国 家 自 然 科 学 基 金 委 员 会
National Natural Science Foundation of China

A Brief Introduction of Software Architecture Research in China

Presented by Software *Architecture Group*

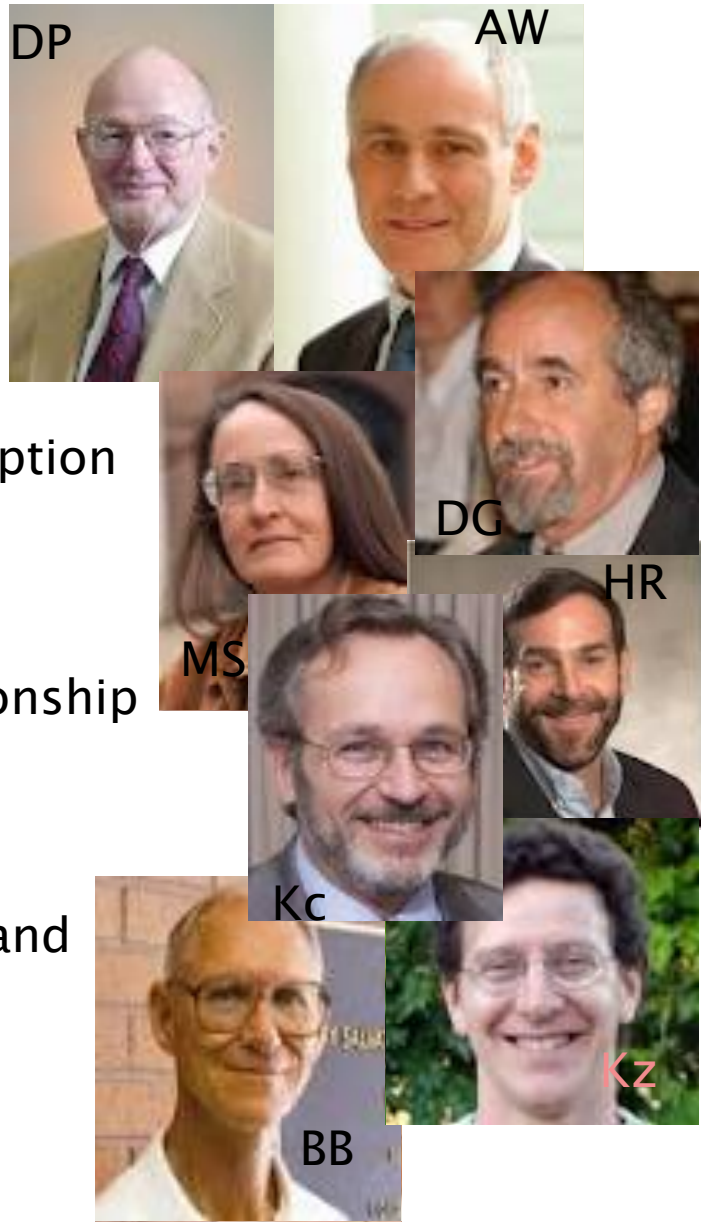
Sep-27-2011

Agenda

- Introduction
- Founding Support
- Basic Research and Program
- Representation Research Groups
- Conclusion

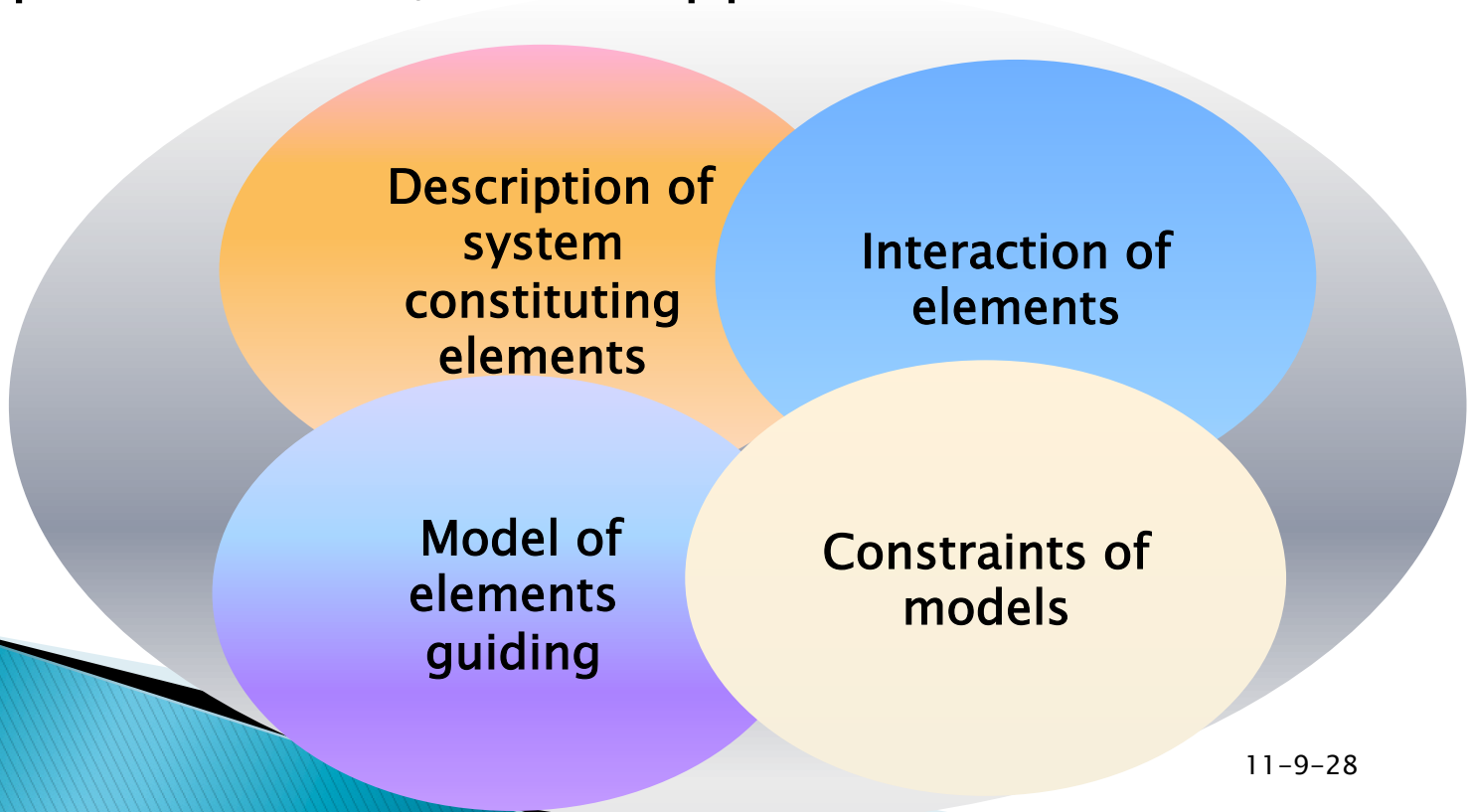
Cognition of SA

- Dewayne Perry & Alexander Wolf
  - Processing, Data, Connection
- Mary Shaw & David Garlan
  - Whole system structure design and description
- Kruchten
  - Concept, Module, Running, Code
- Hayes Roth
  - Function, connection, interface and relationship
- David Garlan & Dewayne Perry
  - Evolution approach of Program/System
- Barry Boehm
  - Software, system component, interactive and restriction
- Bass, Ctements & Kazman
  - Software components, external behavior, relationship



Our Cognition of SA

- Software architecture provides a high-level abstraction of structure, activities and properties for software system in reflection of the underlying platform and/or the application domain.

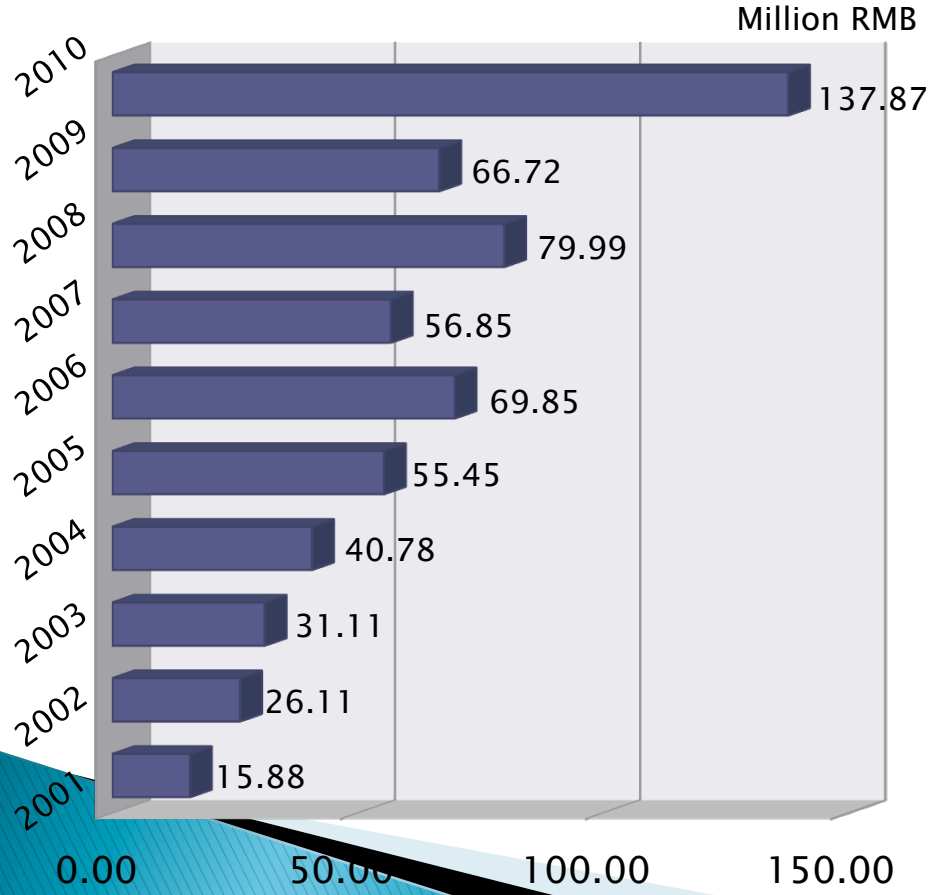


Agenda

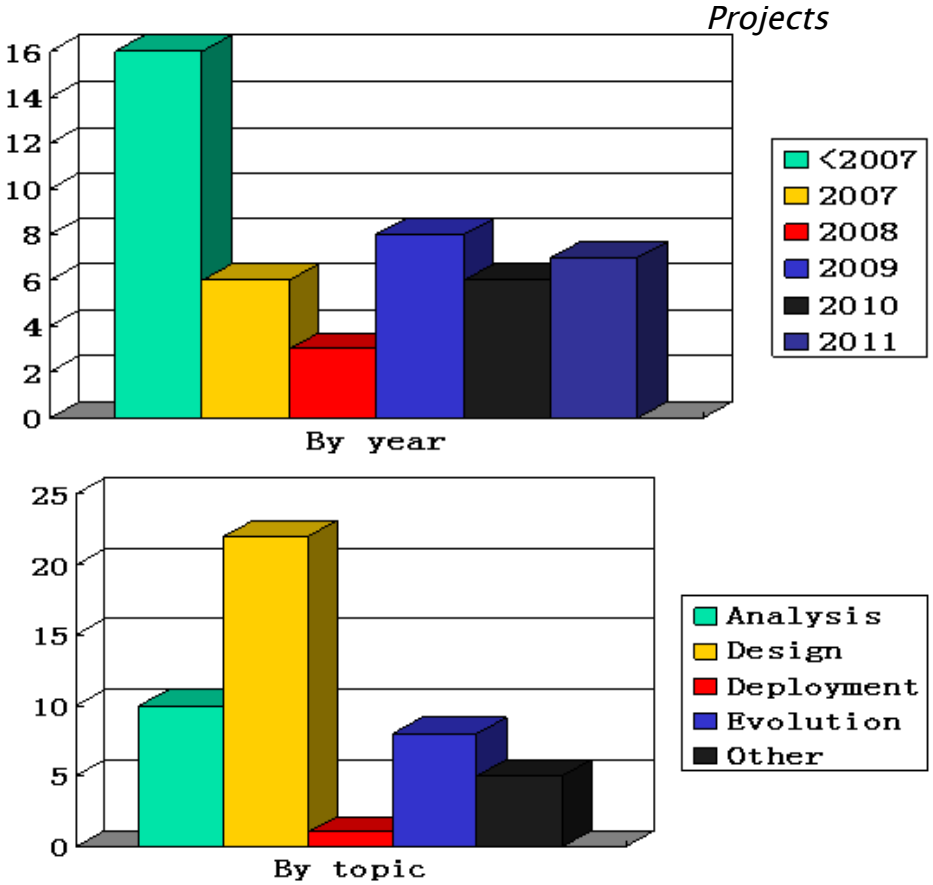
- Introduction
- Founding Support
- Basic Research and Program
- Representative Research Groups
- Conclusion

