Faculty of Computers and Information
Joint Master In Software Engineering Program (JMSE)

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

Based on slides by © Tripathy & Naik

Software Evolution : TOC

1. Introduction to Software Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Reuse and Domain Engineering
5. Program Comprehension
6. Impact Analysis
7. Refactoring
8. Reengineering
9. Legacy Information Systems

General Idea of Reuse

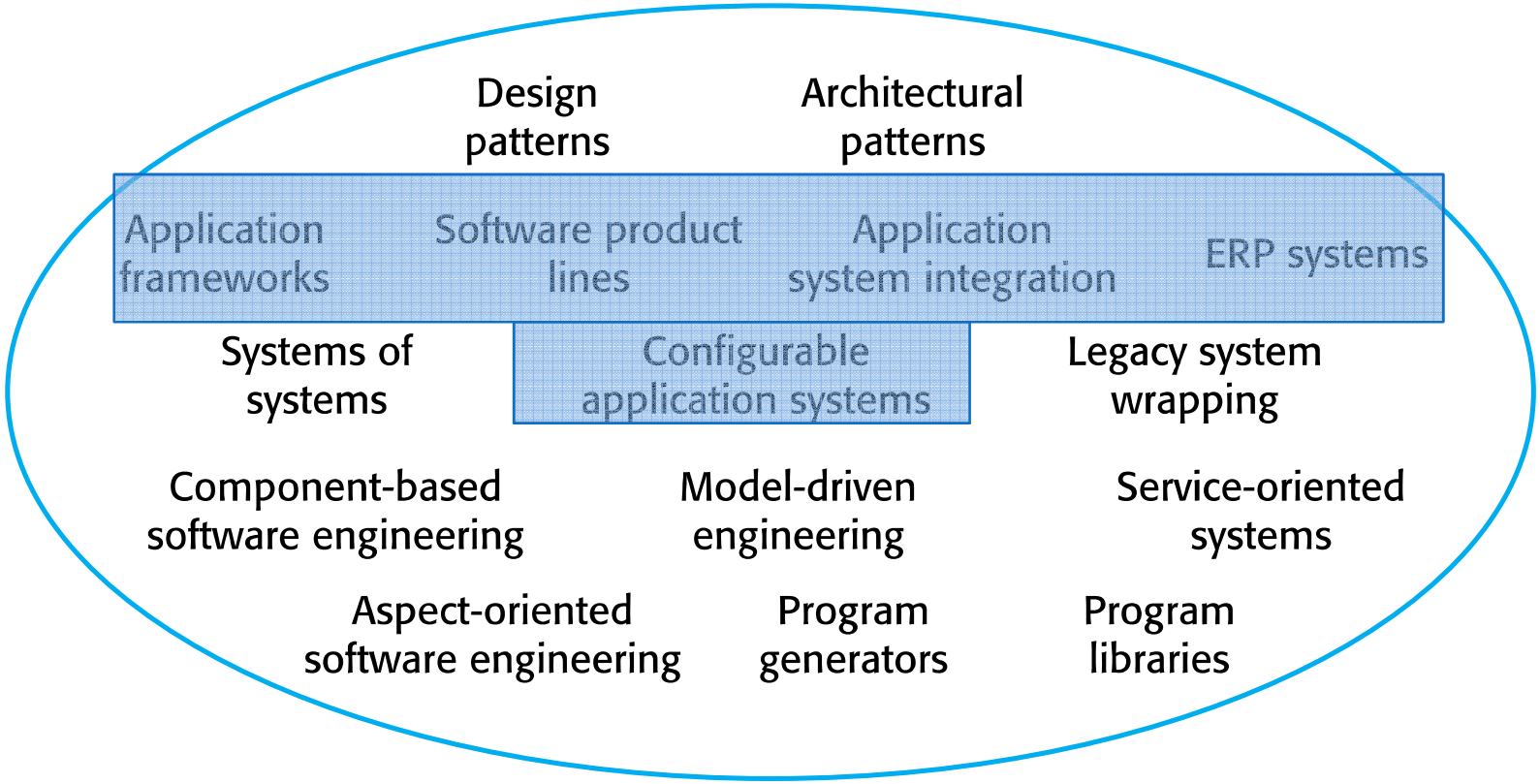
- Software reuse involves two main activities:
  1. Software development with reuse
  2. Software development for reuse.

