

# SS ZG653 (RL1.2): Software Architecture

## A Brief History of Software Architecture

**BITS** Pilani

Pilani|Dubai|Goa|Hyderabad

Instructor: Prof. Santonu Sarkar

# Informally what is meant by (Software) Architecture

---

- Essentially a blueprint of a software system that helps **stakeholders** to understand how the system would be once it is implemented
- What's should be there in this blueprint?
  - A description at a higher level of abstraction than objects and lines of codes

So that

- Stakeholders understand and reason about without getting lost into a sea of details

# Who are Stakeholders?

A complex software has multiple stakeholders who expect certain features of the software

Stakeholder	Area of Concern
Chief Technologist	<ul style="list-style-type: none"> <li>• Does it adhere to organization standards ?</li> </ul>
Database Designer	<ul style="list-style-type: none"> <li>• What information to be stored, where, how, access mechanism???</li> <li>• Information security issues?</li> </ul>
Application Development team	<ul style="list-style-type: none"> <li>• How do I implement a complex scenario?</li> <li>• How should I organize my code?</li> <li>• How do I plan for division of work?</li> </ul>
Users/Customers	<ul style="list-style-type: none"> <li>• Does it perform as per my requirement?</li> <li>• What about the cost/budget?</li> <li>• Scalability, performance and reliability of the system?</li> <li>• How easy it is to use?</li> <li>• Is it always available?</li> </ul>
Infrastructure Manager	<ul style="list-style-type: none"> <li>• Performance and scalability</li> <li>• Idea of system &amp; network usage</li> <li>• Indication of hardware and software cost, scalability, deployment location</li> <li>• Safety and security consideration</li> <li>• Is it fault tolerant-crash recovery &amp; backup</li> </ul>
Release & Configuration Manager	<ul style="list-style-type: none"> <li>• Build strategy</li> <li>• Code management, version control, code organization</li> </ul>
System Maintainer	<ul style="list-style-type: none"> <li>• How do I replace of a subsystem with minimal impact ?</li> <li>• How fast can I diagnosis of faults and failures and how quickly I can recover?</li> </ul>

# Why Architecture needs to be described?

---

## Any Large Software Corporation

- ❑ Hundreds of concurrent projects being executed
  - 10-100 team size
- ❑ Projects capture requirements, there are architects, and large Development teams
- ❑ Architect start with requirements team & handover to Development teams

- Each stakeholder has his own interpretation of the systems
  - Sometimes no understanding at all
  - Architect is the middleman who coordinates with these stakeholders
- How will everyone be convinced that his expectations from the system will be satisfied?
- Even when the architect has created the solution blueprint, how does she handover the solution to the developers?
- How do the developers build and ensure critical aspects of the system?
- Misunderstanding leads to incorrect implementation
  - Leads to 10 times more effort to fix at a later stage

# Software Architecture Definition

---

- No unique definition though similar...
  - (look at <http://www.sei.cmu.edu/architecture/start/glossary/classicdefs.cfm> )
- .. “**structure** or structures of the system, which comprise **software elements**, the **externally visible properties** of those elements, and the **relationships** among them”