National Investment in Computer Science and Software Architecture

NSFC Funding for Computer Science(2001~2010)

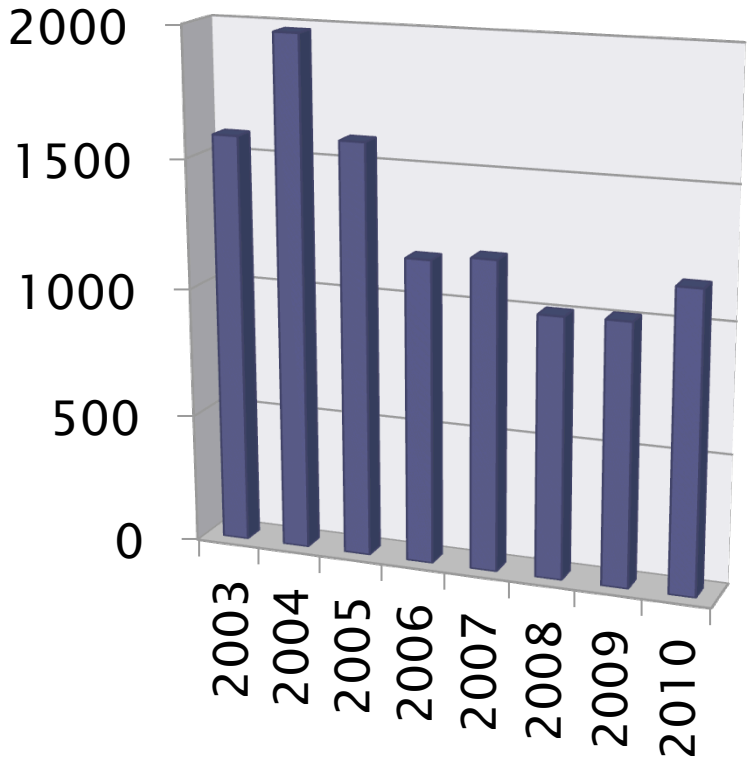


NSFC Funding for Software Architecture(2001~2010)

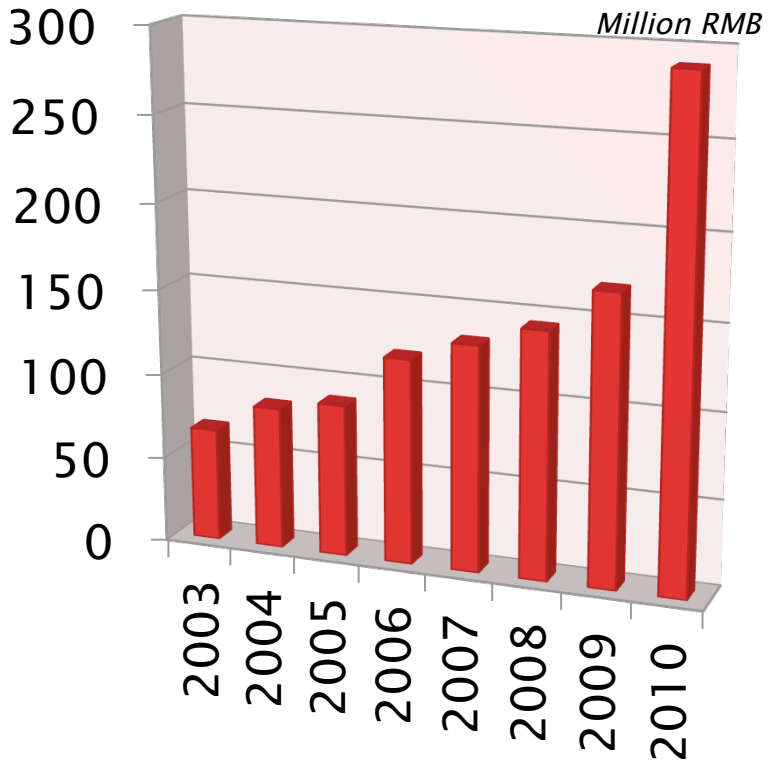


International Cooperation and communication

- By projects



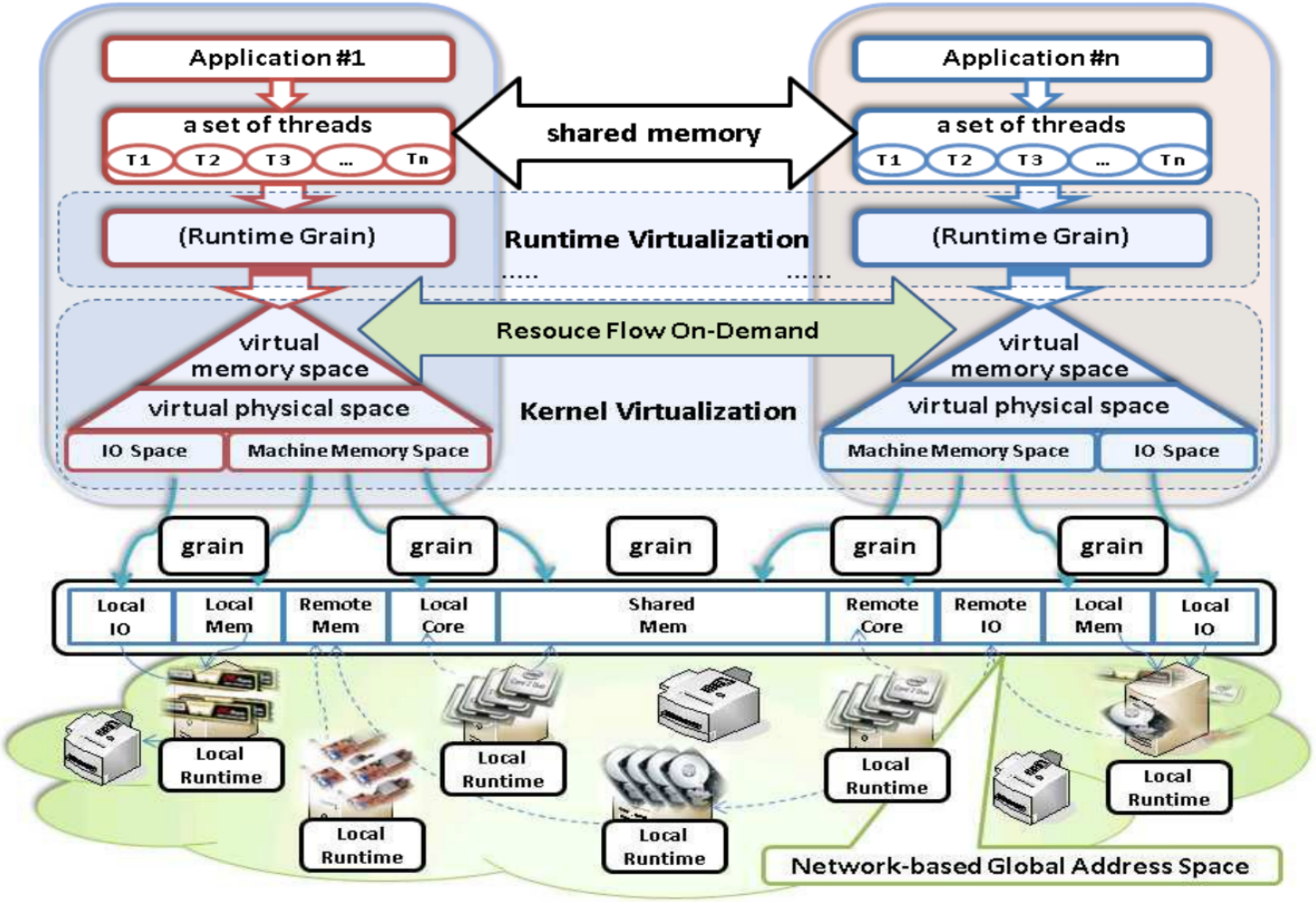
- By funding



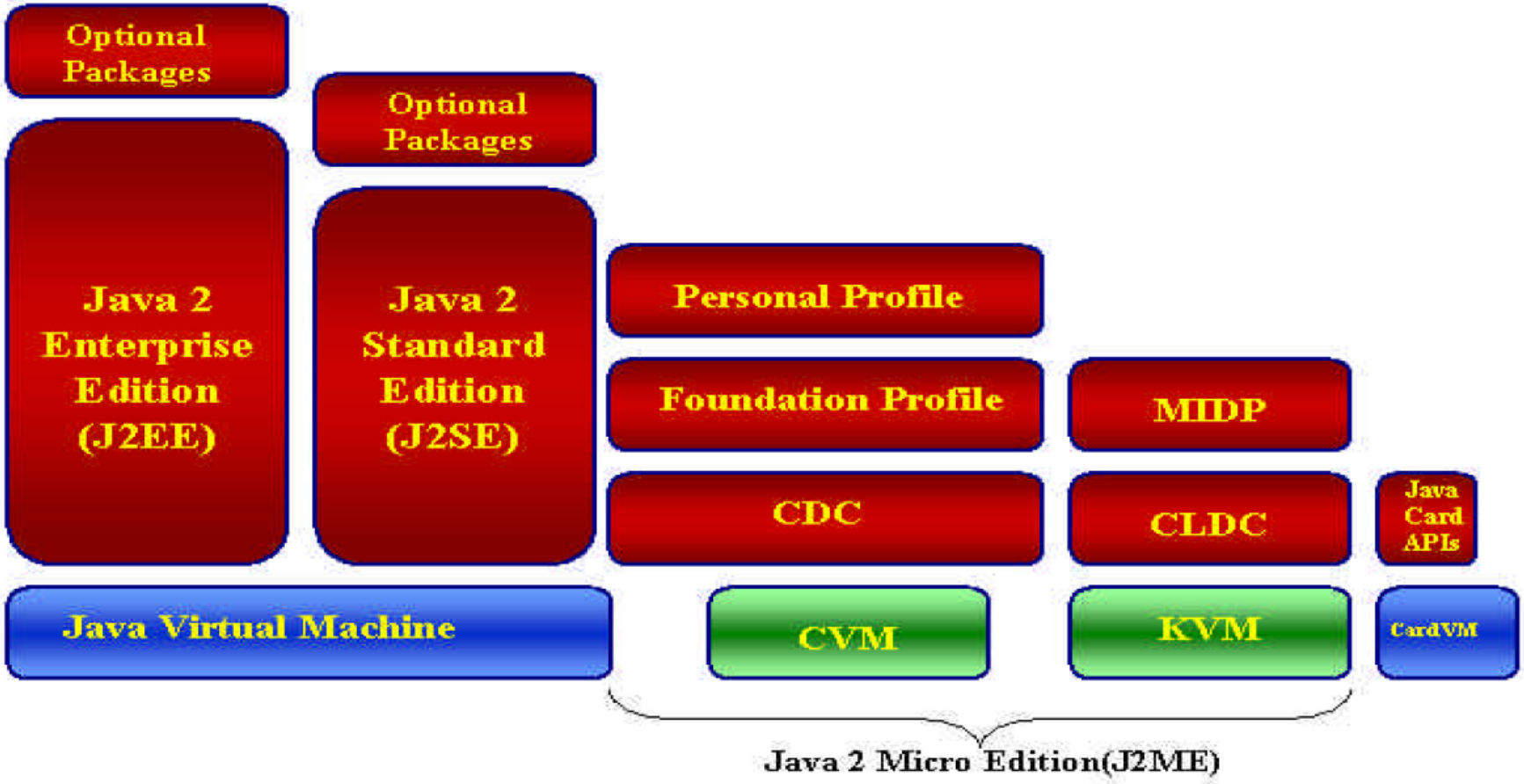
Agenda

- Introduction
- Founding Support
- Basic Research and Program
- Representative Research Groups
- Conclusion

