ReflectAll

Combining Reflective Component Model and Reflective Middleware

Gang HUANG, Ling LAN, Jie YANG, Hong MEI

School of Electronics Engineering and Computer Science

Peking University, Beijing, 100871, China

July 3, 2006, Nantes, France

Peking University

Introduction to SEI@PKU

- Software Engineering Institute in Peking Univ.
  - 7 full professors, 10 associate professors, 9 assistant professors
  - >30 Ph.D students, >70 graduates
    - The biggest SE team in Chinese universities
  - Cover almost all areas of software engineering with emphasis on component based reuse
    - domain engineering, object oriented modeling, software architecture, middleware, component repository, testing, program comprehension, software process
  - http://www.sei.pku.edu.cn

Peking University

Background of This Work

- Component Model
  - Fractal & ABC
    - Software architecture group is the core group of ABC
    - ABC/ADL & ABCTool
    - 7 PhD students, 6 graduates
- Next Generation J2EE
  - JonAS & PKUAS
    - PKUAS group is the biggest group in SEI@PKU
    - 7 PhD students, 5 graduates in experience sub-group
    - >20 graduates in practice sub-group
- Autonomic System Management
  - JADE & ABC/PKUAS

Peking University

Agenda

- ❑ **Motivation**
  - ➢ **Why leverage reflective component and reflective middleware**
- ❑ **Prototype and Demo**
  - ➢ **Prototype on J2EE (PKUAS & JonAS)**
  - ➢ **Demo of JPS: Password Protection**
- ❑ **Lessons Leant**
  - ➢ **Fractal v2 controllers are not sufficient & necessary**
  - ➢ **Evolution other than revolution to reflection**
  - ➢ **Managing reflective systems in the whole lifecycle**
- ❑ **Conclusion and Future Work**

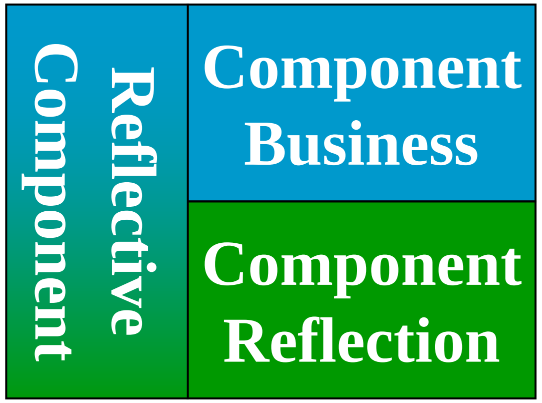
Reflection

❑ Reflection

- Also known as computational reflection, is originated by B.C. Smith to access and manipulate the LISP program as a set of data in execution.
  - *Smith, B.C. Procedural Reflection in Programming Languages. Ph.D Thesis, MIT, 1982.*
- As a promising way to achieve high adaptability, reflection is propagated into more programming languages, operating systems and distributed systems, and so on.
  - *3-KRS, Prolog, CLOS, Smalltalk, Java, C# …*
  - *Apertos, MetaOS, 2K …*
  - *CodA, GARF …*
- Component based systems also need reflection

