# Software Architecture

Ergude Bao

Beijing Jiaotong University

# Contact


- Ergude Bao, PhD., Professor
- Office: YF709
- Email: baoe@bjtu.edu.cn
- Telephone: 51684116

- Office hour: By appointment

# References

- Software Architecture in Practice, Second Edition, By Len Bass, Paul Clements, Rick Kazman, Publisher: Addison Wesley, ISBN：0-321-15495-9
- Design Principles and Design Patterns, By Robert C. Martin, Free online book from www.objectmentor.com
- Design Patterns in Java Tutorial, Free online book from tutorialspoint.com
- The Design Patterns Java Companion, By James W. Cooper, Free online book
- These slides

# Evaluation

- Final score is composed of three parts:
  - Written exam 50%
  - Course project 40%
    - See project description file
  - Thesis 10%
    - Writing about knowledge structure of this course
    - Handed-in together with course project

# Course Overview

- Introduction to software architecture (2 units)
- Modeling of software architecture (2 units)
- Quality attributes of software architecture (2 units)
- Design principles (8 units)
- Architectural styles (2 units)
- Design patterns (12 units)
- Software product line (2 units)

# Introduction to Software Architecture

Ergude Bao

Beijing Jiaotong University

# Content

- History of Software Development
- Definition of Software Architecture
- Related Concepts of Software Architecture
- Influential Factors on Software Architecture
- Benefits of Software Architecture
- Current Researches & Practice in Software Architecture

# History of Software Development

# History of Software Development

- ***Assemble Language***
- ***Small size of program***

***1970'*** ▸

- ***Advanced Language***
- ***Structure-Oriented Theory***
- ***Dataflow/Control flow Design Methods***

***1980'*** ▸

- ***Application Development Library: Class/Functions Library***
- ***Object-Oriented Theory***
- ***Object Modeling & Design Technology***

***1995*** ▸

- ***Application Development Framework: J2EE, .NET***
- ***Component Technology: COM/DCOM, CORBA …***
- ***Object Modeling & Design Standardization: UML***

***Future*** ▸

- ***Model-Driven Development: MDA***
- ***……***

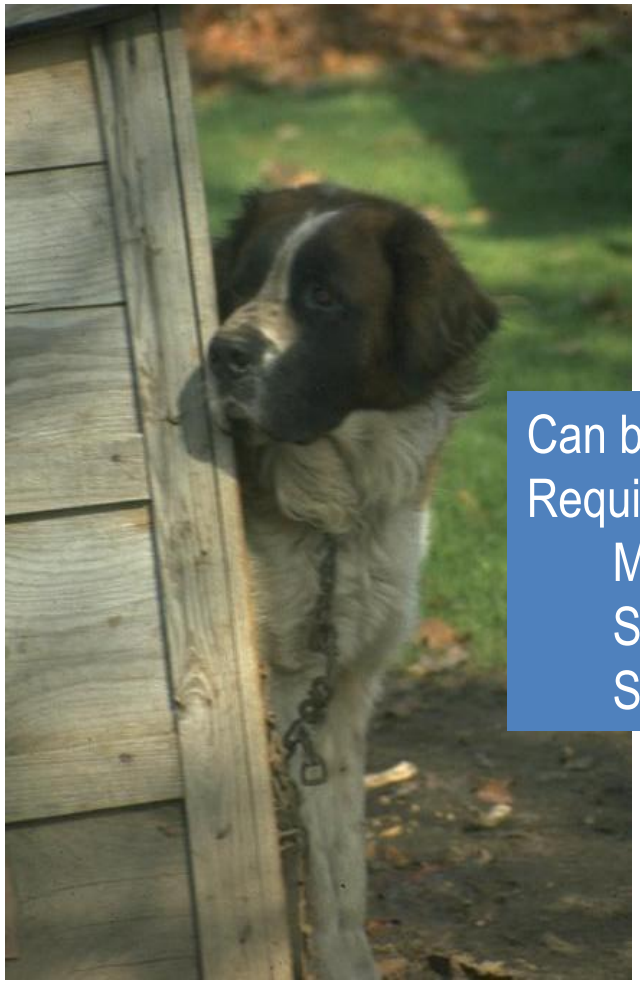
# Analysis of the Software History

- The size and the complexity of software is becoming larger and more complex
  - The application areas of software: science computing, manufacturing, commerce, education and amusement
- The abstraction level of software is becoming higher
  - Machine Language—>Assemble Language—>Advanced Language—> Application Framework
- The design of software is becoming more and more sophisticated
  - Structure-Oriented Programming —> Object-Oriented Programming—> Aspect-Oriented Programming

