An aspect-oriented programming language is needed
  - Its compiler is called a *weaver*
- Development and maintenance are performed on the uncompiled source code, including its aspects and pointcuts
  - Separation of concerns is thereby achieved
- Now, a change to the audit trail mechanism is localized to the aspect

Aspect-Oriented Technology (contd)

- There are aspect-oriented extensions for many programming languages, including:
  - AspectJ (for Java)
    - » Currently the most popular AOPL

Aspect-Oriented Technology (contd)

- Aspect-oriented programming is one part of *aspect-oriented software development* (AOSD)
  - Aim: Early identification of both functional and nonfunctional cross-cutting concerns
- Once the cross-cutting concerns have been identified, they are
  - Specified (aspect-oriented analysis),
  - Modularized (aspect-oriented design), and
  - Coded (aspect-oriented implementation)

Aspect-Oriented Technology (contd)

- Aspect-oriented programming has been used in a number of commercial applications, including
  - IBM Websphere (a framework for building online information systems in Java), and
  - JBoss (an open-source Java application server)

18.2 Model-Driven Technology

- Problem: moving a software product to a new platform
- *Model-driven architecture* (MDA) solves the problem at the analysis level rather than at the design level

Model-Driven Technology (contd)

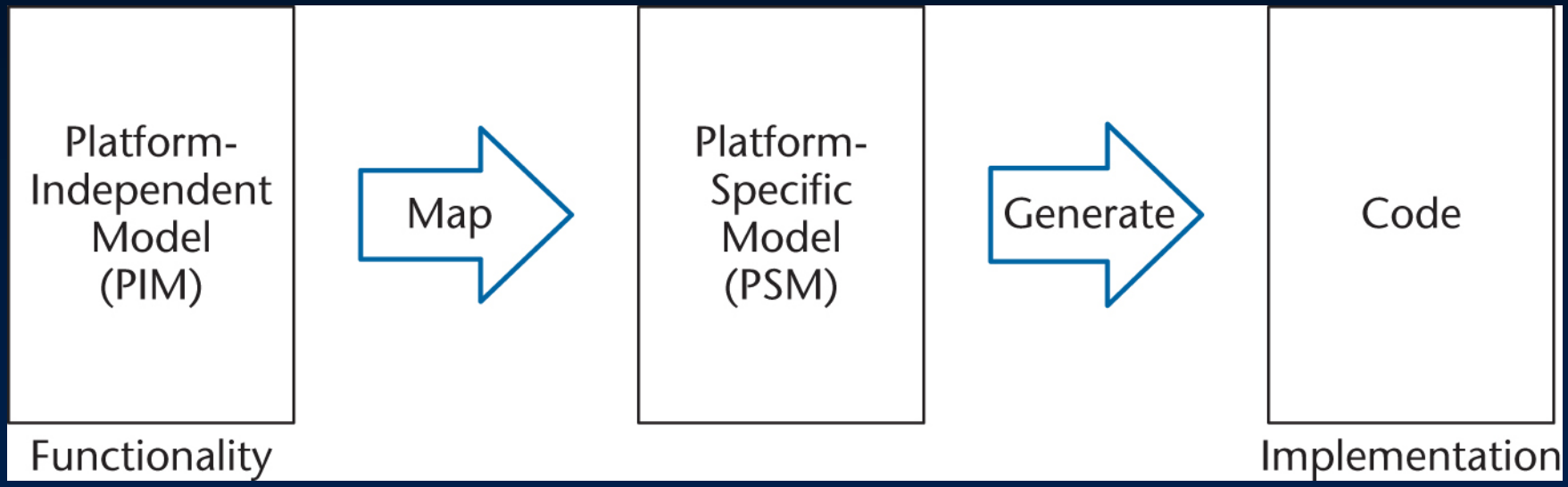

Figure 18.2

- 1. The functionality of the desired software product is specified by means of a platform-independent model (PIM)
  - This is done using UML, or an appropriate domain-specific language

Model-Driven Technology (contd)