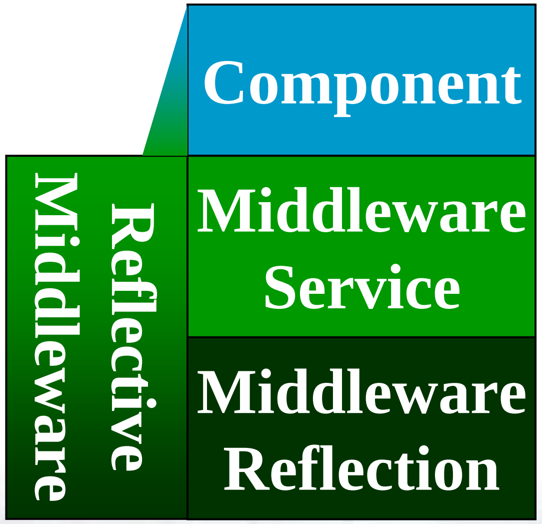
Peking University

Reflection in Component based Systems



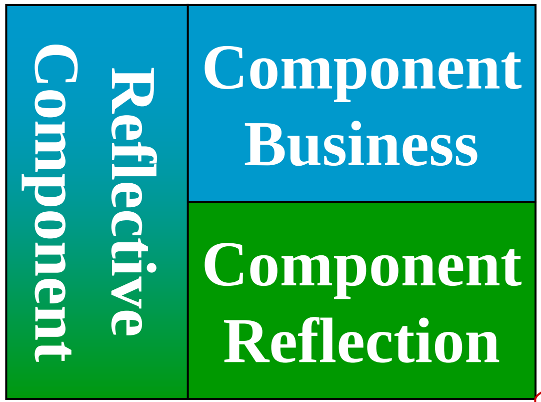
Reflective Component Model:
What is a reflective component
e.g. Fractal, OpenORB, K-Component

Middleware for Reflection:
How can a component be reflective
e.g. Julia, AOKell, OpenCOM



Reflective Middleware:
Making traditional middleware reflective
e.g. OpenCORBA, dynamicTAO, FlexiNET, MChaRM, PKUAS
Different with middleware for reflection though some functions are similar

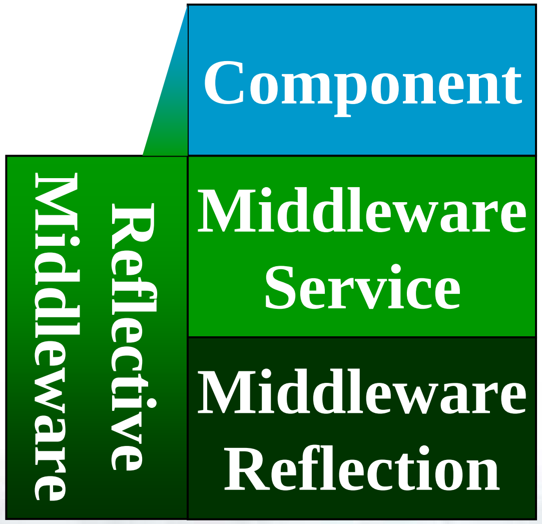
Reflective Component vs. Reflective Middleware



Reflective Component:

- + Formal programming model
- ± Easy to understand (by application developers)
- ± Encourage as well as rely on application developers for implementing reflection

Unfortunately, neither of them is sufficient for popular application of reflection



Reflective Middleware:

- – Ad hoc programming model
- ± Easy to understand (by middleware vendors)
- ± Release as well as prevent application developers from reflection impl
- + Well monitoring and controlling out side of components