# Conclusion of the Software History

- The architecture and designing is more important than the data structure and the program algorithm
- Good architecture design has always been a major factor in determining the success of a software system

# Analog



Can be built by one person
Requires
- Minimal modeling
- Simple process
- Simple tools

# Analog

Built most efficiently and timely by a team
Requires
- Modeling
- Well-defined process
- Power tools

# Analog

# Analog

- Different in
  - Scale
  - Process
  - Cost
  - Schedule
  - Skills and development teams
  - Materials and technologies
  - Stakeholders
  - Risks

# Definition of Software Architecture

# Existing Definitions

- The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them
  - “Software Architecture in Practice”, Addison-Wesley, 1997
- Architecture is the organizational structure of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems
  - UML 1.3
- Software architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution
  - IEEE 1471-2000

# Analysis of Existing Definitions

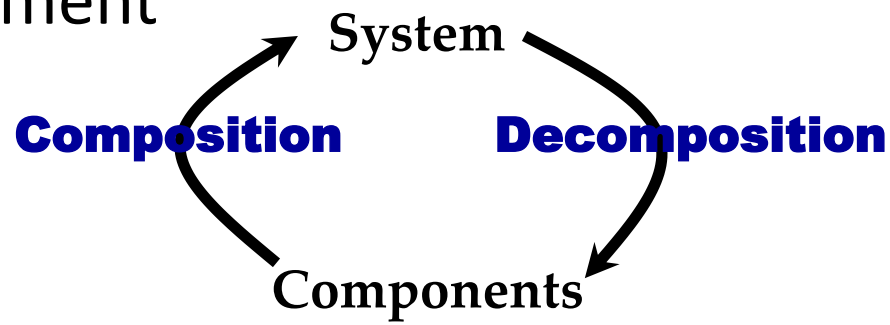
- Software architecture is composed of
  - Software elements: functions, Interfaces, programs, class modules, layers, subsystem, clients/servers, etc.
  - Properties: provided services, performance characteristics, fault handling, usage of shared resources, etc.
  - Relationships: composition mechanism and style of the elements

# Brief Definition

- Software architecture can be defined briefly as
  - Components comprised in the system, and the relationships or interaction mechanisms of those components
    - Component: constituent elements
    - Connector: Interaction rules/mechanism
- An architecture is the result of a set of business and technical decisions

# Software Architecture Design

- Software architecture design = decomposition + composition
  - Reduce the complexity of software design and construction
  - Control the risks of software development
  - Improve the efficiency of organization and management



# Software Architecture Design

- But, we must consider
  - How do we break the system down into pieces?
  - Do we have all the necessary pieces?
  - Do the pieces fit together?

# More Definitions

- Hundreds of definitions on CMU web page: http://www.sei.cmu.edu/architecture/definitions.html

# Related Concepts of Software Architecture

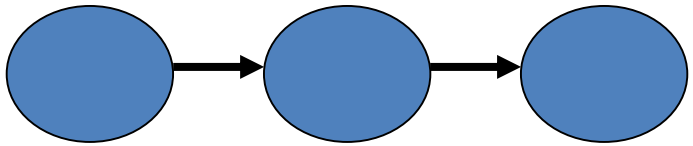
# Component

- Logical and functional unit of the system serving certain responsibilities
- Could be different specific objects in different scenarios
  - Modules, subsystems, layers, packages, classes etc.
- Could be divided into smaller units of components