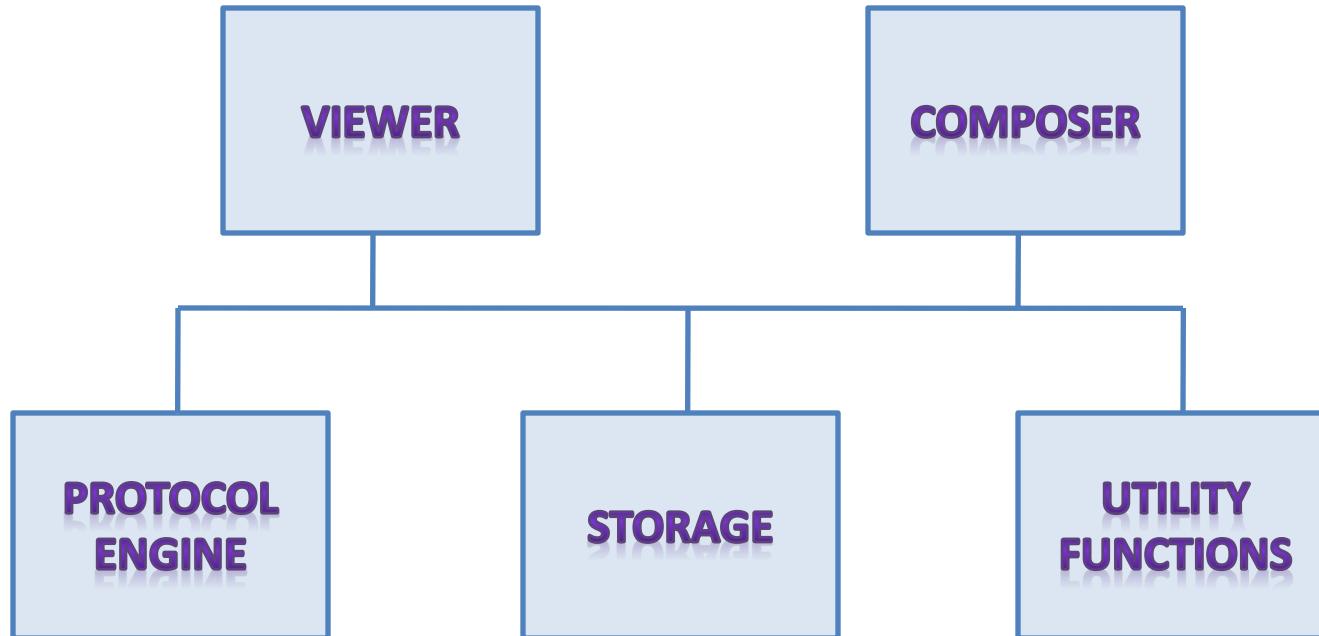
(Bass, Clements and Kazman, Software Architecture in Practice, 2<sup>nd</sup> edition)
- “description of elements from which systems are built, **interactions** among those elements, **patterns** that guide their **composition**, and **constraints** on these patterns. In general, a particular system is defined in terms of a collection of **components** and interactions among these components”

Shaw and Garlan “Software Architecture: Perspectives on an Emerging Disciplines”
- “description of the **subsystems** and **components** of a software system and the **relationship** between them. Subsystems and components are typically specified in different **views** to show the relevant **functional** and **nonfunctional** properties of a software system”

F. Buschmann et al, Pattern Oriented Software Architecture

# Is this Architecture

---



What we understand

- The system has 5 elements
- They are interconnected
- One is on the top of another

Typically we describe architecture as a collection of diagrams like this

# What's Ambiguous?

---

- Visible responsibilities
    - What do they do?
    - How does their function relate to the system
    - How have these elements been derived, is there any overlap?
  - Are these processes, or programs
    - How do they interact when the software executes
    - Are they distributed?
  - How are they deployed on a hardware
  - What information does the system process?
-

# What's Ambiguous?

---

- Significance of connections
  - Signify control or data, invoke each other, synchronization
  - Mechanism of communications
- Significance of layout
  - Does level shown signify anything
  - Was the type of drawing due to space constraint

# What should Architecture description have?

---

- A structure describing
    - Modules
      - Services offered by each module
      - and their interactions- to achieve the functionality
    - Information/data modeling
    - Achieving quality attributes
    - Processes and tasks that execute the software
    - Deployment onto hardware
    - Development plan
-