Recap of Reflection's Promise

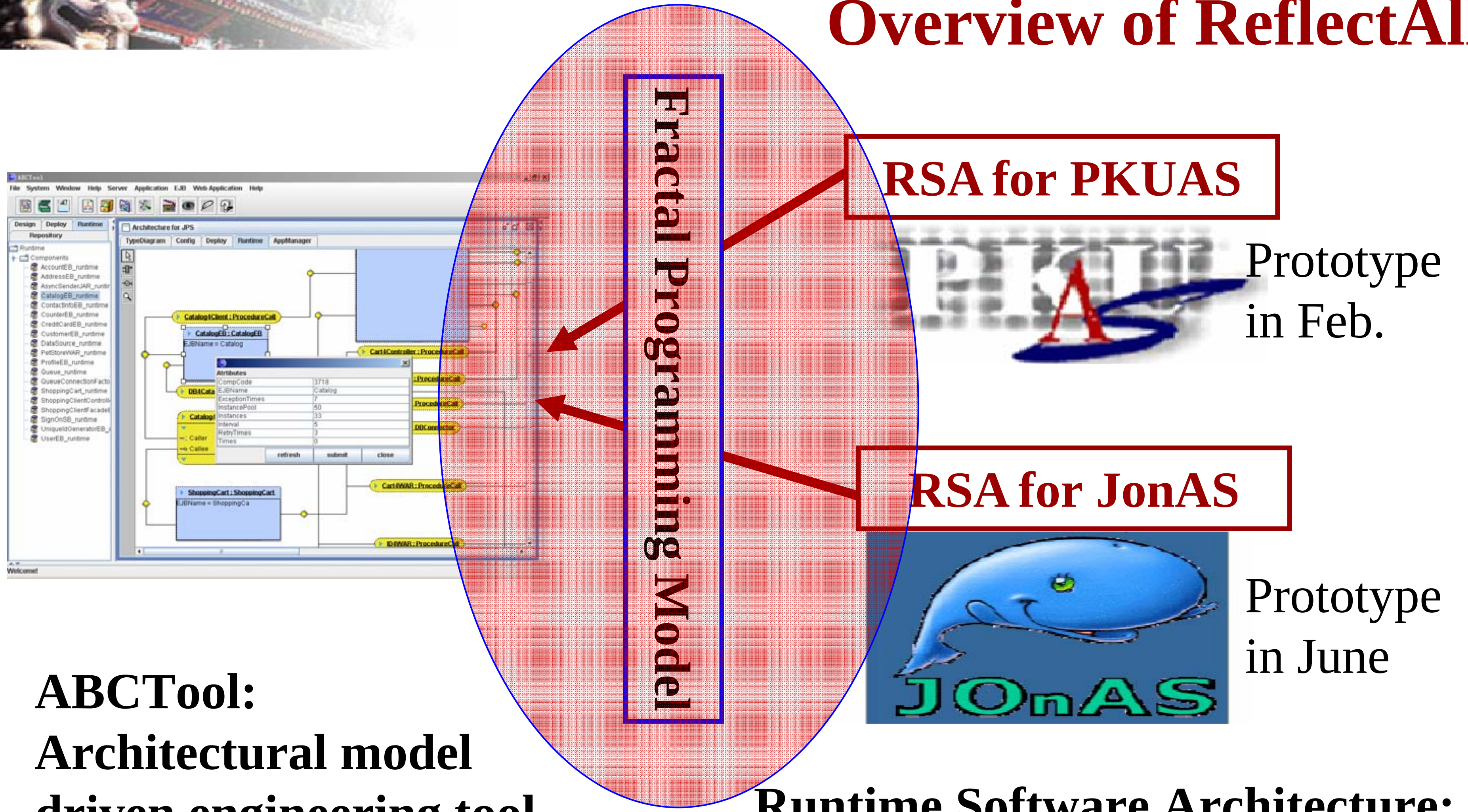
- Reflection is a promising way to achieve high adaptability
  - Everything in a runtime system may be to change
    - Reflective component cannot change middleware and vice versa
  - Everything is changed by a condition at a time
    - Different changes may be understood from different views (application or middleware)
- Usability is a key to practice of new technology
  - Easy to use (programming model of reflective component)
  - Easy to reuse (reusable functions of reflective middleware)
- It's the time to combine RC & RM

Goal of ReflectAll

- Demonstrate the combination of RC & RM
  - The combination is feasible
    - Reflective component & reflective middleware can be combined
  - The combination is promising
    - Keep the advantages while remove the disadvantages
- Review existing RC & RM
  - Limitation
  - Killer application