- "development-for-reuse" process is used to create reusable software assets (RSA)

What Is Typically Reused?

- System reuse
  - Complete systems, which may include several application programs may be reused.
- Application reuse
  - An application may be reused either by incorporating it without change into other or by developing application families.
- Component reuse
  - Components of an application from sub-systems to single objects may be reused.
- Object and function reuse
  - Small-scale software components that implement a single well-defined object or function may be reused.

The Software Reuse Landscape



General Idea of Reuse

- M. D. McIlroy in 1969 proposed *Mass Produced Software Components* as means for the industrialization of the production of application software from off-the-shelf components.
- Early Concepts of reuse are:
  - Program families introduced by Parnas
    - Program families are a set of programs with several common attributes and features
  - Domain and domain analysis introduced by Neighbors
    - Domain analysis means finding objects and operations of a set of similar software systems in a specific problem

Reusability

- The **reusability** property of a software asset indicates the degree to which the asset can be reused in another project.
- For a software component to be reusable, it needs to exhibit the following properties that directly encourage its use in similar situations

1. Environmental independence: makes minimal interactions with other components to perform a task
2. High cohesion: achieves a single objective
3. Low coupling: has few or no dependencies
4. Adaptability: easily changed to run in a new environment.
5. Understandability: easily comprehended
6. Reliability: consistently perform its intended function without degradation or failure
7. Portability: usability of the same software in different environment

Reuse Models

- The organization can select one or more reuse models that best meet their business objectives, engineering realities, management styles, and additional capital investment.
- Reuse models are classified as:
  - **Proactive**: the system is designed and implemented for all conceivable variations; this includes design of reusable assets. It is an investment risk if the future product requirements are not aligned with the projected requirements
  - **Reactive**. while developing products, reusable assets are developed if a reuse opportunity arises. However, in the absence of a common, solid product architecture in a domain, continuous reengineering of products can render this approach more expensive.
  - **Extractive.** To make a domain engineering's initial baseline, an extractive approach reuses some operational software products. Therefore, this approach applies to organizations that have accumulated both artifacts and experiences in a domain, and want to rapidly move from traditional to domain engineering.

Factors Influencing Reuse

- Frakes and Gandel identified four major factors for systematic software reuse: managerial, legal, economic and technical.

1. Management support:
   - Reuse may need years of investment before it pays off.
   - Reuse involves changes in organization funding and management structure that can only be implemented with executive management support.
2. Legal:
   - This factor is linked with cultural, and social factors, and it presents very difficult problems.
   - Potential problems include proprietary and copyright issues, and responsibilities of reusable software, and contractual requirements involving reuse (especially reuse of third-party software).

Factors Influencing Reuse

3. Economic:
   - Software reuse will succeed only if it provides economic benefits.
4. Technical
   - Much attention from the researchers actively engaged in library development, object-oriented development paradigm, and domain engineering.
   - A reuse library stores reusable assets and provides an interface to search the repository.
   - library assets can be collected by:
     1. reengineer the existing system components
     2. design and build new assets
     3. purchase assets from other sources.

Success Factors of Reuse

1. Ensure that management understands reuse issues at technical and nontechnical levels.
2. Support reuse by means of tools and methods.
3. Support reuse by placing reuse advocates in senior management.
4. Develop software with the product line approach.
5. Develop generic software architectures for product lines.
6. Perform domain modelling of reusable components.
7. Practice reusing requirements and design in addition to reusing code.
8. Follow a software reuse methodology and measurement process.
9. Develop software architectures to standardize data formats and product interfaces.
10. Incorporate off-the-shelf components.

Development-for-Reuse - Domain Engineering

- Domain Analysis refers to the set of activates that support defining multiple related products from the very beginning of the software development process.
- **Software Product Lines** (also called **Product families**) apply the concept of *product lines* defined in manufacturing to the software development process. It moves the software development from a product-based development to a product line-based development
- Examples:
  - HP has a product line for their Printers (drivers software)
  - A product Line for Content Management Systems
  - Philips has a product line for their MRI scanners
  - MS has a product line for their SQL server