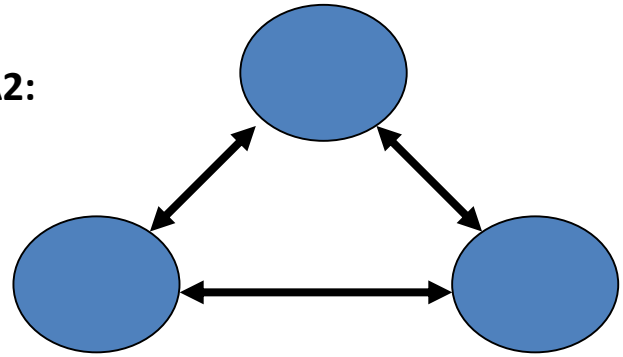
# Connector

- Interaction rules or mechanisms among components

**SA1:**



**SA2:**



**SA3:**

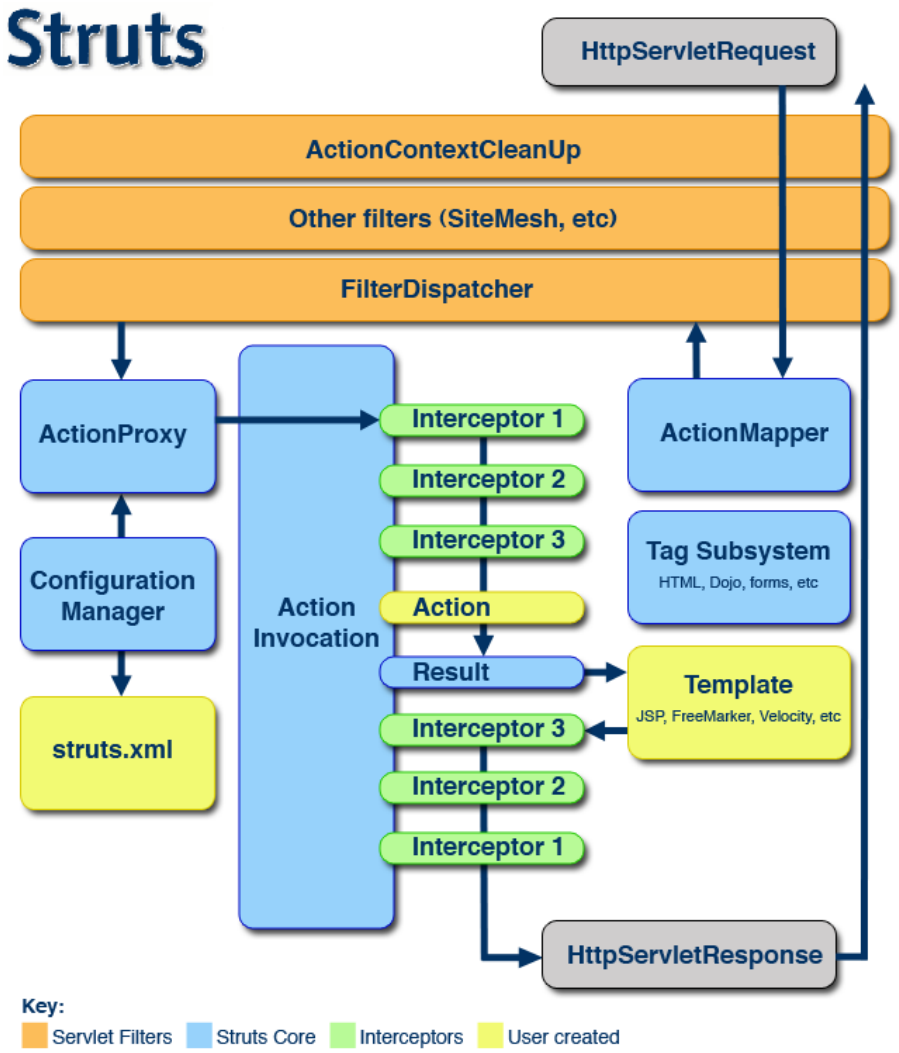
# Properties

- Functional property
  - Characteristic meeting  functional requirement
- Non-functional property
  - Characteristic meeting non-functional requirement
    - Performance
    - Portability
    - Flexibility/extensibility
    - Reliability/security

# Framework

- Reusable application infrastructure for specified problems
  - With necessary basic components and connectors for the specified problems
- Context or environment for the applications developed based on it
- J2EE framework, .Net framework, etc.