Peking University

Overview of ReflectAll



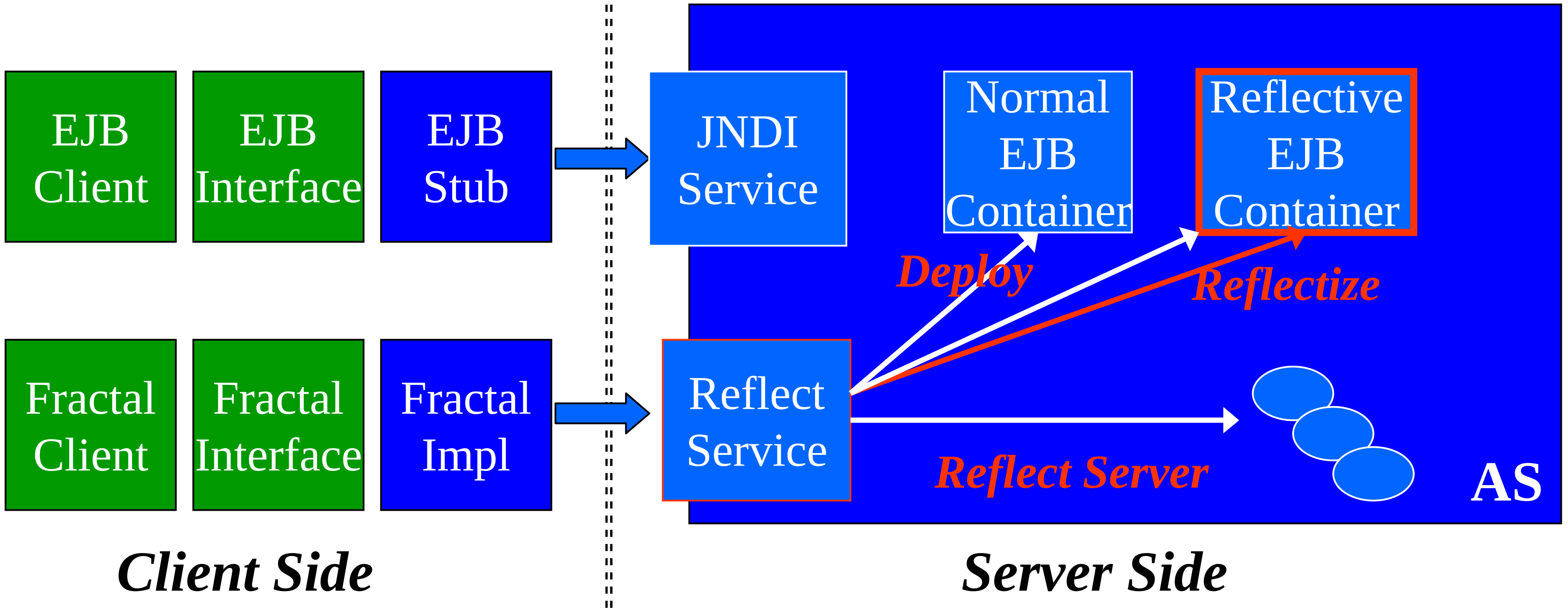
ABCTool: Architectural model driven engineering tool covering the whole lifecycle of component based systems

Runtime Software Architecture: Reference model for architecture based reflective middleware

Huang G, Mei H, Yang F. Runtime Recovery and Manipulation of Software Architecture of Component-based Systems. Journal of Automated Software Engineering, Springer, Vol. 13 No. 2, 251-278, Feb. 2006

ReflectAll: Server Level Architecture

- Leveraging reflective component and reflective middleware for reflecting all things in a component based system