# What should Architecture description have?.....

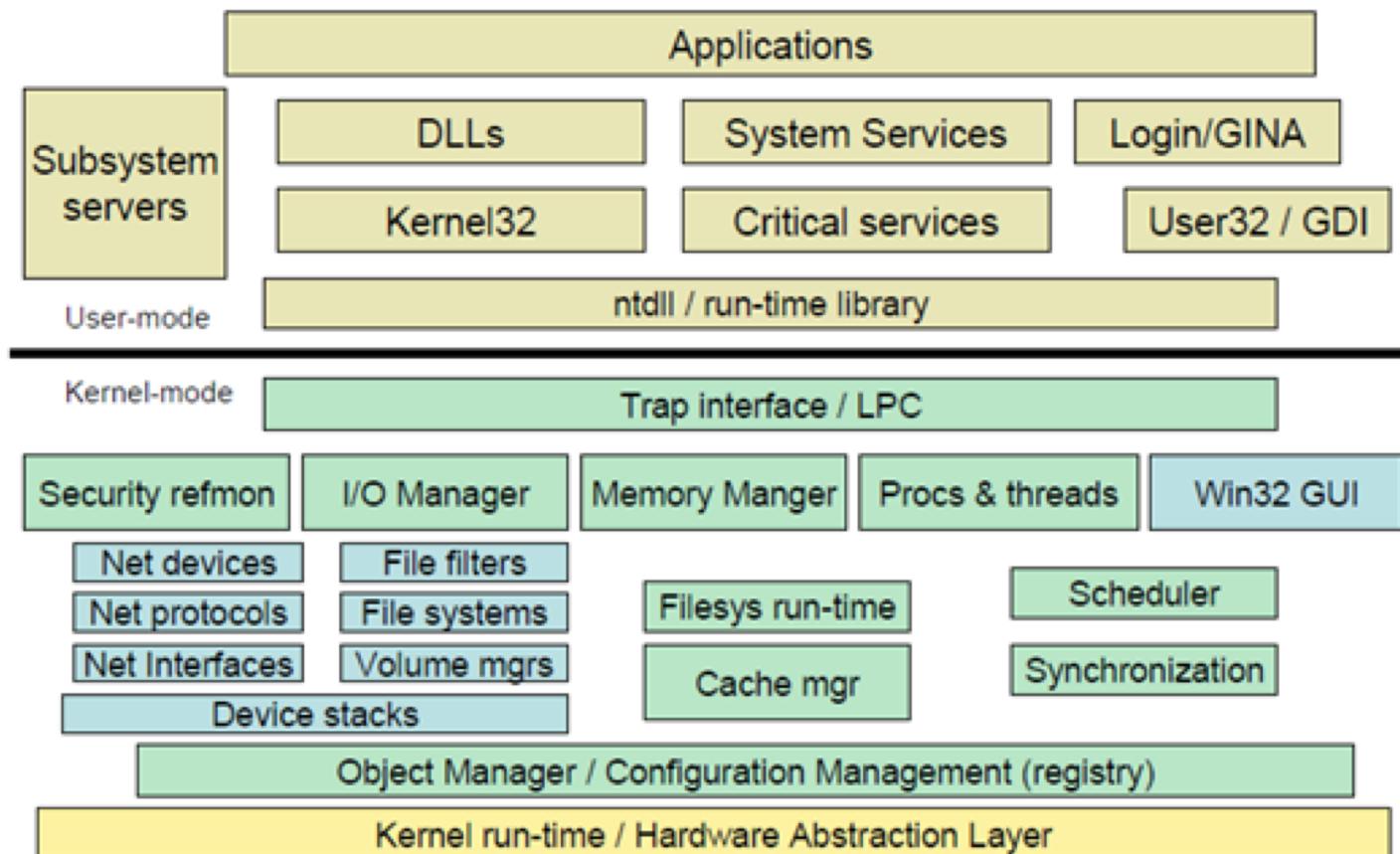
---

- A behavioral description
  - describing how the structural elements execute “important” and “critical” scenarios
    - E.g. how does the system authenticates a mobile user
    - How does the system processes 1 TB of data in a day
    - How does it stream video uninterrupted during peak load
  - These scenarios are mainly to implement various quality attributes