# Struts2 Framework



# Influential Factors on Software Architecture

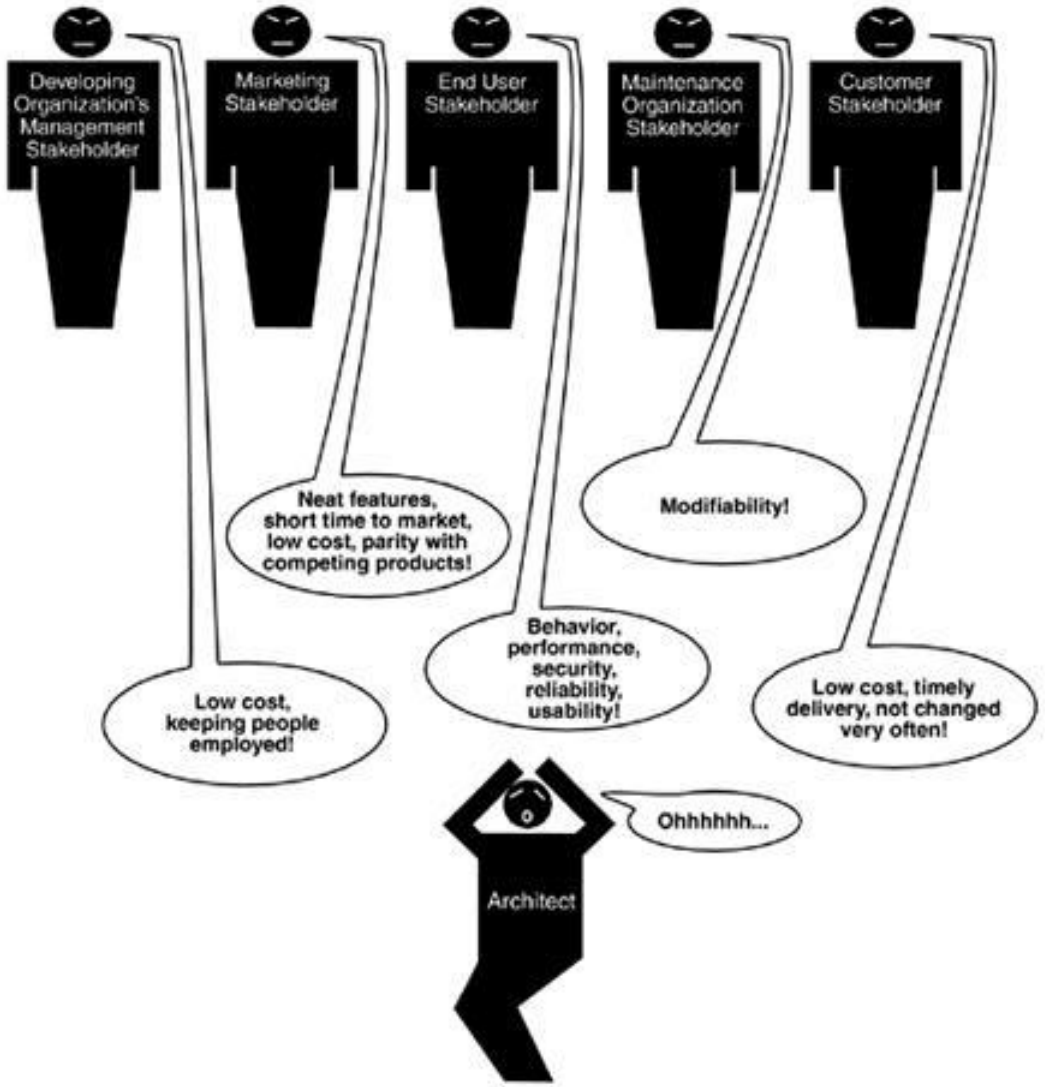
# Influential Factors

- System stakeholders
  - Many stakeholders, many views
- Developing organization
- Background and experience of the architect
- Technical environment