Credible computing environment based on Virtual machine architecture and credible software design



The new generation network middleware architecture, protocol and implementation mechanism



Agenda

- Introduction
- Founding Support
- Basic Research and Program
- Representative Research Groups
  - Shanghai Jiao Tong University
  - National University of Defense Technology
  - Peking University
  - Fudan University
  - Nanjing University
- Conclusion

Representative Research Groups (1/5)

Shanghai Jiao Tong University

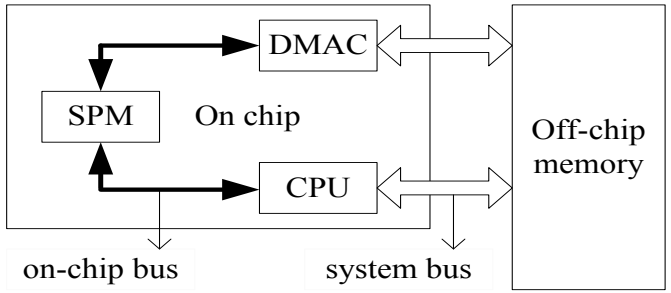
- Focus: Architecture for Distributed Computing
- Typical project
  - Key technologies of Compiler Optimization for on-chip Multi-core Processor
  - Computer Networks and Distributed computing Systems
- Website: http://epcc.sjtu.edu.cn
- Main publications:
  - IEEE TPDS，IEEE TC, TVC, CCPE, CIKM, WWW, JPDC, IPDPS, PACT, etc.

Compiler Optimization for on-chip multi-core processor

- **Multi/Many-core processor** is the inevitable trend of development of processor, and has been applied in desktop, server and mobile and embedded system.
- However, the **power**, **memory** and the **parallelization of software** limited the development of multi-core processor.
- In this project, we studied several key technologies of the compiler optimization for on-chip multi-core processor:
  - **Architecture of Multi-core Processor**,
  - **Acceleration Model**
  - **Compiler Optimization**
  - **Low-power framework and methods**
  - **Task Stealing Scheduling**
  - etc.

Multi-core Processor Modeling

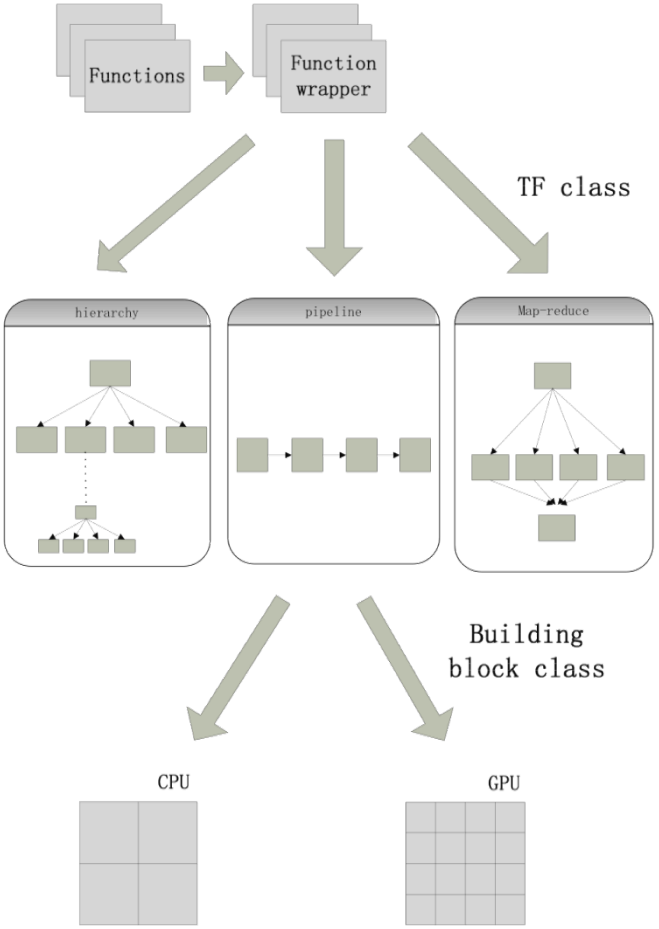
- **Scratch Pad Memory Model**
  - Characteristic: Small-Capacity; low-power Consumption; high- speed access
  - Allocation strategy: Static Allocation; Dynamic allocation
- **Data Pipeline Model**
  - Dynamic voltage scaling
  - Instruction-level speed modeling
  - Graphical data fetch modeling
  - Independence between DMA and CPU
  - Data Predictability from loop iteration
- **Power Cost Model**
  - Quantify static power consumption
  - Quantify command power
  - Quantify power of cache and memory access
  - Multi-core communication power modeling



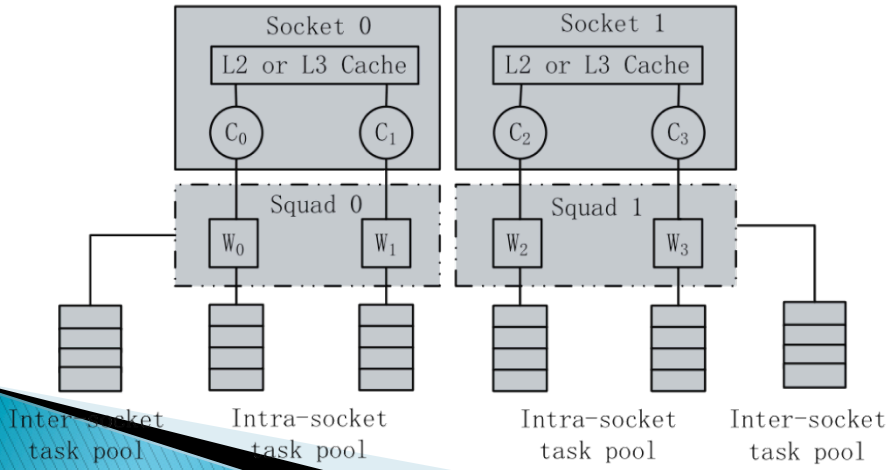
Basic model of McP

Multi-thread parallelization

- Template Meta-programming method
- Architecture of bi-tier task-stealing
- Cycle of sentence migration transformation



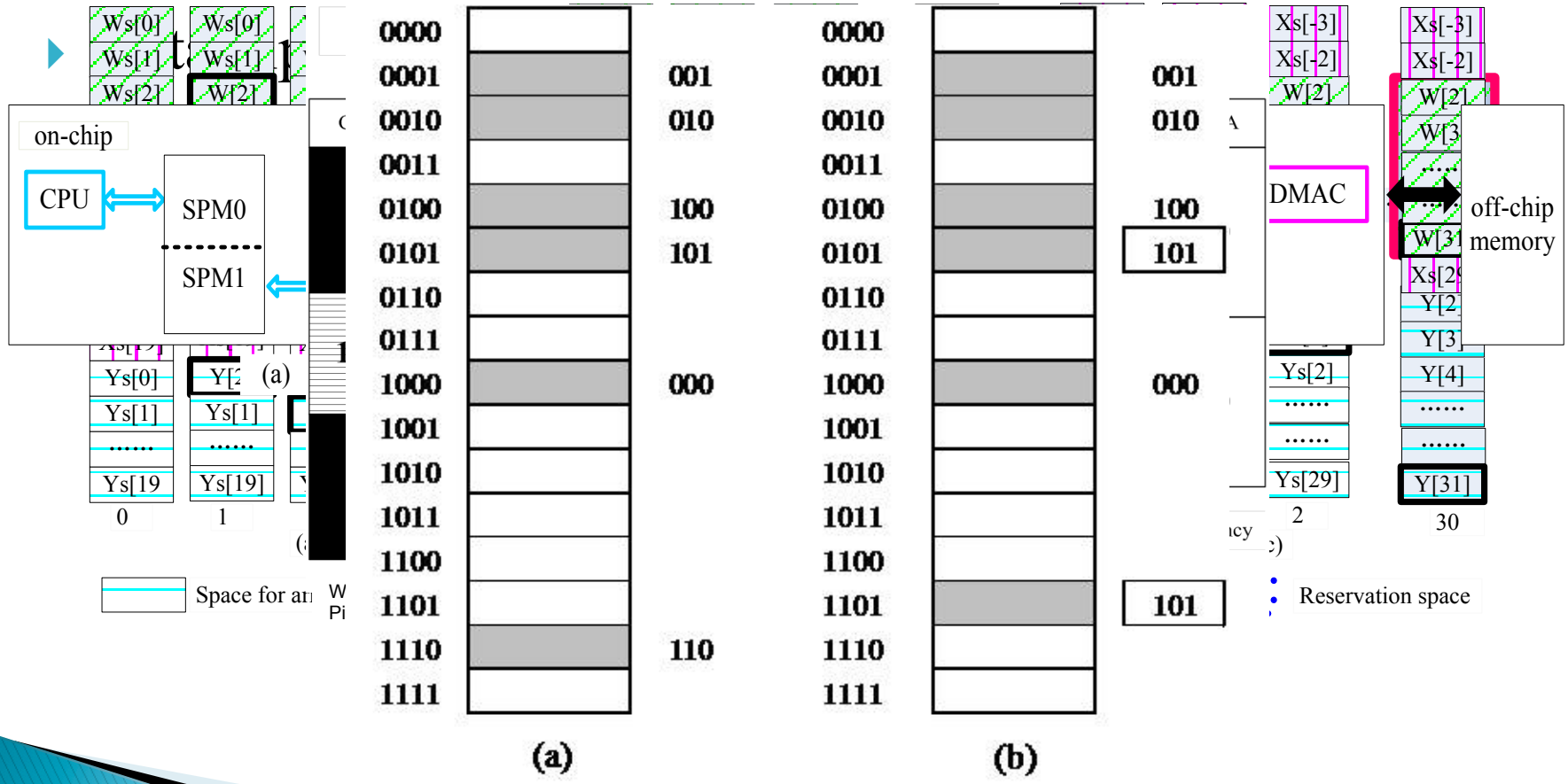
Programming Model Overview



bi-tier task-stealing

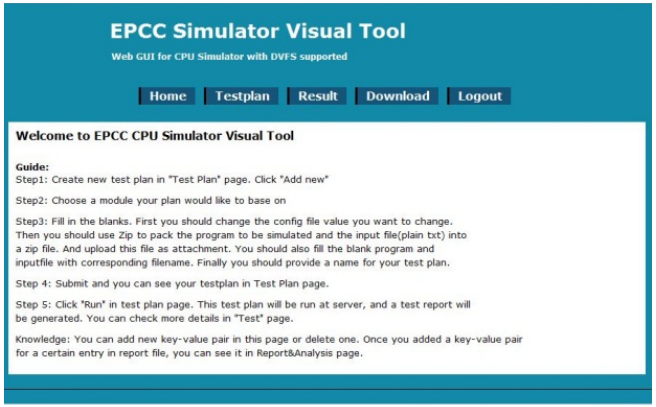
Compiler optimization and low-power-cost method