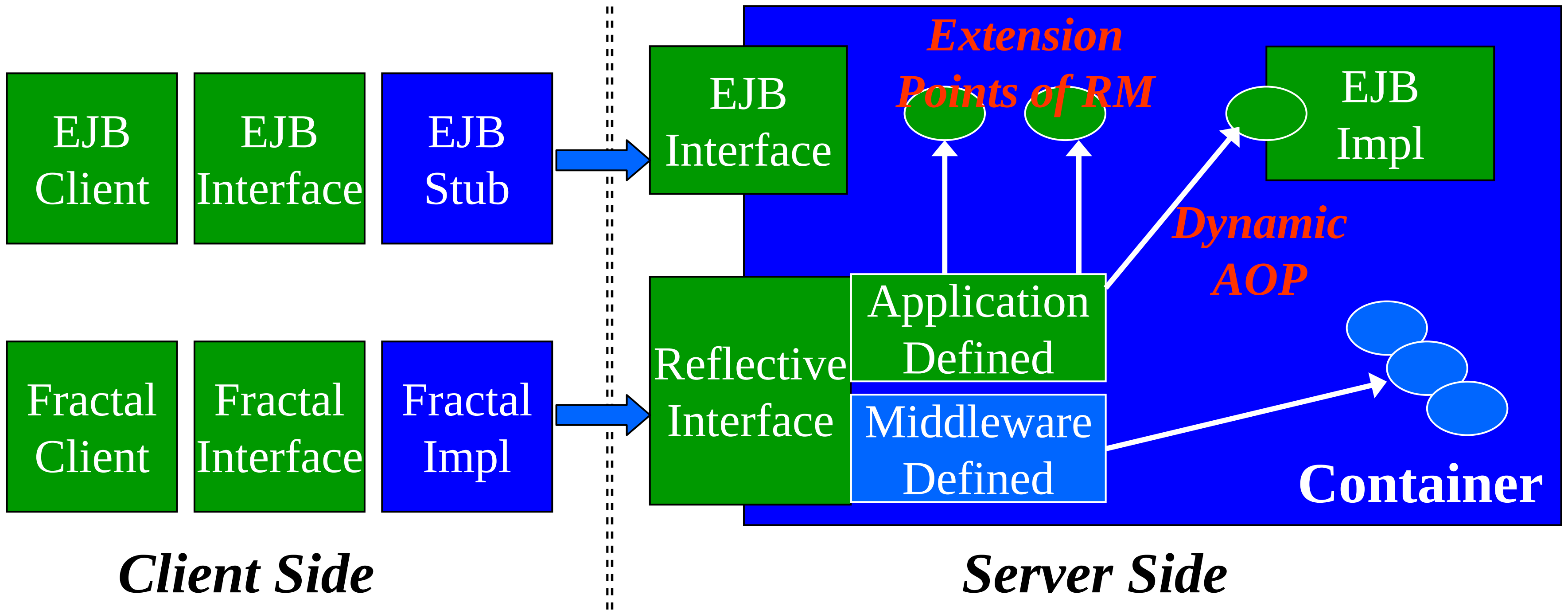


🟩	Objects by App Developers
🟦	Objects of Reflective Middleware
●	Meta Objects of Reflective Middleware

Peking University

ReflectAll: Container Level Architecture

- **All things can be reflected by the collaboration between middleware vendors and application developers**



🟢 Meta Objects by App Developers

🔵 Meta Objects of Reflective Middleware

AOP is not enough for reflection

Demo of JPS: Password Protection

- Change JPS at runtime without any modification to the source code

- Four steps
  - Opening the design artifacts of the application to be managed
  - Incarnating the runtime software architecture
  - Customizing the reflective components when necessary
  - Managing the runtime system

- JonAS Demo will be published in ObjectWeb
  - Modified JonAS v4.7.1
  - Source code of controllers, JPS deployable package
  - ABCTool English version