# Architecture of Windows

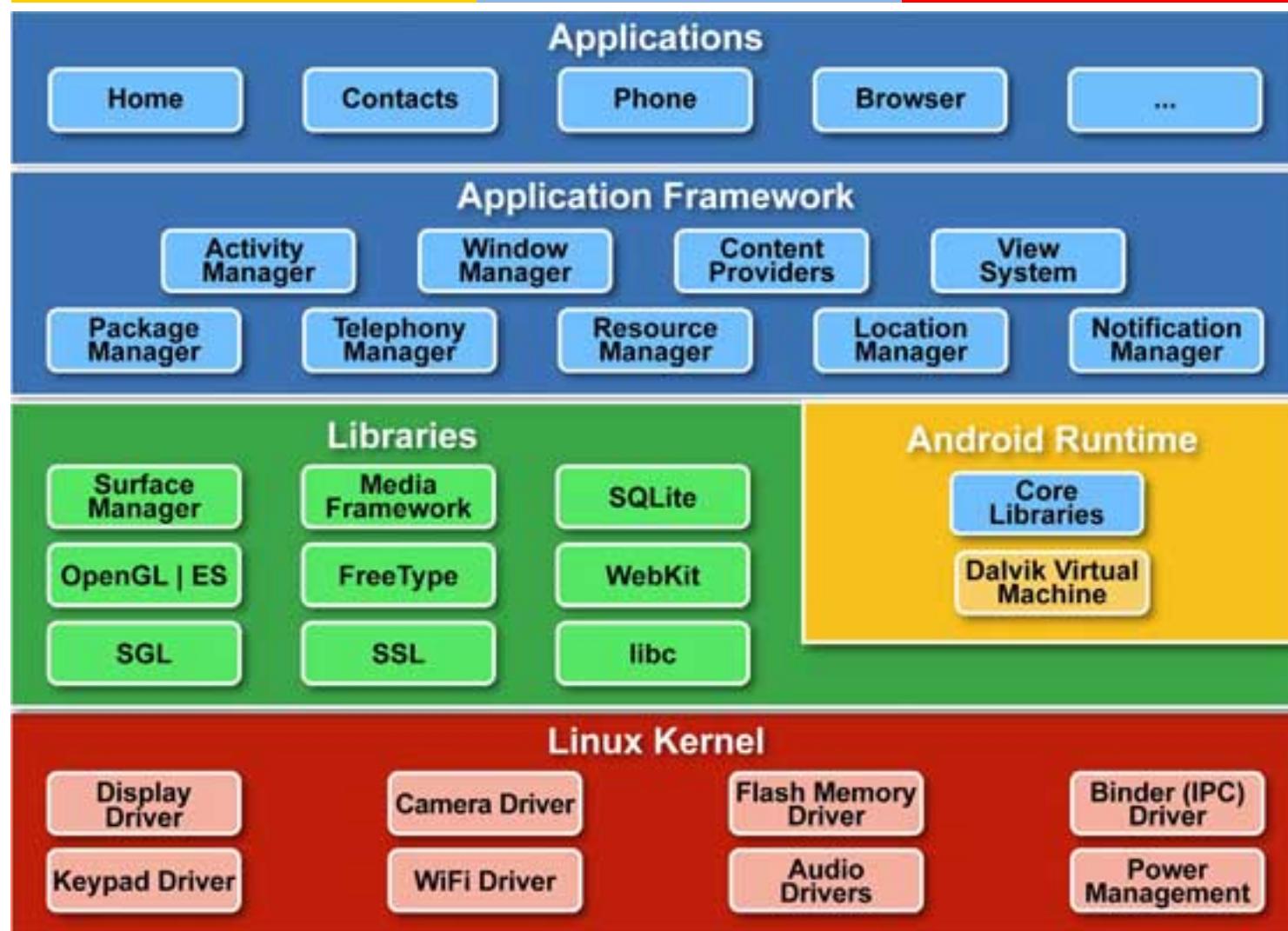
<https://http://blogs.msdn.com/b/hanybarakat/archive/2007/02/25/deeper-into-windows-architecture.aspx>

## Windows Architecture



# Architecture of Android

[http://www.techotopia.com/index.php/An\\_Overview\\_of\\_the\\_Android\\_Architecture](http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture)