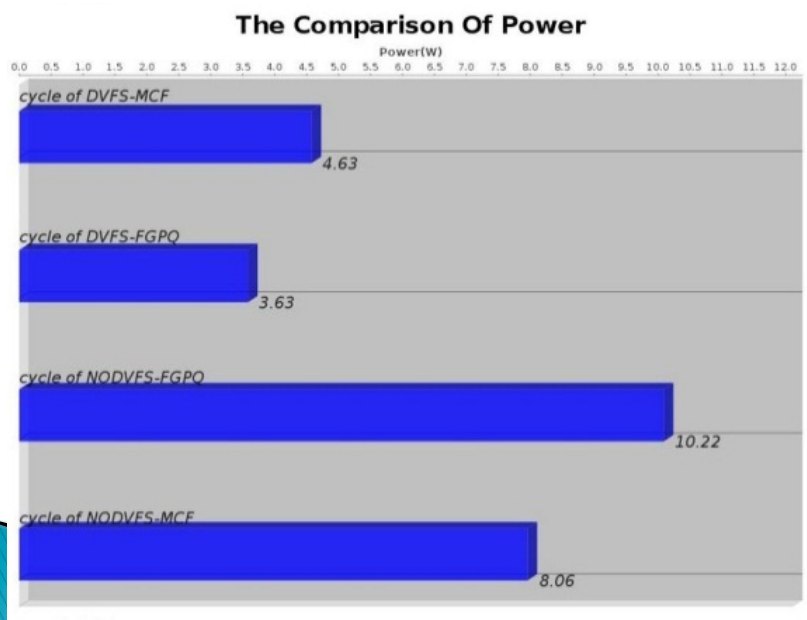
- Space overlapping model



Multi-core simulation tools

- http://epcc.sjtu.edu.cn/EPCC_SESC.htm
- Add DVFS instructions
- Calculate DVFS power using Watch model
- Provide three Benchmark examples
- Web-based visualization, user-friendly



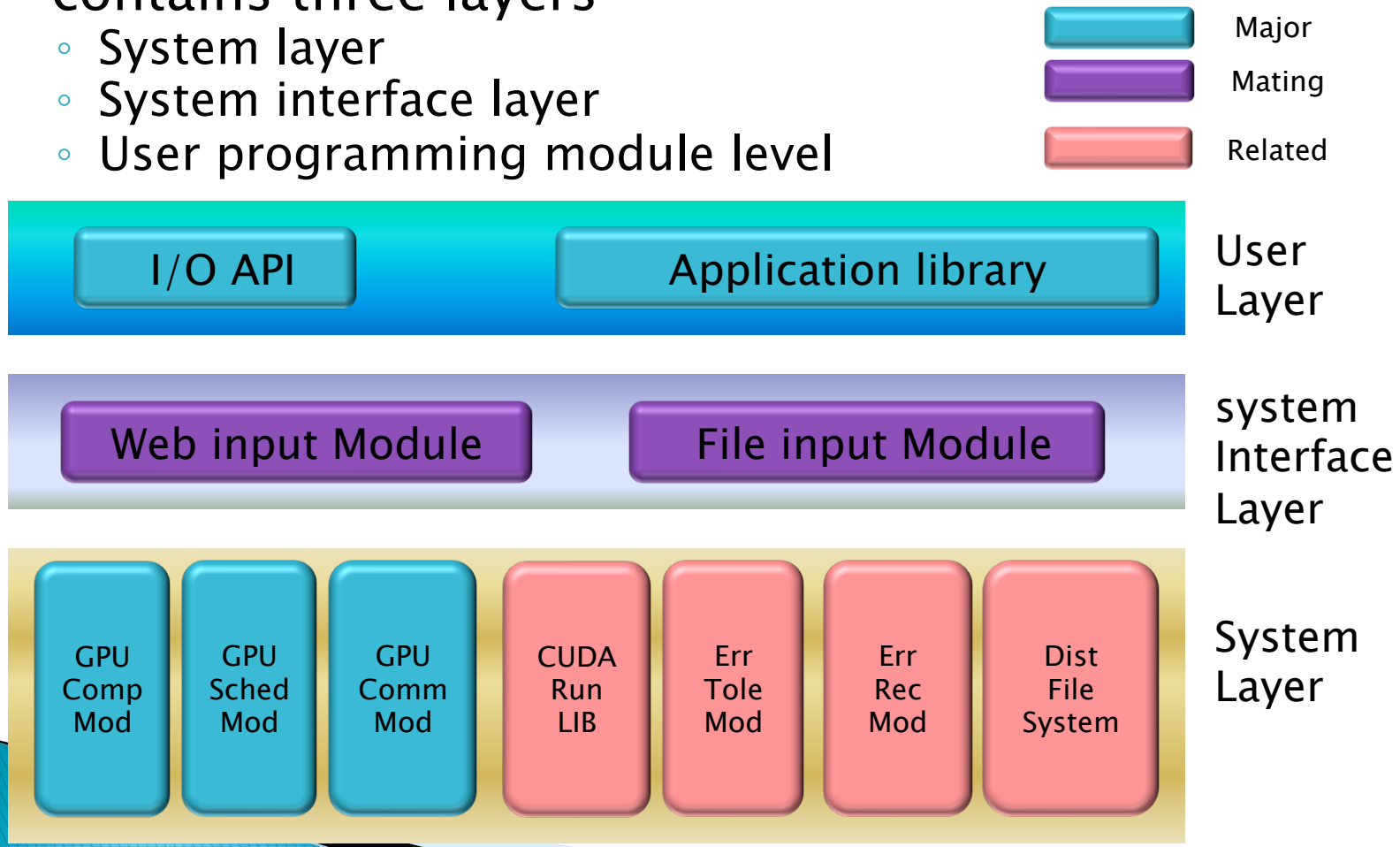


Distributed Architecture for multi CPU & GPU cluster

- GPGPU computing becomes more and more popular, traditional distributed systems can no longer ignore the computing power of GPUs.
- GPU's basic computing model is SIMD, which makes it very suitable for data intensive, loosely coupled computing.
- Distributed system architectures still need much users' attention when using GPUs for computing. Many problems including scalability, easy user interface, hardware and software configuration, etc. need to be solved.

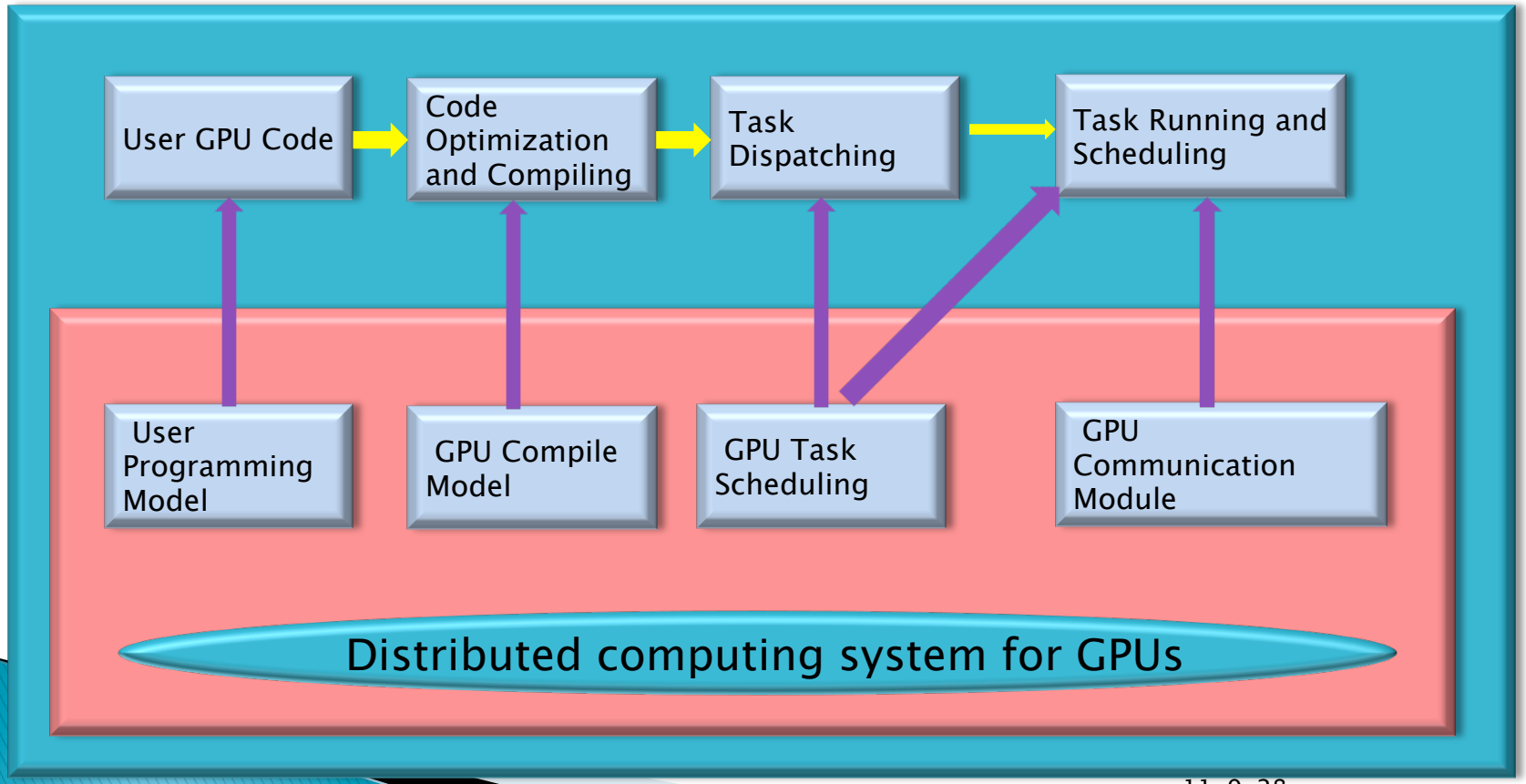
Distributed GPU parallel computing system

- The architecture of distributed GPU system contains three layers
  - System layer
  - System interface layer
  - User programming module level



Flow chat of System Layer

- Provide compiler and optimization for GPU, task distribution, task scheduling and task return



Representative Research Groups (2/5)

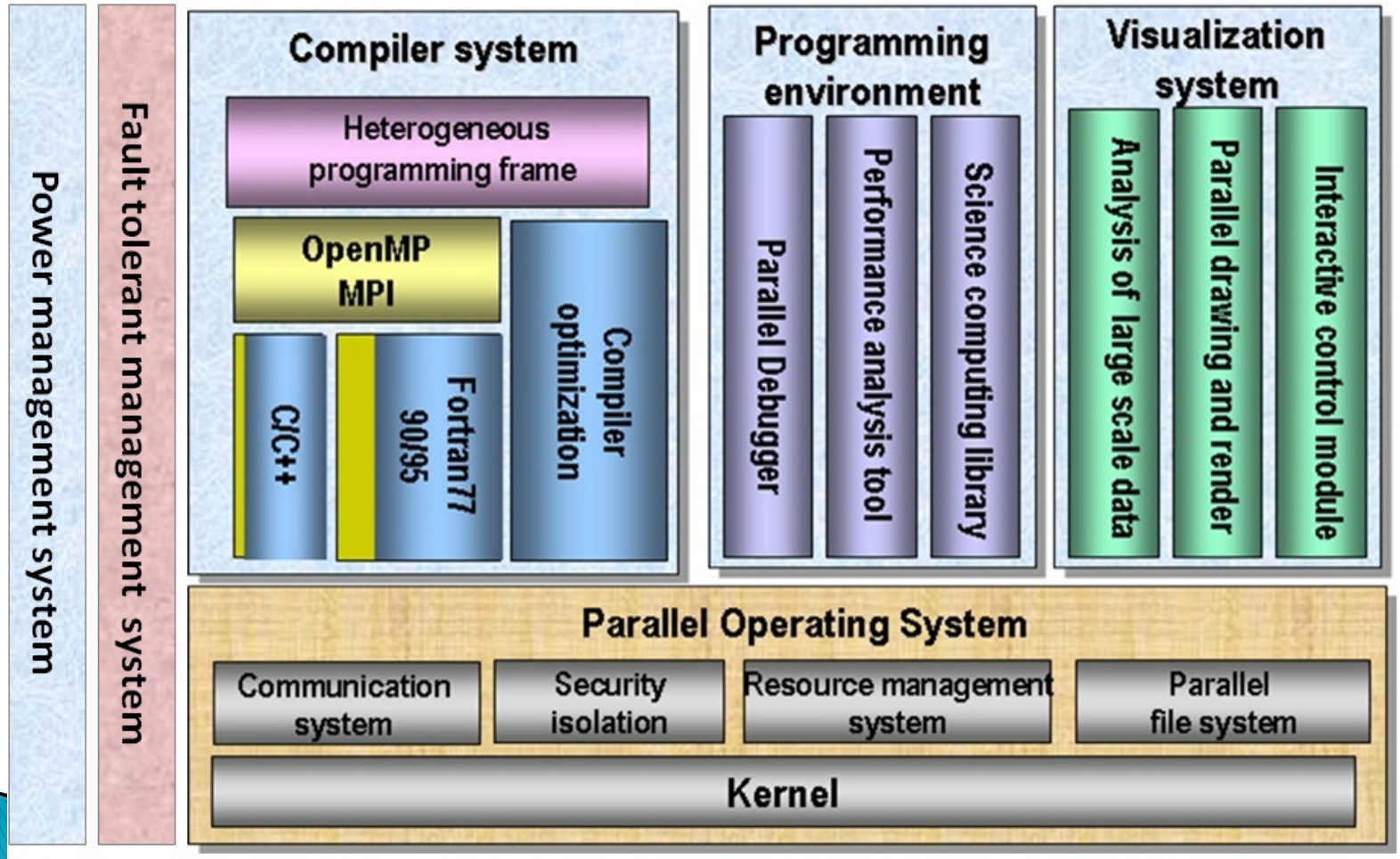
National University of Defense Technology