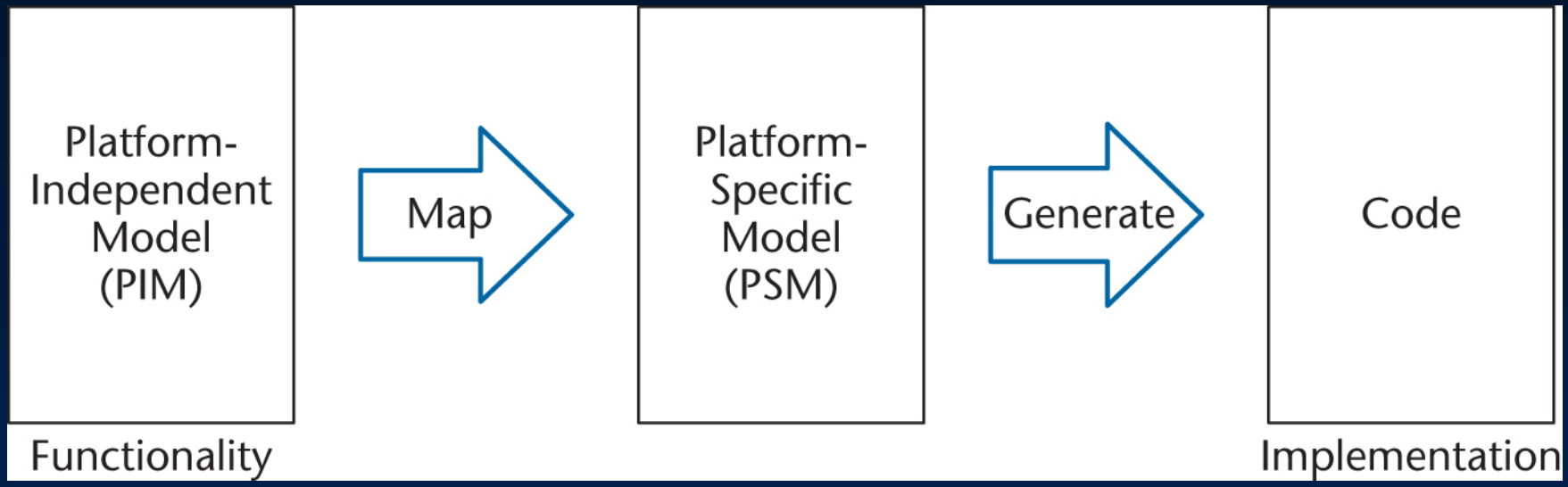

Figure 18.2 (again)

- 2. A platform-specific model (PSM) is chosen
  - Examples: CORBA, .NET, or J2EE
- The PIM is mapped into the selected PSM
  - The PSM is expressed in UML

Model-Driven Technology (contd)

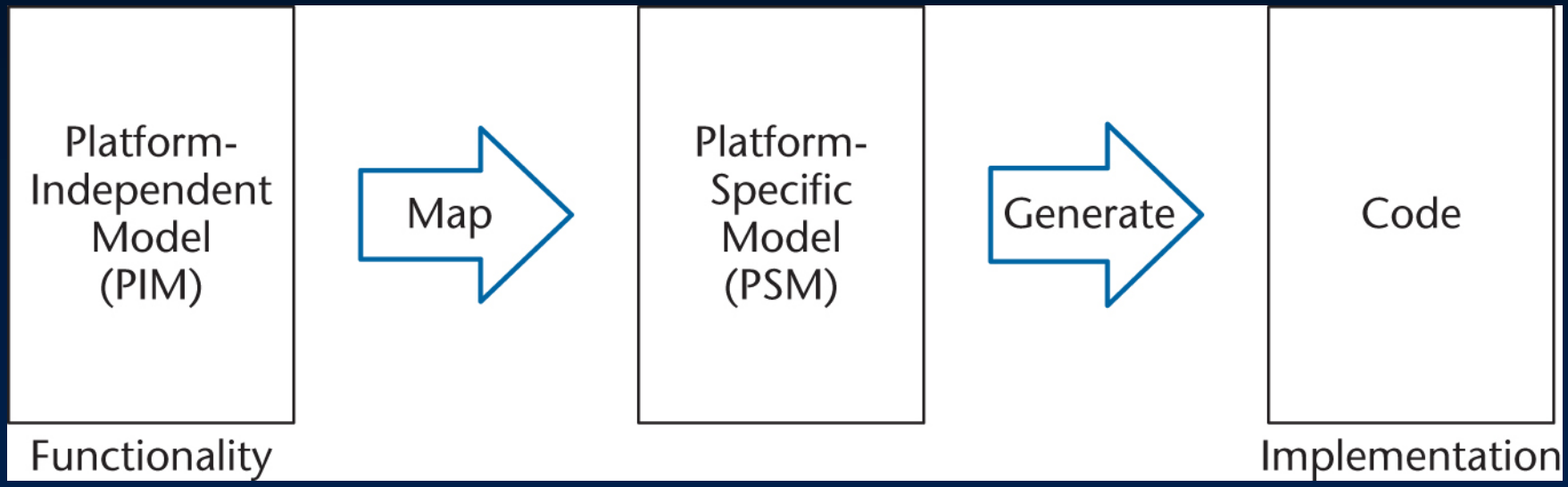

Figure 18.2 (again)

- 3. The PSM is translated into code, using an automatic code generator, and run on a computer

- 4. If multiple platforms are required, Steps 2 and 3 are repeated for each PSM

Model-Driven Technology (contd)

- MDA
  - Totally decouples functionality from implementation, and
  - Thereby provides a powerful mechanism for achieving portability
- Patterns play an important role in MDA-based software products
- MDA raises the level of abstraction from the platform-dependent code level to the platform-independent model level

18.3 Component-Based Technology

- The goal of *component-based technology*:
  - To construct a standard collection of reusable components
- Then, all software will be constructed by
  - Choosing a standard architecture,
  - Choosing standard reusable frameworks, and
  - Inserting standard reusable code artifacts into the hot spots of the frameworks

Component-Based Technology (contd)

- That is, all future software will then be built from those reusable components
  - Using an automated tool
- Result:
  - Product automation

Component-Based Technology (contd)

