Architectural Design

V. Paúl Pauca

Department of Computer Science
Wake Forest University

CSC 331-631
Fall, 2013

Architectural Design

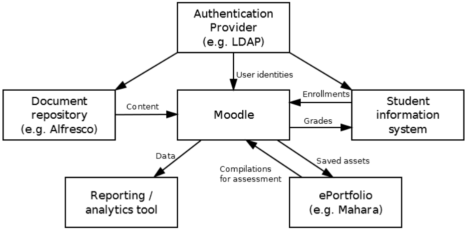
Creative process concern with:

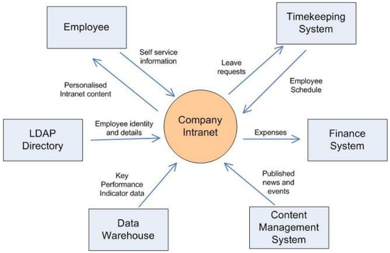
- How a system should be organized
- Designing overall structure of the system, before implementation

Importance

- Key step between requirements and implementation
- Non-functional requirements depend on it
- Conceptual view – facilitate discussion with stakeholders and developers
- Huge impact on performance, robustness, maintainability, etc.

Some Basic Examples


Authentication Provider (e.g. LDAP)
User identities
Document repository (e.g. Alfresco)
Content
Moodle
Enrollments
Grades
Student information system
Data
Compilations for assessment
Saved assets
Reporting / analytics tool
ePortfolio (e.g. Mahara)



- Block diagrams are typically used for high-level conceptual representation – interaction with stakeholders
- Interaction with developers requires different detailed models

Key Design Decisions

System architect's job:

- Is there a *generic* application architecture that can be used?
- What *architectural patterns* might be used? pros and cons?
- How will the structural components be decomposed?
- What architecture fits best the non-functional requirements?
- How should the architecture be evaluated or documented?

Practical questions

- If using agile methodology, when should system architecture be tackled?
- What types of diagram tools (formal vs. informal) should be used?
- How much detail to include in the diagrams?

Experienced system architects know:

- What architecture/pattern to use for specific requirements: performance, security, safety, maintainability, etc.
- Pros and cons of choosing an architecture over another
- What views/perspectives of the system are useful: logical view, process view, development view, physical view (See Krutchen's 4+1 view model in previous slides)

Basic Architectural Patterns – Sommerville

- Model-view controller
  Separates data (model) from how it is displayed (view)
- Layered architecture
  Separates functionality into layers: from core services to higher-level services
- Repository architecture
  Data managed through a repository, components interact indirectly
- Client-server architecture
  Functionality organized into services delivered from separate servers
- Pipe and filter architecture
  Data flows from one processing component to another

Application Architectures – Sommerville

Sommerville describes architectures for business applications separately

- Transaction processing applications
  Data centered, banking systems, e-commerce, etc.
- Language processing systems
  Interpreters, XML data representation/translation, compilers