- Focus: High-performance computer system software architecture
- Typical project

  Key technology of petascale computing system
- Website: http://www.nudt.edu.cn
- Main publications:
  - IEEE TC, TPDA, TACO, ISCA, PACT, PPOPP, ICDCS, IPDPS, Hoti, Science in China, JCST…
- Scientific Research Directions
  - High Performance Computing
  - Micro-processors
  - System Software
  - Network and communications

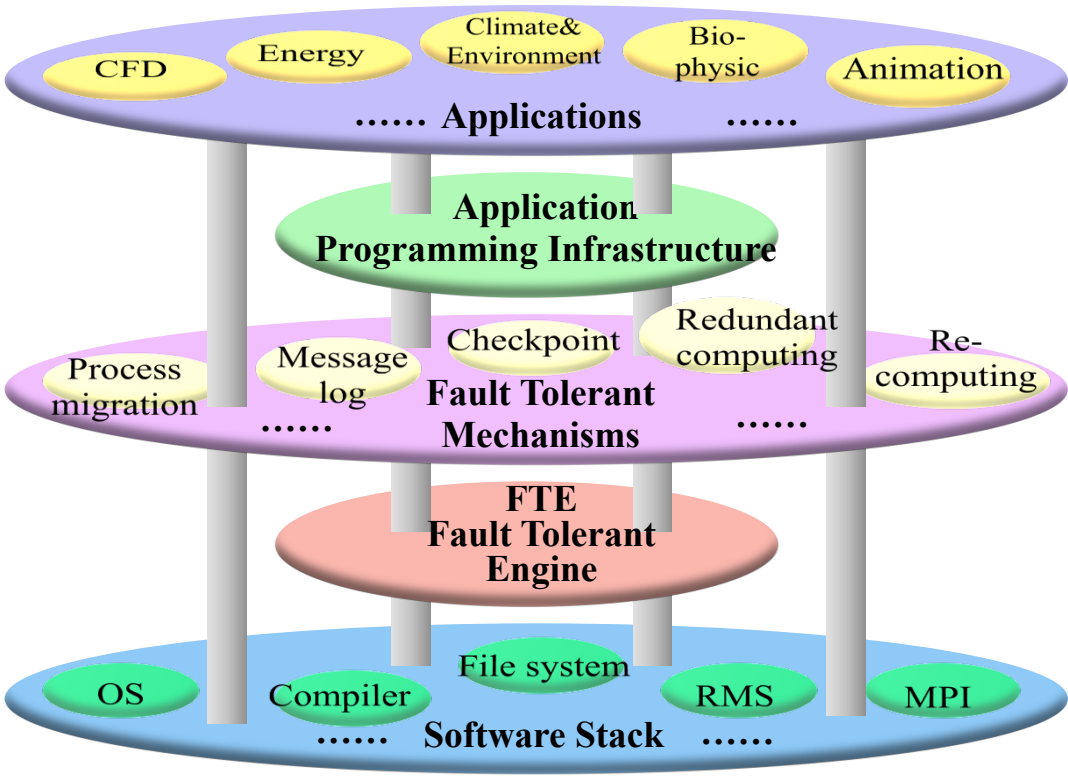
Some mature technology and application

HPC system software stack



Resilience computing Framework

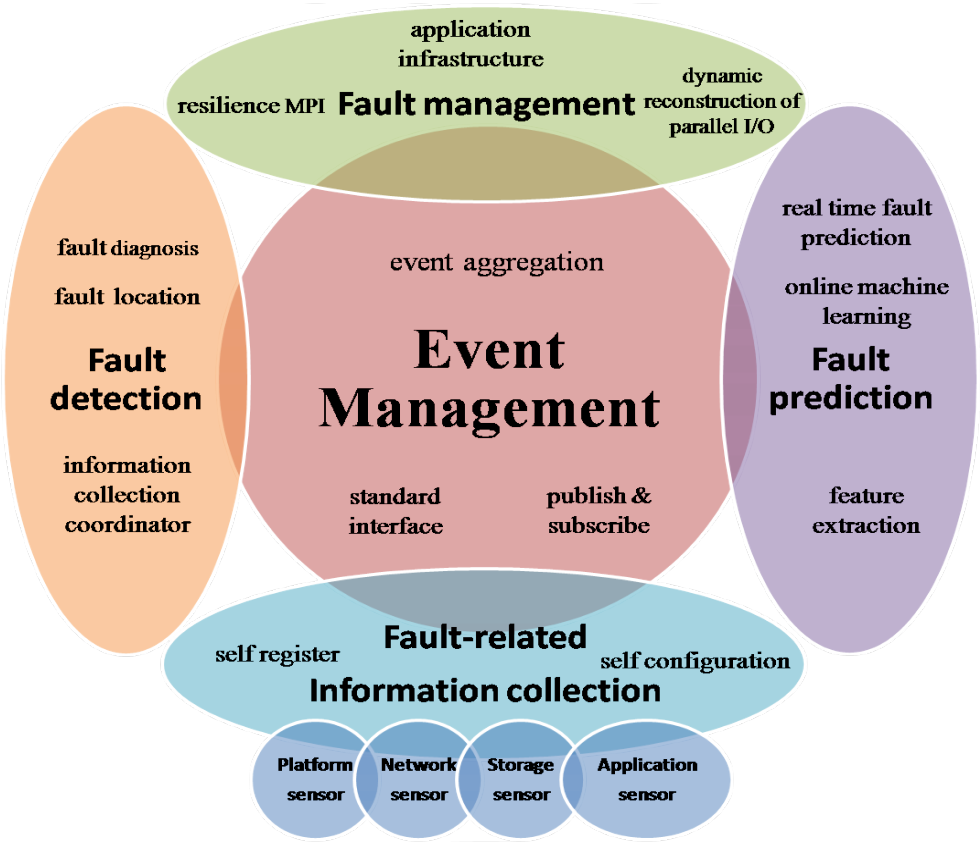
- Capability to support post-petaflops and Exascale computing
- Collaboration with whole system software stack
- Coherent fault detection
- Coordinate fault tolerant decision
- Cooperation of multiple fault recovery mechanics
- Combination of proactive and reactive strategies
- Customizable fault detection, prediction and recovery approaches
- Support various parallel models



Resilience computing Framework

Fault tolerant engine

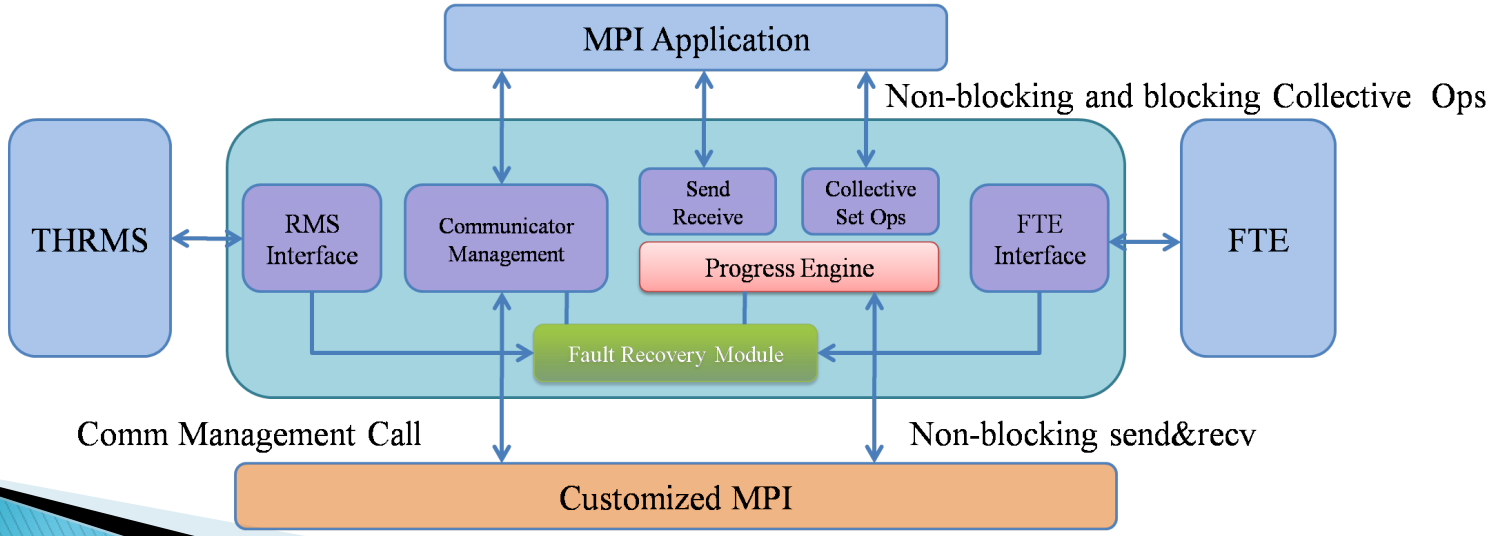
- Fault related information collection based on sensors
- Fault detection based on rules
- Fault prediction based on machine learning
- Event management based on publishing and subscribing
- Scalable and low-overhead fault-related information sharing and distribution
- Coherent fault detection and fault location
- Online learning and real time fault prediction