- For this technology to work, the components have to be
  - Independent, that is, fully encapsulated
  - At a higher level of abstraction than objects, because they cannot share state

Component-Based Technology (contd)

- Achieving component-based software engineering would lead to
  - Order-of-magnitude increases in
    - » software productivity and
    - » quality, and
  - Order-of-magnitude decreases in
    - » time to market and
    - » maintenance effort
- Unfortunately, the current state of the art is far from this ambitious target

18.4 Service-Oriented Technology

- With *service-oriented technology*, capabilities are provided
  - By service providers,
  - Over a network (frequently the Internet),
  - To meet specific needs of service consumers

Two Ways to Create a Document on a Computer

- 1. Install a copy of Microsoft Word on the user's computer, and then use Microsoft Word to create the document on that computer

- 2. Open a Web browser and create the document using Google Docs
  - The word processing software stays on the Google computer
  - The document also resides on the Google computer
    - » But a copy can be downloaded to the user's computer, for additional security

- This is service-oriented computing

- Both service-oriented technology and component-based technology:
  - Are instances of distributed computing
  - Are primarily reuse technologies
  - Require encapsulation
  - Are accessed through their interfaces
  - Must have the highest possible cohesion and the lowest possible coupling
  - Have low entry costs
  - Automatically download the latest version of the software
  - Are generally geographic location-independent

- One major difference between the two technologies is granularity
  - The basic building blocks of component-based technology are components, whereas
  - The basic building blocks of service-oriented technology are complete executable programs
- A second difference is that
  - Early versions of service-oriented technology are already widely used today, whereas
  - Component-based technology still requires breakthrough research before it could be used in practice

18.6 Social Computing

- The term *social computing* is used in two different contexts:
  - With the emphasis on the “social”
    - » Not an emerging technology
  - With the emphasis on the “computing”
    - » An emerging technology

Social Computing (contd)

- First, the term is in used in the context of the ways in which computers support social behavior
  - Examples include:
    - » Chat rooms
    - » Instant messaging
    - » E-mail
    - » Blogs
    - » Shared work spaces like wikis
- That is not an emerging technology

Social Computing (contd)

- Second, the term is used in the context of group computations
  - Examples include
    - » Online auctions
    - » Multiplayer online games
    - » Collaborative filtering
- This usage relates to an emerging technology

18.7 Web Engineering

- Analogous to software engineering, *Web engineering* is a discipline whose aim is the production of
  - Fault-free Web software
  - Delivered on time,
  - Within budget, and
  - Satisfying the user's needs

Web Engineering (contd)

- Web software is a subset of software in general
  - Accordingly, Web engineering is technically a subset of software engineering
- However, Web software has characteristics of its own
  - Web engineering should therefore be considered a separate discipline

Characteristics of Web Software

- Unstable requirements
- Wide range of user skills
- No opportunity to train users
- Varied content
- Exceedingly short maintenance turnaround times
- The human–user interface is of prime importance
- Diverse runtime environments
- Stringent privacy and security requirements
- Accessibility through multiple devices

Characteristics of Web Software (contd)

- Some researchers feel that Web technology is essentially different to computer technology
  - They have put forward a new discipline, Web science, analogous to computer science

18.8 Cloud Technology

- The Internet is sometimes referred to as “The Cloud”
- *Cloud technology* is a synonym for Internet-based technology
- Specific to cloud computing is the idea that the users are not expected to have any knowledge of the underlying infrastructure
  - The metaphor is that users are operating “in a cloud”

18.9 Web 3.0

- The World Wide Web (or Web for short) is a collection of hypertext documents
- Web 2.0 is a term that refers to the technology that individuals *now* use when they make use of the Web
- *Web 3.0* (or the Semantic Web) is an emerging technology
  - The term refers to ways that the Web will be used in the future

18.10 Computer Security

- *Computer security* is not a branch of software engineering
  - Nevertheless, there are aspects of computer security that are also of concern to software engineers
- One important area of overlap between software engineering and computer security is human factors

Human Factors

- Claim:
  - “Given a choice between dancing pigs and security, users will pick dancing pigs every time”
  - Dancing pigs problem
- The claim is supported by the “cute” swimming bear on a fraudulent web page for Bank of the West

Human Factors (contd)

- The design of human interfaces has to take into account that many users simply do not care about security
- Security has to be built into a software product, rather than offered as an option
  - This is a hard problem

18.11 Model Checking

- Model checking is a testing technology for hardware that is starting to be applied to software
- Correctness proving is still somewhat problematic
  - We need an alternative to a human having to construct a proof

Model Checking (contd)

- Model checking
  - Use temporal logic to specify a software product that is intended to run without stopping
  - Realize the temporal logic specification as a finite state machine
  - Then determine the properties of that finite state machine
- In this way, we can mathematically show that a software product is correct without explicitly constructing a proof of correctness

18.12 Present and Future

- This chapter contains an outline of 10 emerging technologies
  - All are promising
  - All have the potential to become mainstream technologies

Present and Future (contd)

- “It’s tough making predictions, especially about the future” – Yogi Berra
  - So, only in the future will we know what the future will bring