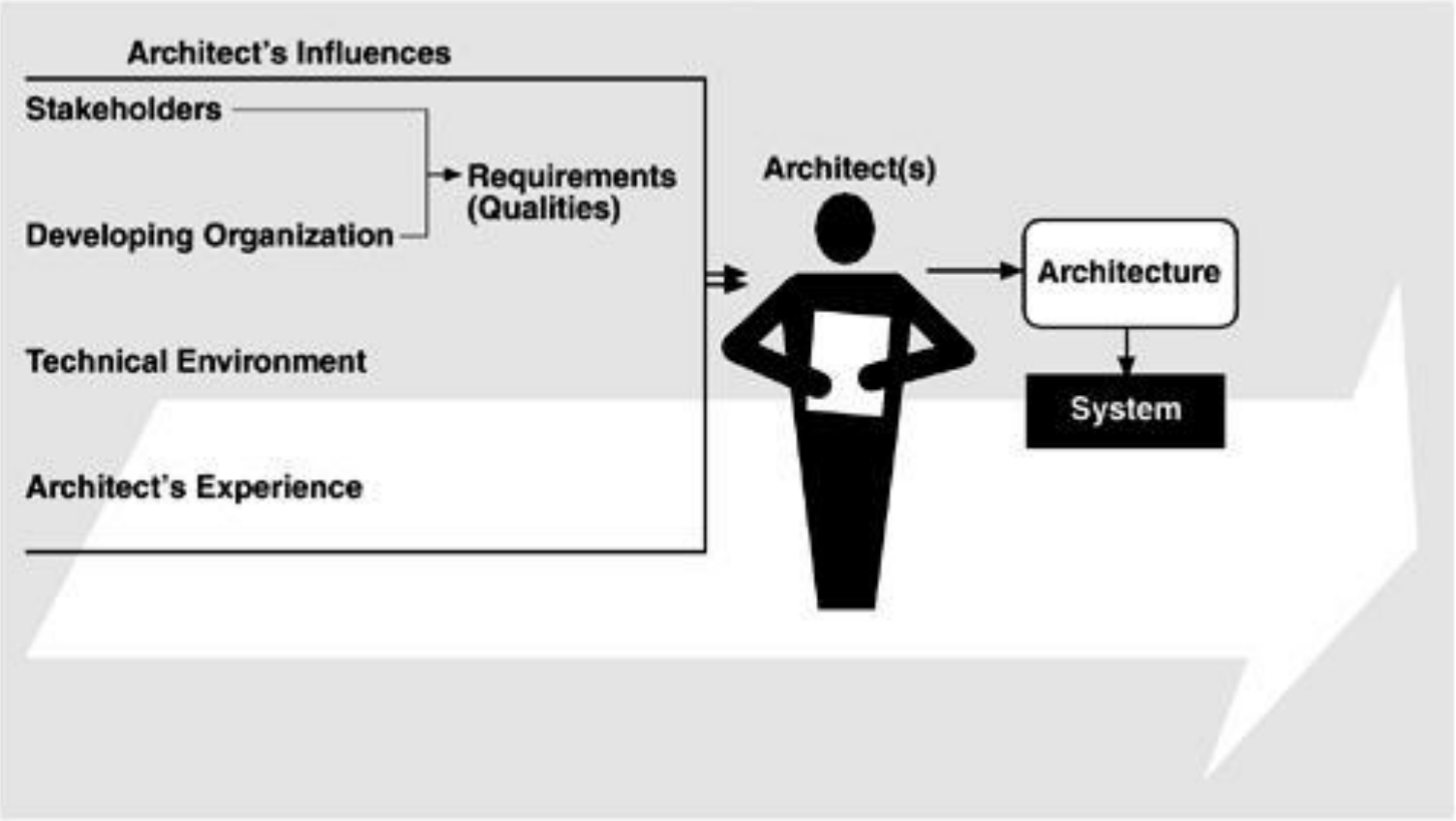
# Many Stakeholders, Many Views

- Architecture is a different thing for different stakeholders
  - End-user
  - Customer
  - Project manager
  - System engineer
  - Developer
  - Architect
  - Maintainer
  - Other developers
- Multiple stakeholders, multiple views, multiple blueprints