Resilience computing Framework

NR-MPI

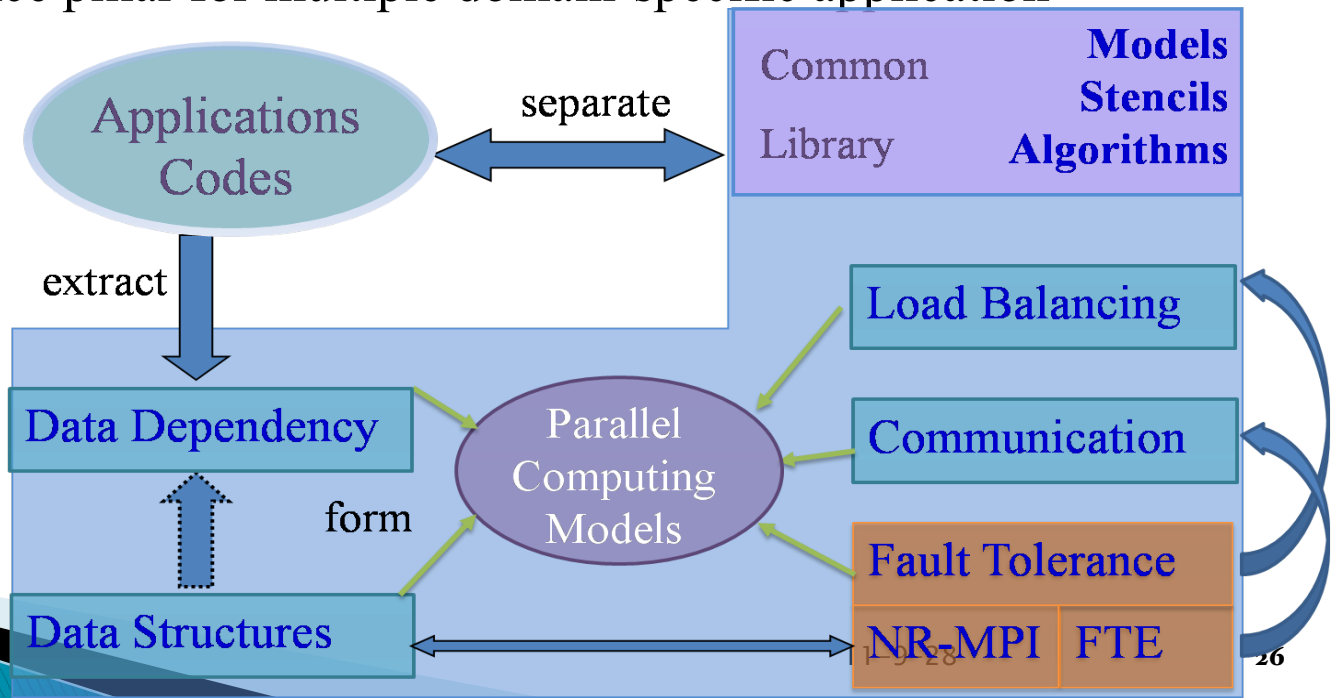
- Integrated with existing MPI (MPICH/OpenMPI)
- Relay on FTE providing failure information of nodes, processes, and network
- Reconstruction of broken communicator
- Recovery of ranks' connections
- Provide non-blocking fault tolerant collective operations
- Provide flexible data recovery mechanisms, CR/Redundant computing/APP



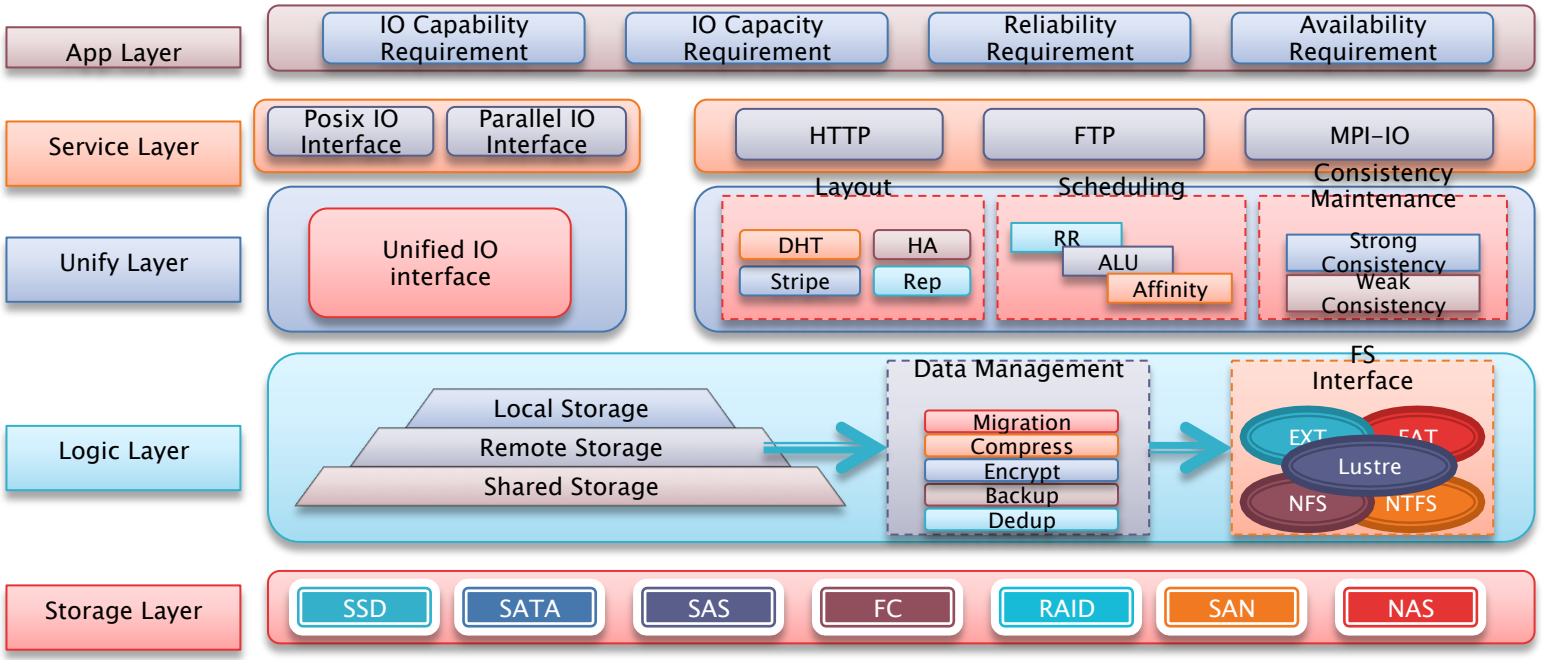
Resilience computing Framework

Application transparent Resilience Computing

- Domain-specific Application programming Infrastructure
  - Domain scientists: focus on physical problem, algorithm, parameters
  - App-Infrastructure: handles parallelization, message passing and resilience
- Application transparent Resilience Computing
  - Now for Jasmin
- Potential fault tolerance pillar for multiple domain-specific application Infrastructures
  - Future for others

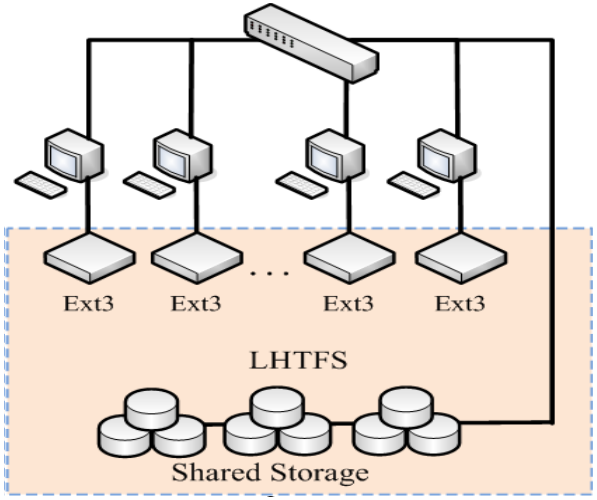


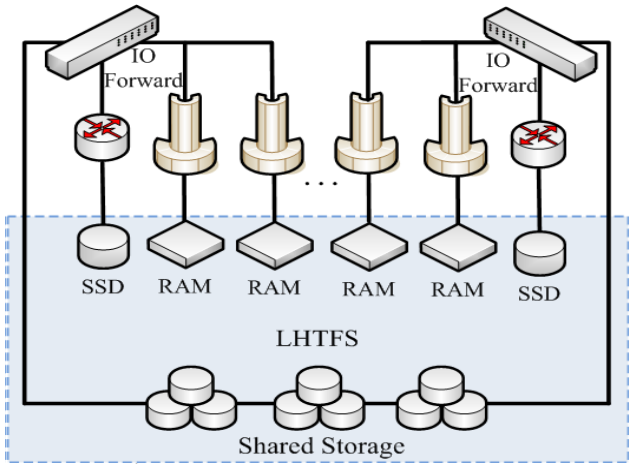
Large-scale Hybrid Tiered File System Architecture

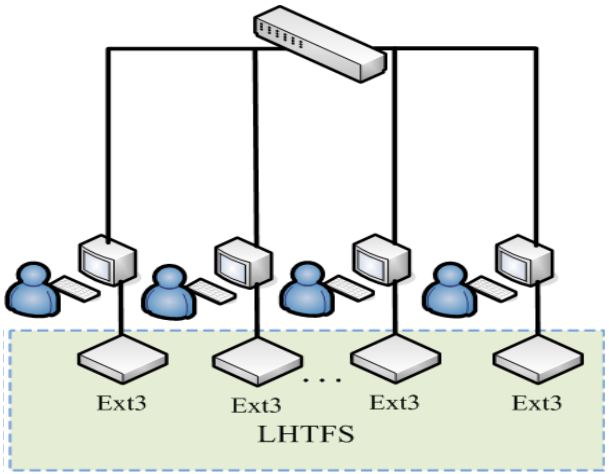


- Capability to support post-petaflops and Exascale computing
- Scalability to achieve >1TB/s I/O bandwidth by leveraging spatial locality
- Applicability for supercomputers and clusters with hybrid infrastructure
- Usability by federating multi-level storage into unified name space
- Flexibility by key components re-configuration for application optimization

Large-scale Hybrid Tiered File System Architecture







- Supports multiple tiers of storage devices and various configurations
- Exploits the features of local storage to improve the overall I/O performance in the case that both local storage and shared storage exist in HPC system
- Achieves IO forwarding by modifying node settings in the complex system configuration
- Acts as a normal distributed File System in the system without shared storage

Large-scale Hybrid Tiered File System Architecture

Key Technologies

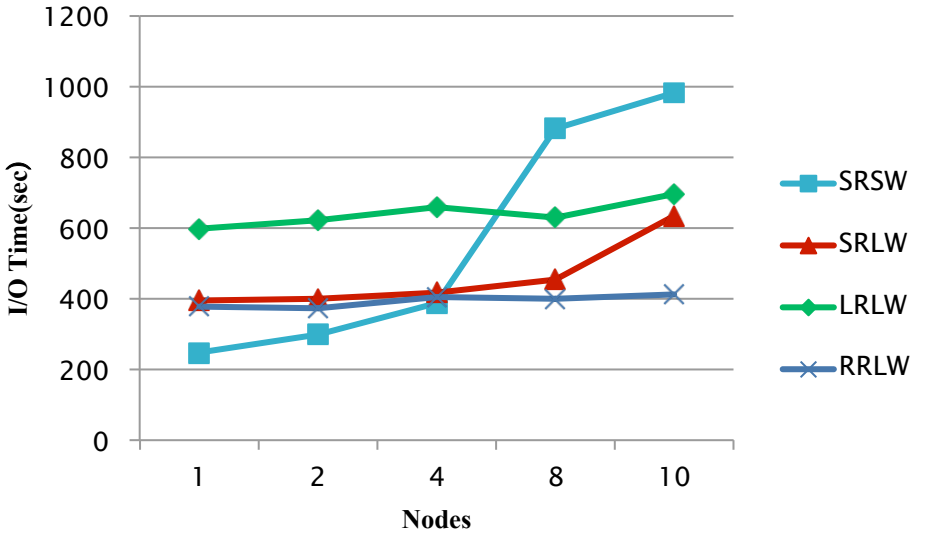
- Affinity scheduling to explore full potential of spatial locality
- Combination of centralized and decentralized metadata management
- Zoning and connecting on demand to reduce resource contention
- Relaxed data consistency control mechanism
- Vertical optimization through I/O path
- Redundant data management
- Smart data migration across storage levels

