Introduction to Software Reuse

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Software Reuse

- The use of existing software or software knowledge to build new software
- In the last 20 years, several reuse techniques have been proposed
  - Libraries, objects, components, and so on
- Open source initiatives have created a large amount of source code available

Granularity of Reuse

- Objects and Functions
  - Most common type of reuse
  - It has been practiced for 40 years
- Components
  - Middle-granularity reuse. Examples are architectural components and subsystems
- Systems
  - A system can be packed for reuse and, for instance, included into a larger system
  - It usually requires customization

Advantages of Software Reuse

- Accelerated development and lower costs
  - The system may be delivered in shorter time and at lower price
- Effective use of specialists
  - A way of use knowledge from experts
- Increase the product dependability
  - Software was used and tested before
- Standards compliance
  - For instance, interface have similar look and feel

Potential Drawbacks

- Creating and maintaining a library
  - You need to find the right software and understand how it works, before reusing
- Increased maintenance costs
  - Reused components may become incompatible with others in future versions
- Lack of tool support
  - Development environments are not fully prepared to support software reuse

Types of Reuse by Motivation

- Opportunistic Reuse
  - While getting ready to begin a project, the team realizes that there are existing components they can reuse
- Planned Reuse
  - A team strategically designs components so that they will be reusable in future

Planning for Reuse

- Effective reuse requires planning
  - Managers have to be engaged and motivate the whole organization
- Companies focused on a specific domain have advantages
  - It is easier to find reuse opportunities in this case

Key Planning Factors

- Several factors have to be considered in planning for reuse
  - Development schedule
  - Expected software lifecycle
  - Skills and experience of the team
  - The application domain

more in the textbook +

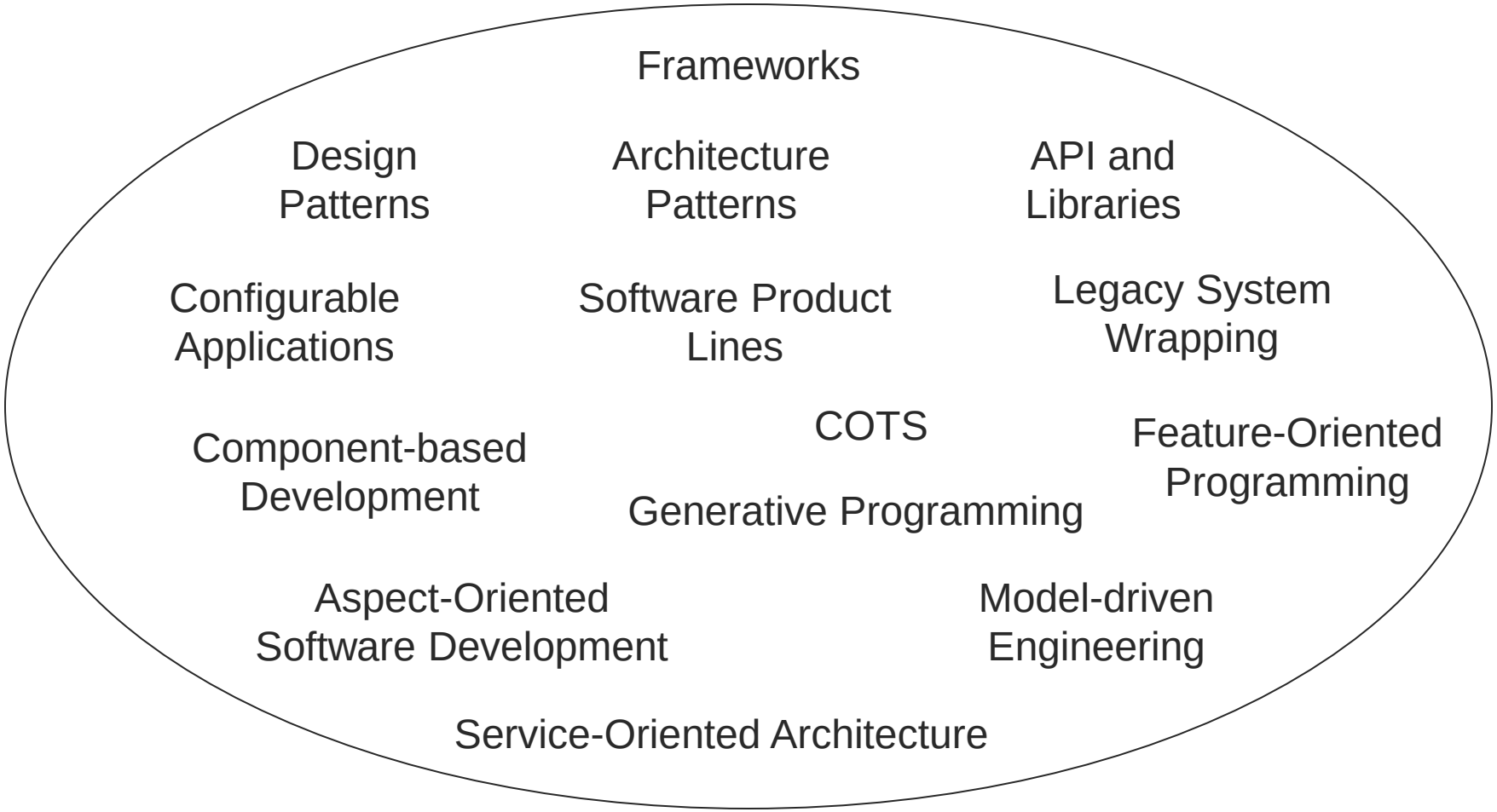
Schedule and Lifecycle

- Development schedule
  - In a tight schedule scenario, reuse may speed up software development
  - The learning curve might be a bottleneck
- Software lifecycle
  - Software reuse might be challenge in highly changed code
  - Third-part components may impose maintenance difficulties

Team and Application Domain

- Expertise and knowledge of the team
  - Some reuse techniques might be complex to be understood and applied
  - Developers have to be familiar with the reuse techniques
- Application Domain
  - In some domains, it is easy to find reusable components and libraries
  - Other domains are not so easy

The Reuse Landscape



Reuse Techniques (1 of 3)

- Library and API
  - Classes and functions that implement common reusable abstractions
- Design and Architecture Patterns
  - Patterns are reusable general solutions for recurring problems
- Application Framework
  - Collection of classes that implements the standard structure of an application

Reuse Techniques (2 of 3)

- Components
  - Collection of objects that can be integrated to create a new system
- Legacy System Wrapping
  - Cheap option that wraps a legacy system and defines new interfaces
- Service-Oriented Architecture
  - New systems are developed by sharing common services

Reuse Techniques (3 of 3)

- Aspect Oriented Software Development
  - Reuse technique to support advanced separation of concerns (modularity)
- Software Product Lines and Configurable Systems
  - Family of applications following a common architecture
- Model Driven Engineering
  - Code is generated by means of refining domain and application models

Bibliography

- Ian Sommerville. **Software Engineering**, 10th Edition. Pearson Education, 2016.
  - Chap. 15 Software Reuse (up to Section 15.1)