# SS ZG653 (RL 1.3): Software Architecture

## Architecture Styles and Views

Instructor: Prof. Santonu Sarkar



**BITS** Pilani

Pilani|Dubai|Goa|Hyderabad

# Architecture Styles

---

- Architecture style first proposed by Shaw and Garlan—synonymous to “architecture pattern”
  - A set of element types (what the element does- data store, compute linear regression function)
  - A set of interaction types (function call, publish-subscribe)
  - Topology indicating interactions and interaction types
  - Constraints
  - Also known as architectural pattern
- We shall cover some of these patterns in details

# Views and Architectural Structure

---

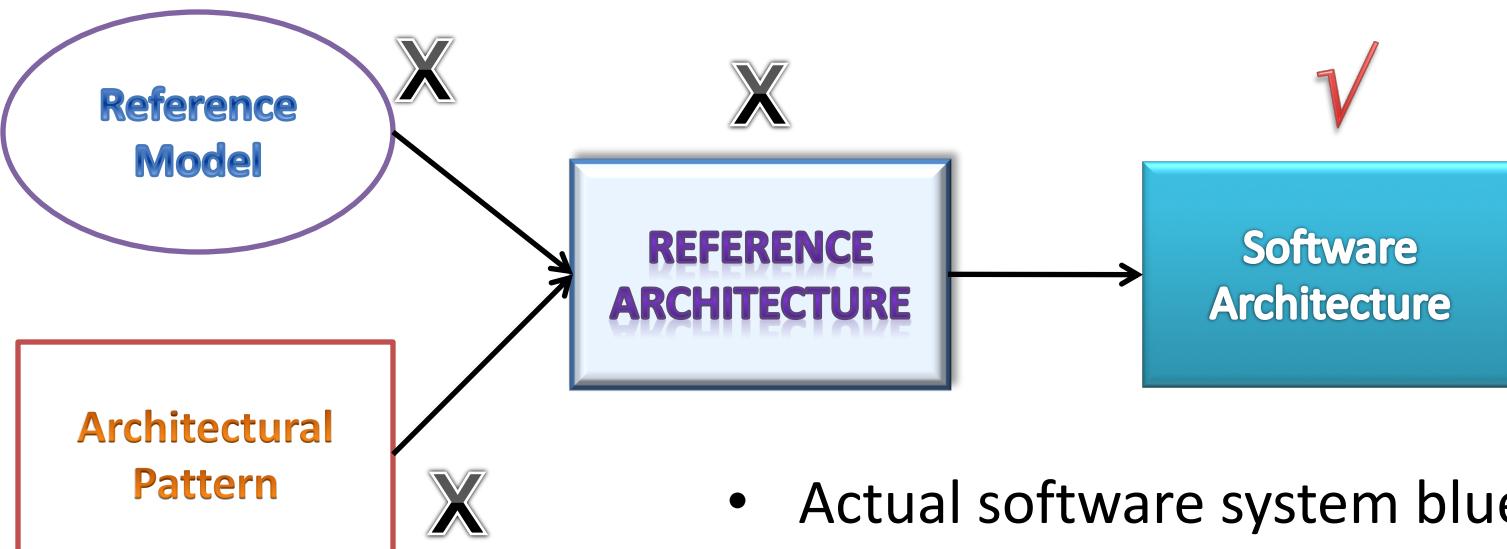
- Since architecture serves as a vehicle for communication among stakeholders
  - And each stakeholder is interested about different aspects of the system
  - It is too complex to describe, understand and analyze the architecture using one common vocabulary for all stakeholders
    - Essentially it needs to be described in a multi-dimensional manner
- View based approach
  - Each view represents certain architectural aspects of the system, created for a stakeholder
  - All the views combined together form the consistent whole
- A Structure is the underlying part of a view- essentially the set of elements, and their properties
  - A view corresponding to a structure is created by using these elements and their inter-relationships

# Reference Model and Reference Architecture

---

- A reference model
    - Decomposes the functionality into a set of smaller units
    - How they interact and share data
    - These units co-operatively implement the total functionality
  - A reference architecture
    - Derived from the reference model
    - Concrete software elements, mapped to the units of the reference model, that implement the functionality
-