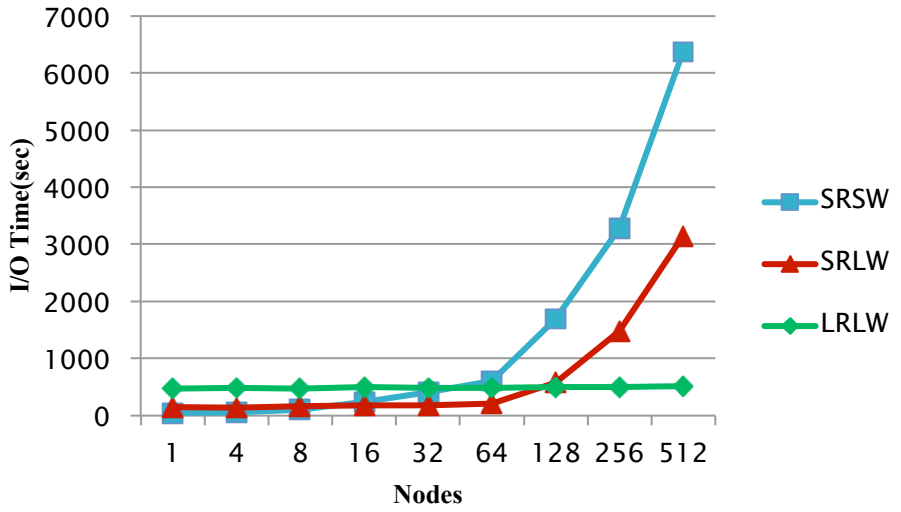
API

- POSIX-compliant UNIX file system interface for ease of use
- Custom API to boost app I/O access with weak consistency
- SDK for third-part interface support

Large-scale Hybrid Tiered File System Architecture

Experimental Results





- Petroleum seismism data analysis: Typical DISC Application
- LHTFS used, Double tiers(Local disks + shared THFS)
- SRLW can reduce the overall IO time by around 50% than traditional SRSW
- Scalable I/O capability, LRLW I/O time is almost a constant

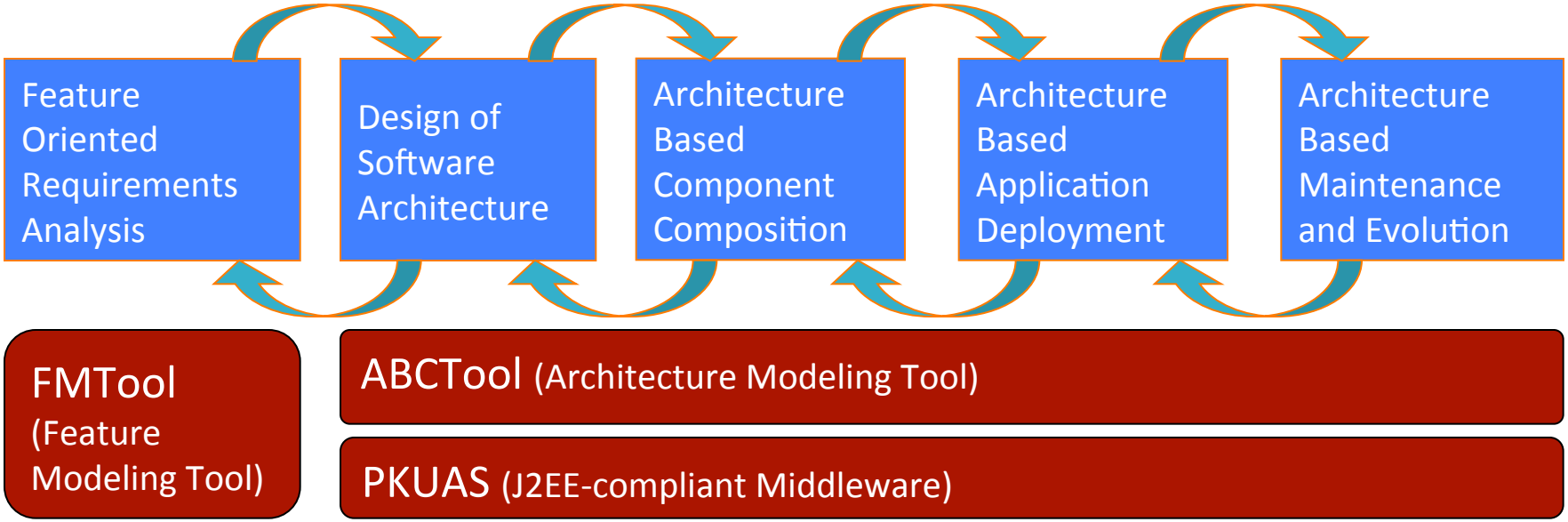
Representative Research Groups (3/5)

Peking University

- Focus: Architecture-Based Composition
- Typical project
  - Theory and methodology of agent-based middleware on Internet Platform
  - Research on Networked Complex Software: Quality and Confidence Assurance, Development Method, and Runtime Mechanism
- Website: http://sei.pku.edu.cn
- Main publications:

  ICSE, FSE, ICWS, COMPSAC, CBSE, IEEE TSC, ASE, JSS, IJSEKE, JSM, SSM, Science in China, JCST…

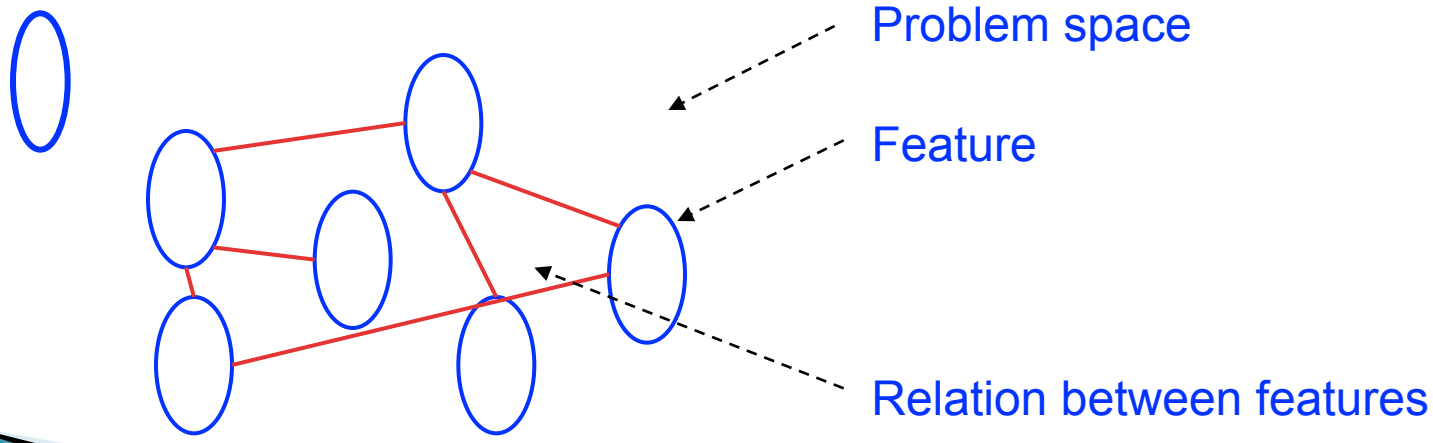
ABC Technical Framework



- Proposed in 1998
  - Introduces software architectures into each phase of software life cycle
  - Aims for automated component based reuse
- Evolved to Internetware from 2002
  - Supports openness (extensible ADL), dynamism (runtime software architecture), changeability (self-adaptive architecture) and autonomy (autonomous component)

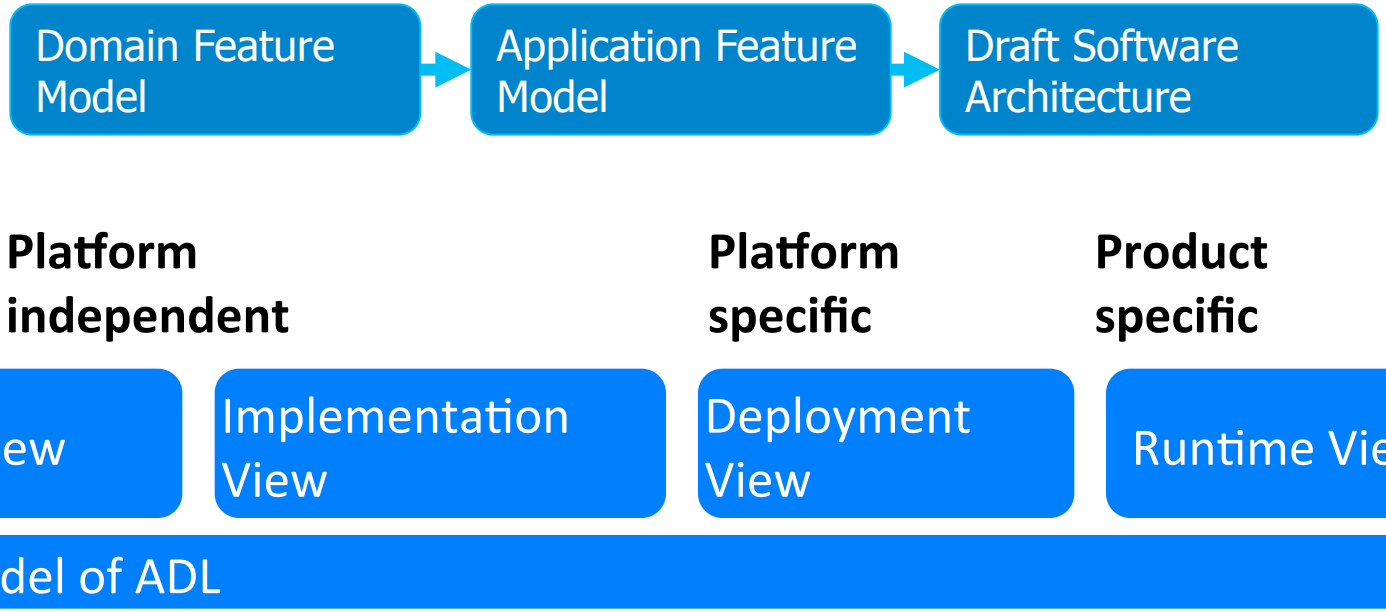
Feature Oriented Modeling

- Treats features as the basic entities in the problem space.
  - A feature describes a software characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements.
- Uses features and relations between features (feature model) to specify the problem space.



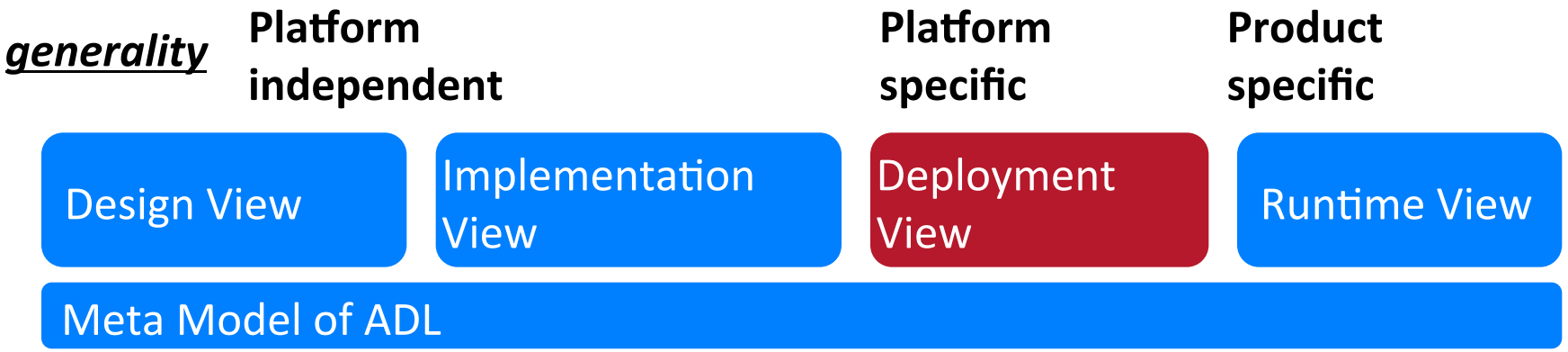
Feature-oriented view of the problem space