Lessons Learnt

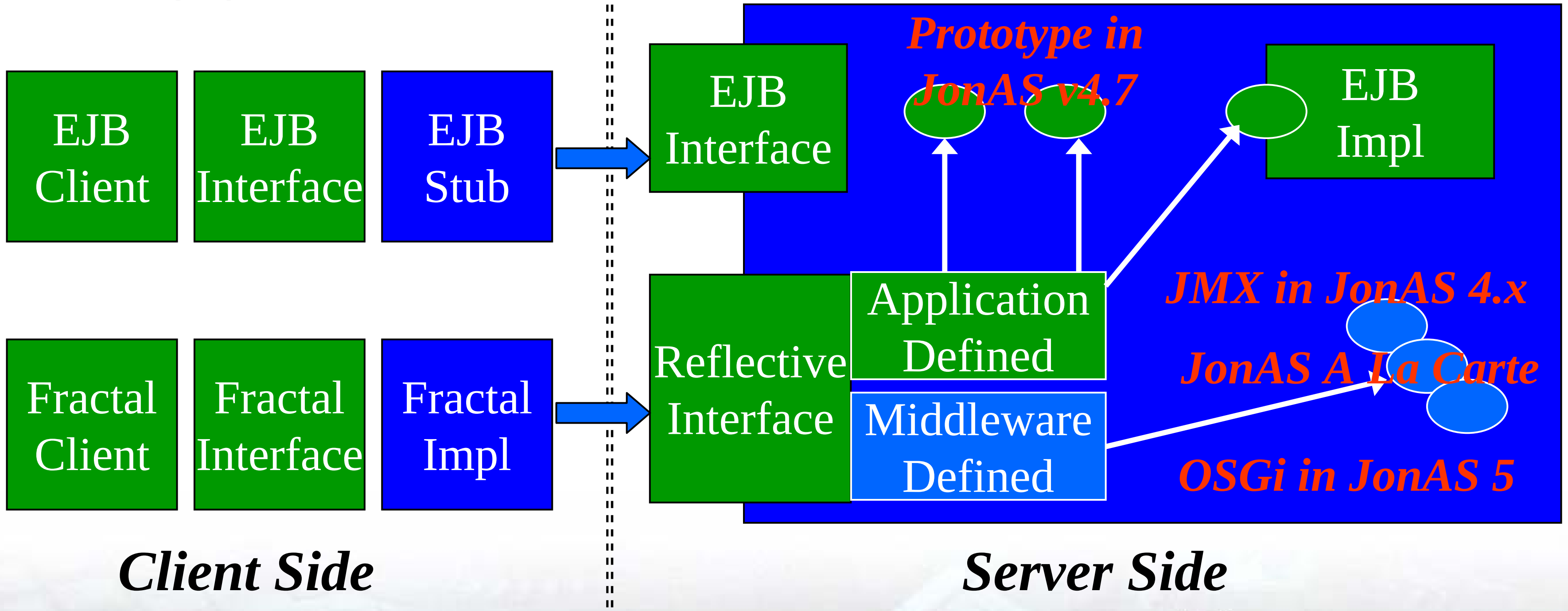
- ❑ Fractal v2 controllers are not sufficient & necessary
  - ➢ Binding controller and some of the following controllers are useless in some cases
  - ➢ Controllers should be customizable at runtime
  - ➢ Connectors may be complex and need to be reflective
- ❑ Evolution other than revolution to reflection
- ❑ Managing reflective systems in the whole lifecycle

	Specific to	Already implemented?	Examples
Built-in	middleware	Yes	Attribute controller Lifecycle controller
Pre-defined	middleware	Yes but need configuration	Persistence controller Polymorphism controller
User-defined	application	Not yet but reusable	The two controllers in JPS demo