# Inter-relationships



- Not architecture by itself!!
- Actual software system blueprint derived from requirement
- Contains design decisions
- Describes how it is deployed
- Addresses Quality of Service concerns

# Benefits of Software Architecture

---

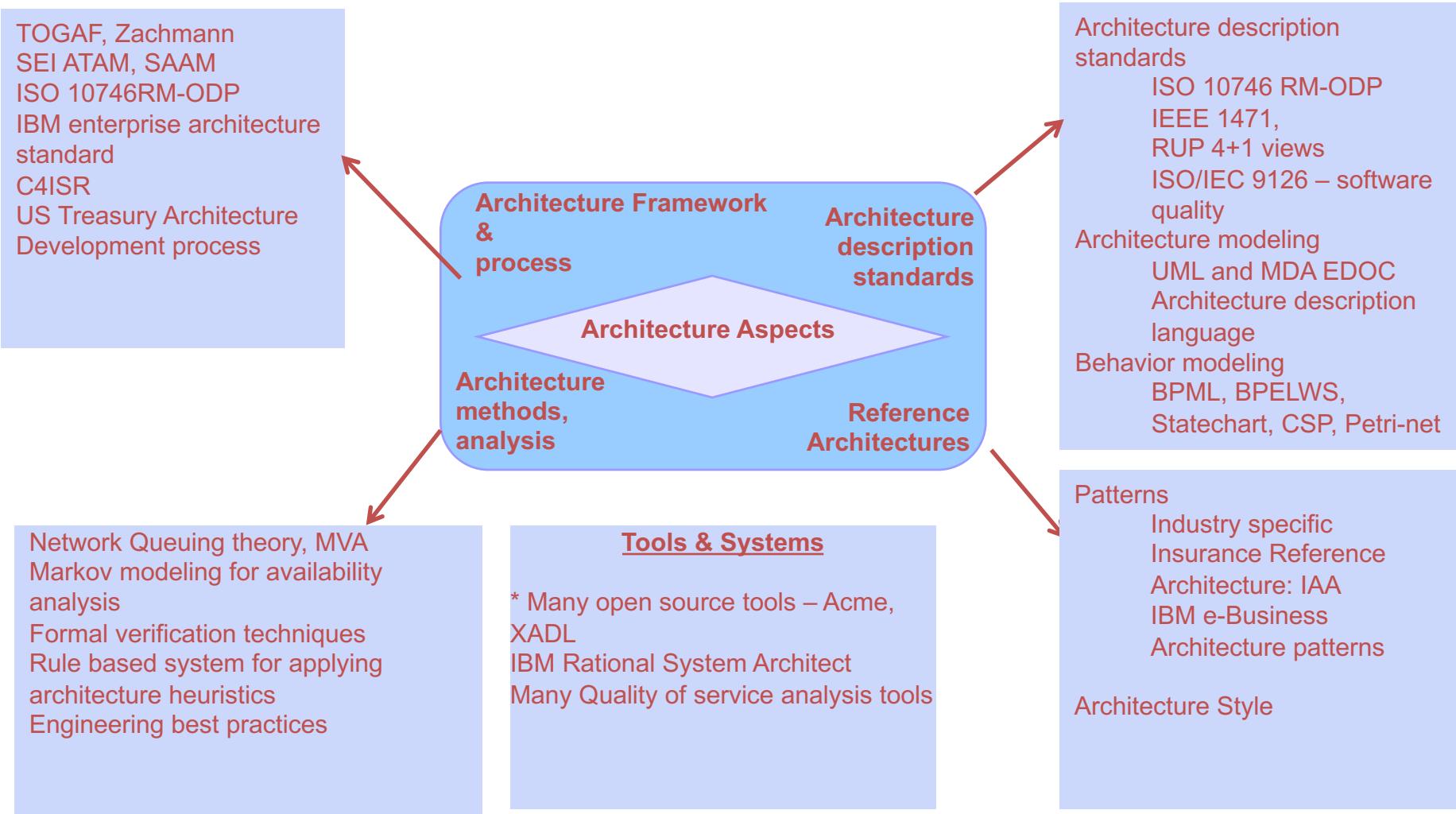
1. Every stakeholder should understand “unambiguously” what the blueprint is
    - Standard approach, vocabulary, output
    - Common language for communication
  2. Streamlining work assignments for multiple teams
    - Avoiding information loss, enforcing traceability
  3. Design decisions are made early
    - Quicker to evaluate these decisions and correct it rather than discovering it later (10 – 100 times more costly)
    - Early analysis of QoS and evaluation of architecture
    - Early analysis of meeting quality requirements and compromise between different QoS requirements
    - Early prototyping of important aspects quickly
    - More accurate cost and schedule estimation
  4. Improve speed of development
    - Reuse
      - Helps in building a large product line faster by sharing common architecture
      - From one implementation to another similar implementation
    - Based on the architecture, one can quickly decide build-vs –use external components
    - Tool that can automate part of development, testing
-