# Influence of stakeholders on the architect

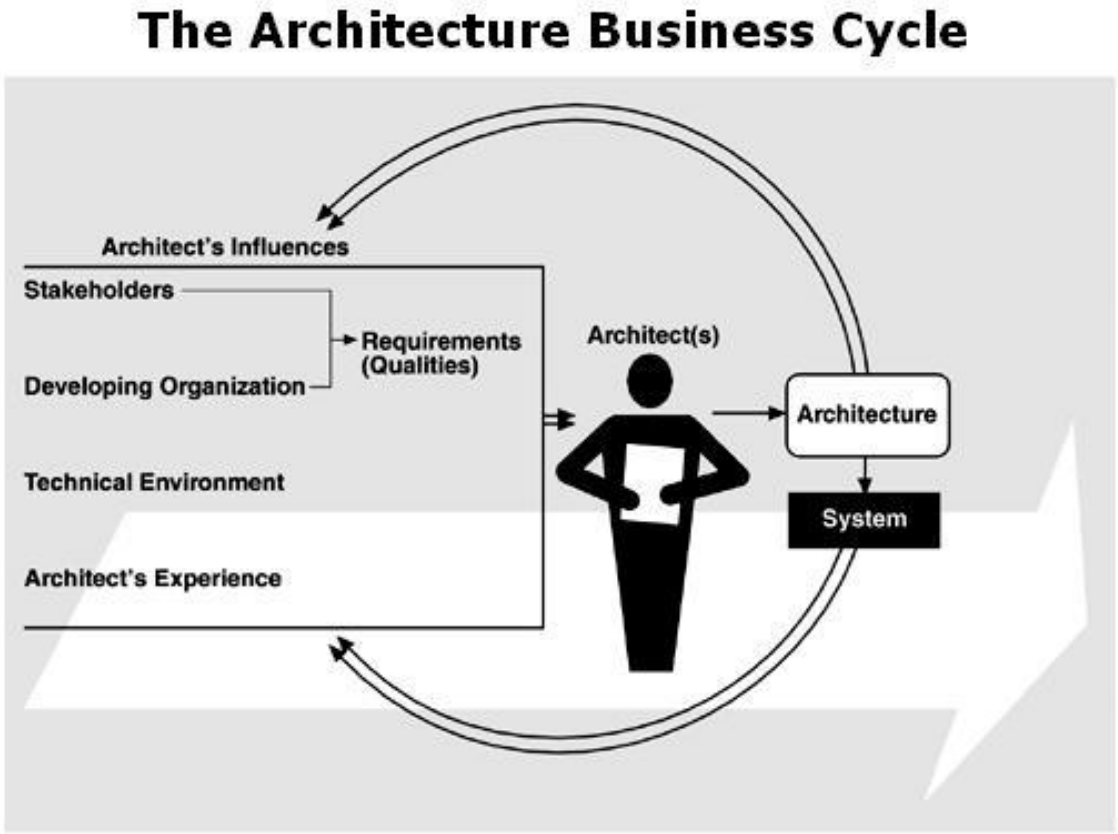


# Influences on the architecture



# Feedback Loop

- Architecture affects the influential factors reversely



# Architecture Business Cycle

- Create the business case for the system
- Understand the requirements
- Create or select the architecture
- Document and communicate the architecture
- Analyze or evaluate the architecture
- Implement the system based on the architecture
- Ensure the implementation conforms to the architecture

# Benefits of Software Architecture

# Technical purposes

- Meet system requirements and objectives
- Specify constraints for implementation and tests
- Enable flexible distribution/partitioning of the system
- Reduce cost of maintenance and evolution
- Increase reuse and integrate with legacy and third party software

# Organizational purposes

- Provide high-level information for stakeholders
  - Costs and risks evaluation
  - Work allocation and project schedule
- Facilitate communications among stakeholders

# Characteristics of Good Architecture

- Resilient
- Simple
- Approachable
- Clear separation of concerns
- Balanced distribution of responsibilities
- Balanced economic and technology constraints

# Current Researches & Practice in Software Architecture

# Current Researches & Practice in Software Architecture

- Formalization Research
  - How to describe software architecture with specific rules
  - Architecture Description Language (ADL)
- Verification & Evaluation Research
  - How to verify and evaluate whether a software architecture meets all the functional and non-functional requirements
  - Architecture Tradeoff Analysis Method (ATAM)

# Current Researches & Practice in Software Architecture

- Most up-to-date research comes from the International Conference on Software Engineering (ICSE): http://2015.icse-conferences.org