Peking University

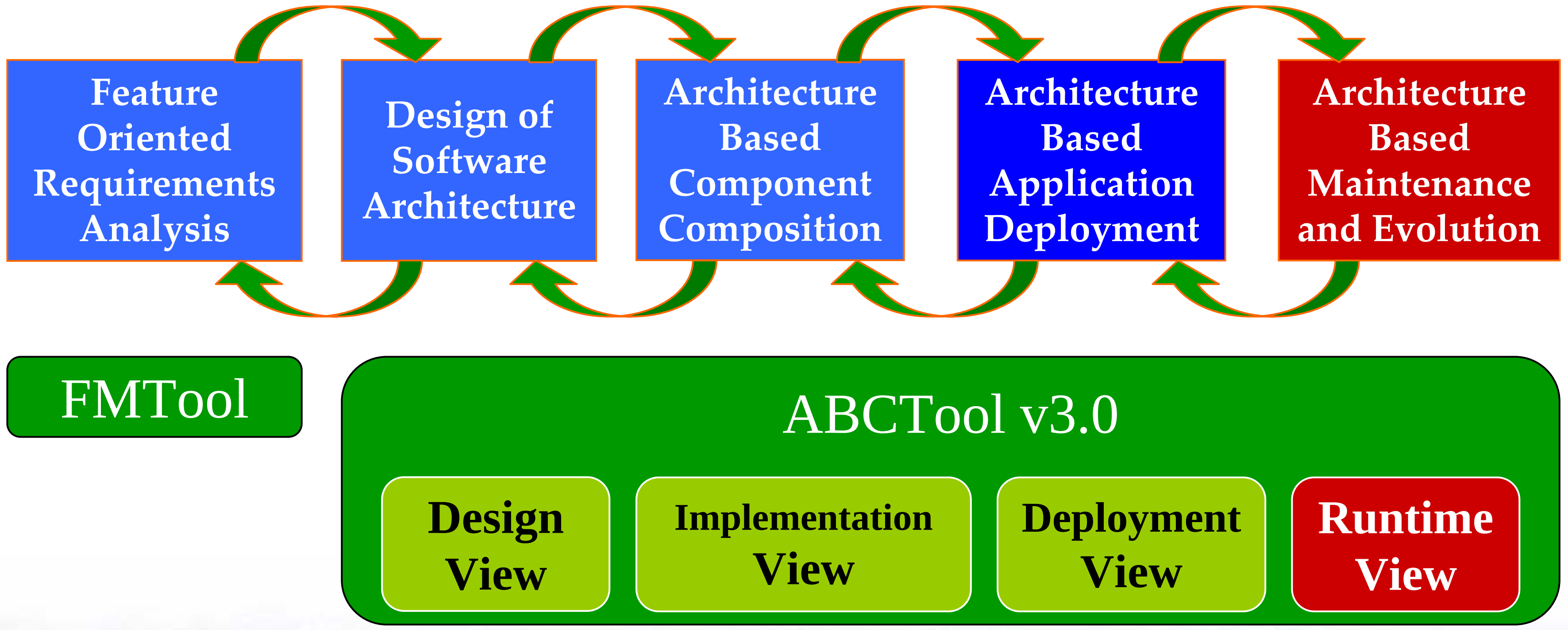
Lessons Learnt

- ❑ Fractal v2 controllers are not sufficient & necessary
- ❑ **Evolution other than revolution to reflection**
  - ➢ **Legacy systems cannot be ignored**
  - ➢ **Reflective mechanisms can be added one by one**
- ❑ Managing reflective systems in the whole lifecycle



Lessons Learnt

- ❑ Fractal v2 controllers are not sufficient & necessary
- ❑ Evolution other than revolution to reflection
- ❑ **Managing reflective systems in the whole lifecycle**
  - ➢ **ABC: architectural model driven approach**


Feature Oriented Requirements Analysis
Design of Software Architecture
Architecture Based Component Composition
Architecture Based Application Deployment
Architecture Based Maintenance and Evolution
FMTool
ABCTool v3.0
Design View
Implementation View
Deployment View
Runtime View

Conclusion & Future Work

- **Combination of reflective component and reflective middleware is necessary, feasible and promising**
  - **Demonstration on J2EE (PKUAS & JonAS)**
- **Combination identifies some future directions**
  - **A more flexible reflective component model**
  - **An evolutionary way to reflective systems**
  - **An architectural model driven approach to systematic use of reflection**
  - **In particular, deeper collaboration between PKU & ObjectWeb**
- **http://www.sei.pku.edu.cn/~huanggang/**

Peking University

Thanks