# Three Structures will be covered

---

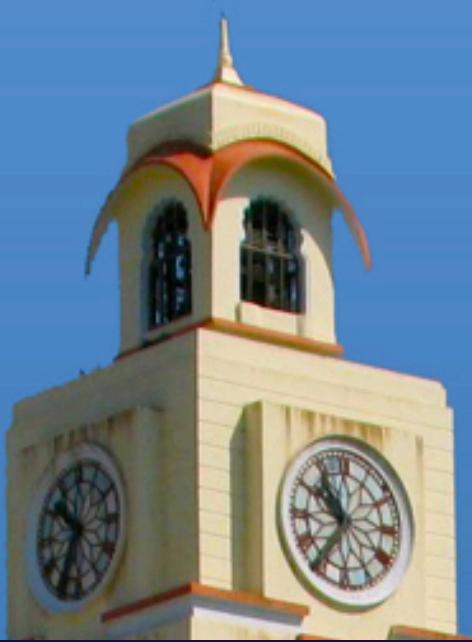
- Module Structure
  - How is the system to be structured as a set of code units (modules)?
- Component-and-connector structures
  - How is the system to be structured as a set of elements that have runtime behavior (components) and interactions (connectors)
  - What are major executing components and how do they interact
- Allocation structures
  - How is the system to relate to non-software structures in its environment (CPU or cluster of CPUs, File Systems, Networks, Development Teams ...)

# In Bits and Pieces (Unfortunately)



---

# Thank You



# SS ZG653: Software Architecture

## Introduction to the Course

**BITS Pilani**

Pilani|Dubai|Goa|Hyderabad

**Instructor: Prof. Santonu Sarkar**

# About the Instructor

---

- Professor at Dept of CSIS, BITS Pilani K.K.Birla Goa Campus
  - Bachelors, Masters, PhD in Computer Science
  - Nearly 20 years of experience in Applied Research, Product and Application Development, Consulting and Teaching
  - Broad Area: Software Engineering
    - Software Architecture, Software Design, Software Modularity, Reengineering
    - Software Development Challenges in Cloud and Heterogenous Computing Platform
-

# About the Course

---

- To study software architecture (we will simply call architecture in this context)
    - What is the abstraction of the software, and how to create, and how to represent
    - What are the relationships between various entities
    - How architectural principles are used during software system analysis and design.
  - To study about the role of architecture patterns in software design
  - To study about the applicability of design patterns in software design
-

# Course Objective

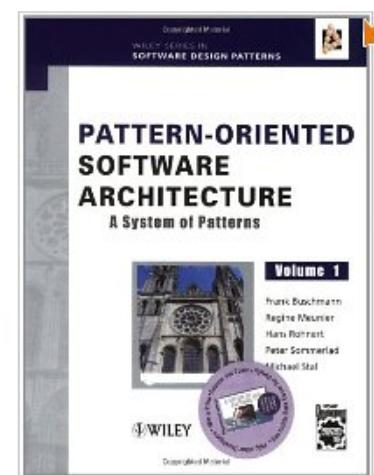
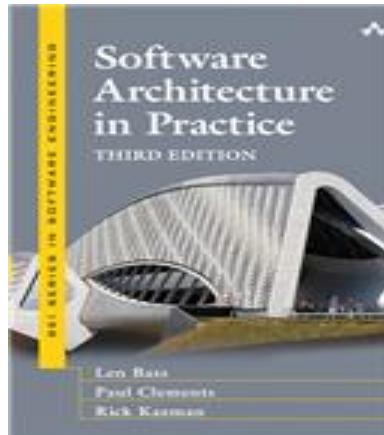
---

- To have sound understanding of software architecture
    - and remove misconceptions
    - the current state of the discipline of Software Architecture
    - Know the way in which architecture can impact design
    - Know various architectural styles, views
    - Importance of nonfunctional requirements, or quality attributes of a system
  - Apply the concept
    - Design new systems in principled ways, using well-understood architectural paradigms
    - Present concrete examples of actual system architectures that can serve as model for new designs
  - Evaluate
    - Evaluate designs of existing software systems from an architecture perspective
-

# Study Material

---

- Text Books
  - Len Bass et al, Software Architecture in Practice, Pearson, Third (or Second) Edition, ISBN 9789332502307
  - F. Buschmann et al, Pattern Oriented Software Architecture – Volume1, Wiley, 1996



# Study Material contd...

---

- References

- R. N. Taylor et al, Software Architecture: Foundations, Theory, and Practice, John Wiley & Sons, 2009
- Mary Shaw & David Garlan, Software Architecture – Perspectives on an Emerging Discipline, PHI, 1996.
- Stephen T. Albin, The Art of Software Architecture, Wiley Dreamtech, 2003.
- Gamma, E. et. Al. Design Patterns: Elements of Reusable Object Oriented Software, Addison Wesley, 1995

# Teaching and Evaluation

---

- Lectures: 16 + 2 Review
- Exams: 2
  - Midterm: 35%
  - Final: 50%
- Quizzes: 15%
- Midterm Exam
  - Closed Book and notes
- Final Exam
  - Open Book and notes
- The exam solutions/answers are expected to be of Masters Level with crisp, to-the-point, concise, proper, neat and readable presentation

# Detailed Schedule

Lecture#	Topics
1	Software Architecture and its Importance
2	Many perspectives of Software Architecture Introducing Quality Attributes
3-5	Role a few quality attributes in details – Architectural Tactics <ul style="list-style-type: none"> <li>• Availability, Interoperability, Modifiability</li> <li>• Performance, Security, Testability</li> </ul>
6-7	Object-oriented concepts and UML <ul style="list-style-type: none"> <li>• Classes, Objects, Encapsulation, Polymorphism, Inheritance and their representation in UML</li> <li>• Class diagram, Sequence diagram, Class Responsibility and Collaboration (CRC) Cards</li> </ul>
8	Styles and Patterns <ul style="list-style-type: none"> <li>• Concept, Categories and descriptions</li> <li>• Architecture style - Layering</li> </ul>
	Review Session
	Syllabus for Mid-Semester Test (Closed Book) : Topics covered in S. No. 1 to 8

# Detailed Schedule contd...

Lecture#	Topics
10-12	Architecture Style: Blackboard style, Pipe and Filter style Distributed System Style: Broker Architecture Interactive System Style Model-View-Controller Adaptable System Microkernel, Reflection
13-17	Design Pattern <ul style="list-style-type: none"> <li>Components of a typical design pattern, and different categories of design patterns</li> </ul> Behavioral Category <ul style="list-style-type: none"> <li>Iterator Pattern</li> </ul> Behavioral Category <ul style="list-style-type: none"> <li>Observer, Strategy, Visitor, Command</li> </ul> Structural Category <ul style="list-style-type: none"> <li>Adapter, Decorator, Composite, Proxy</li> </ul> Creational Category <ul style="list-style-type: none"> <li>Factory Pattern, Factory Method, Singleton</li> </ul>
Review Session	
Syllabus for Comprehensive Examination (Open Book): All topics given in the Plan.	