Software Architecture Modeling



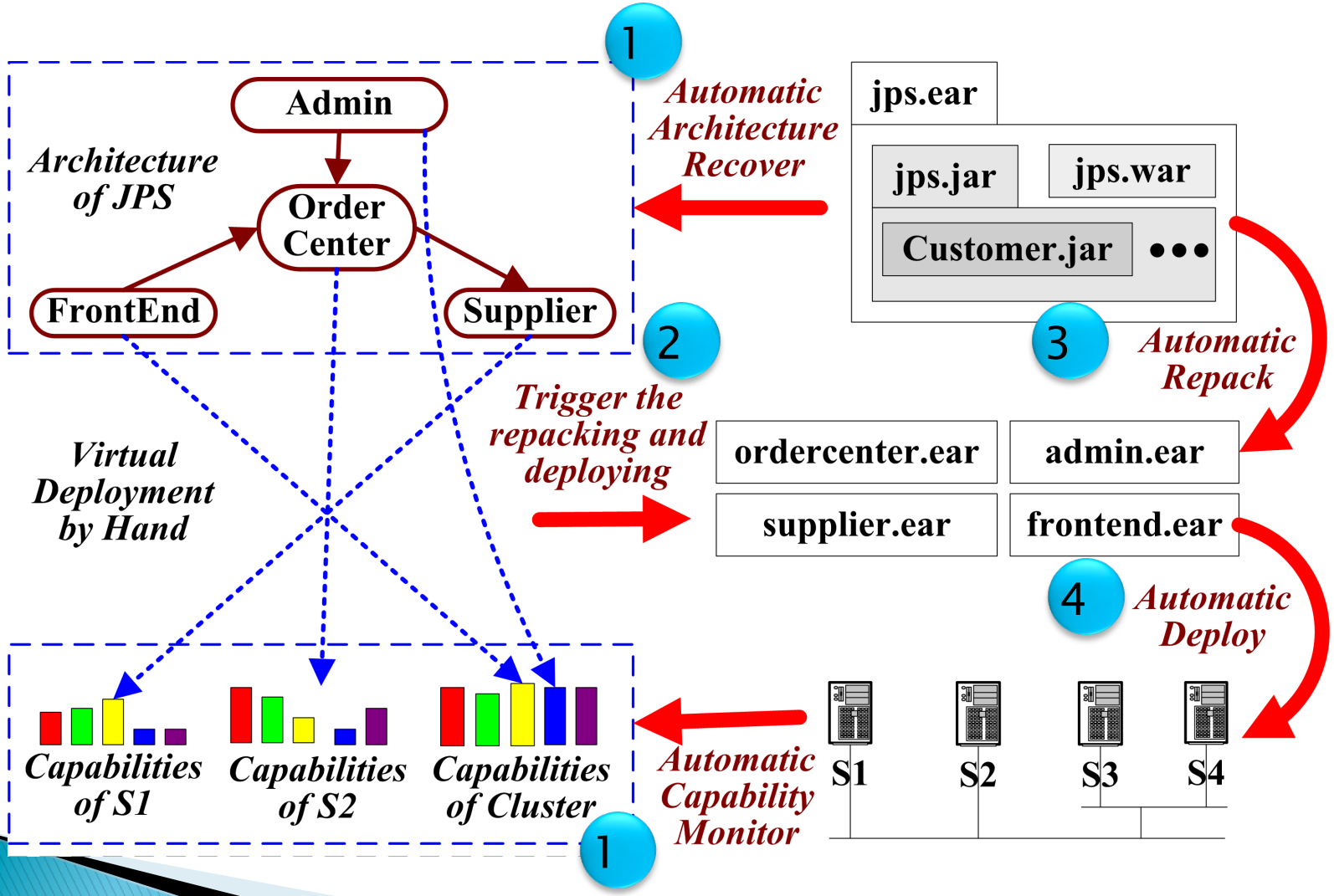
- The meta model of ADL only defines the elements common to all views and necessary for traceability

Architecture based Deployment

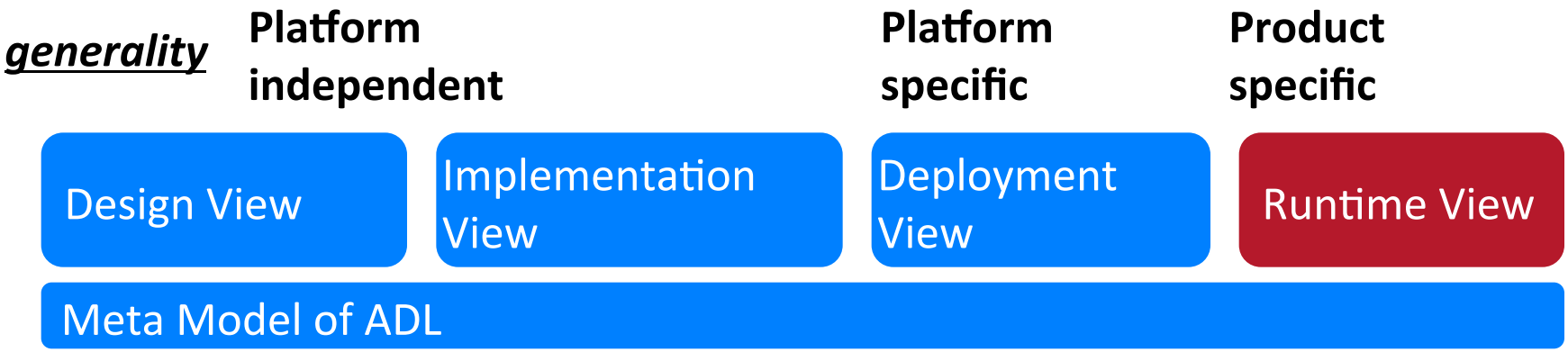


- Online Maintenance and Evolution
  - very valuable for large-scale distributed systems and 7x24 (7 days 24 hours) high availability
  - But very challenging to why, when, what & how
- ABC's Runtime Software Architecture Leverages
  - Software architecture knowledge for why, when & what
  - Middleware adaptability for how

Architecture based Deployment Process



Architecture based Maintenance and Evolution

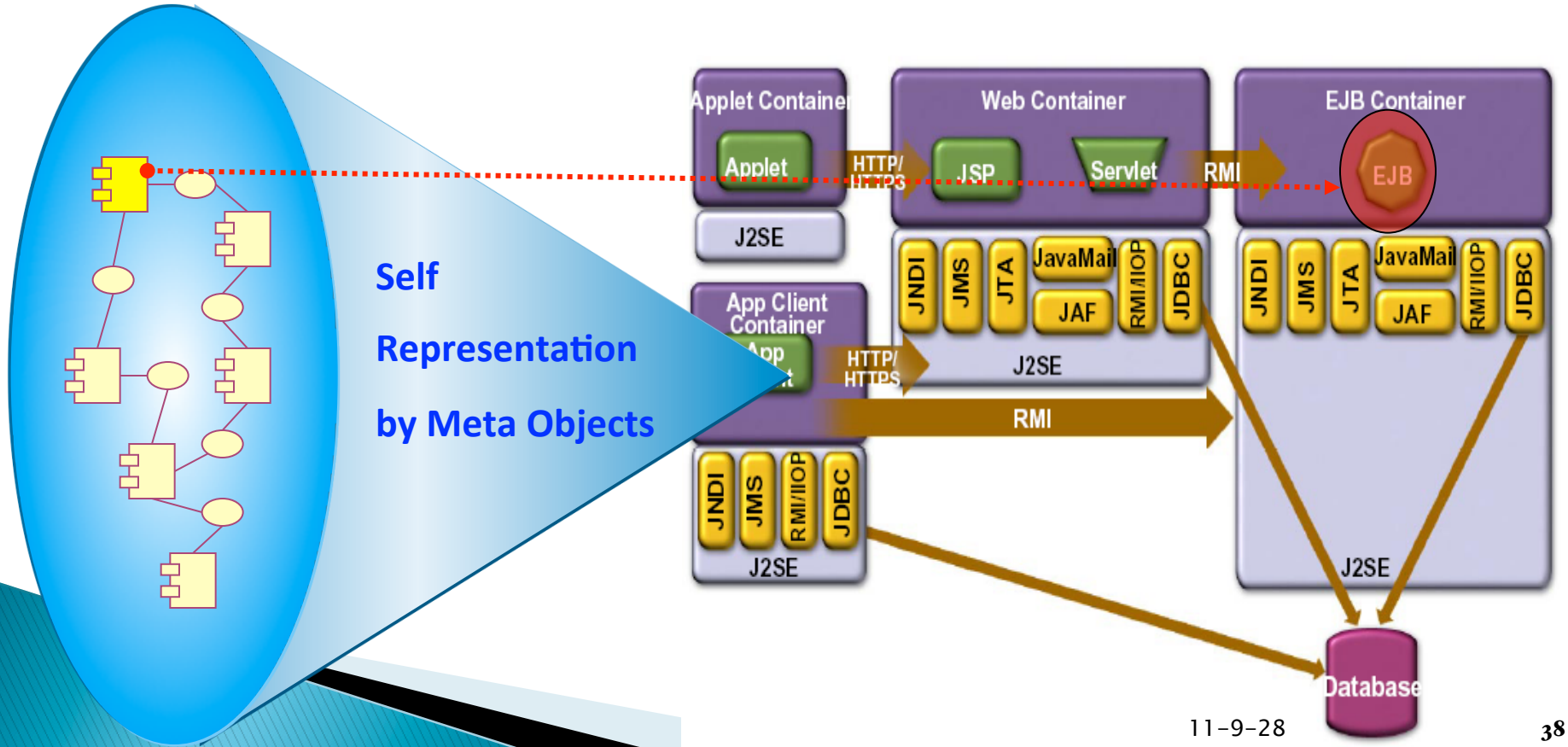


- Online Maintenance and Evolution
  - very valuable for large-scale distributed systems and 7x24 (7 days 24 hours) high availability
  - But very challenging to why, when, what & how
- ABC's Runtime Software Architecture Leverages
  - Software architecture knowledge for why, when & what
  - Middleware adaptability for how

Runtime Software Architecture

Runtime Software Architecture

- A model representing a runtime system as a set of architectural elements which are causally connected with the internal states and behaviors of the runtime system
  - Causal connection means changes at one side will immediately lead to the corresponding changes at the other side, and vice versa



Architecture based Reflective Framework

Reflective Programs

Reflective GUI

Reflective APIs

Correctness and Security of Reflection

AppSA Specific Meta Entities

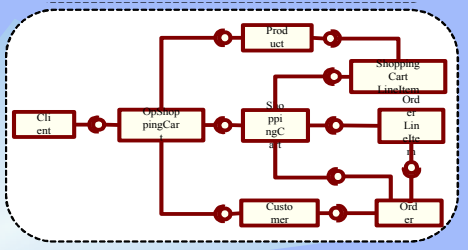
PlaSA Specific Meta Entities

Base Entities

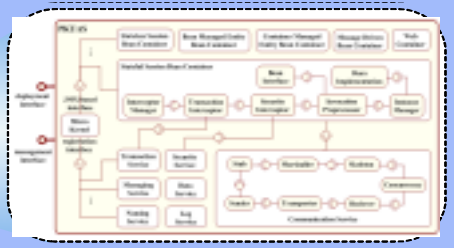
Causal Connection

Causal Connection

Reflective Middleware Based System



Application SA



Platform SA

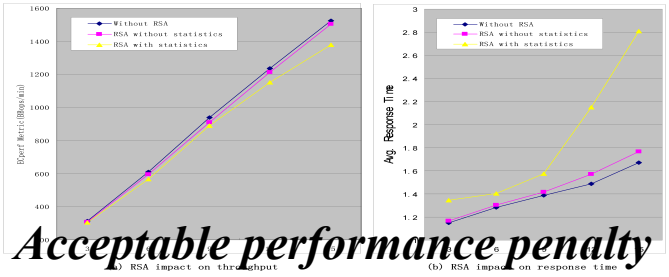
Formal Description of SA

SA in Lifecycle

Refine

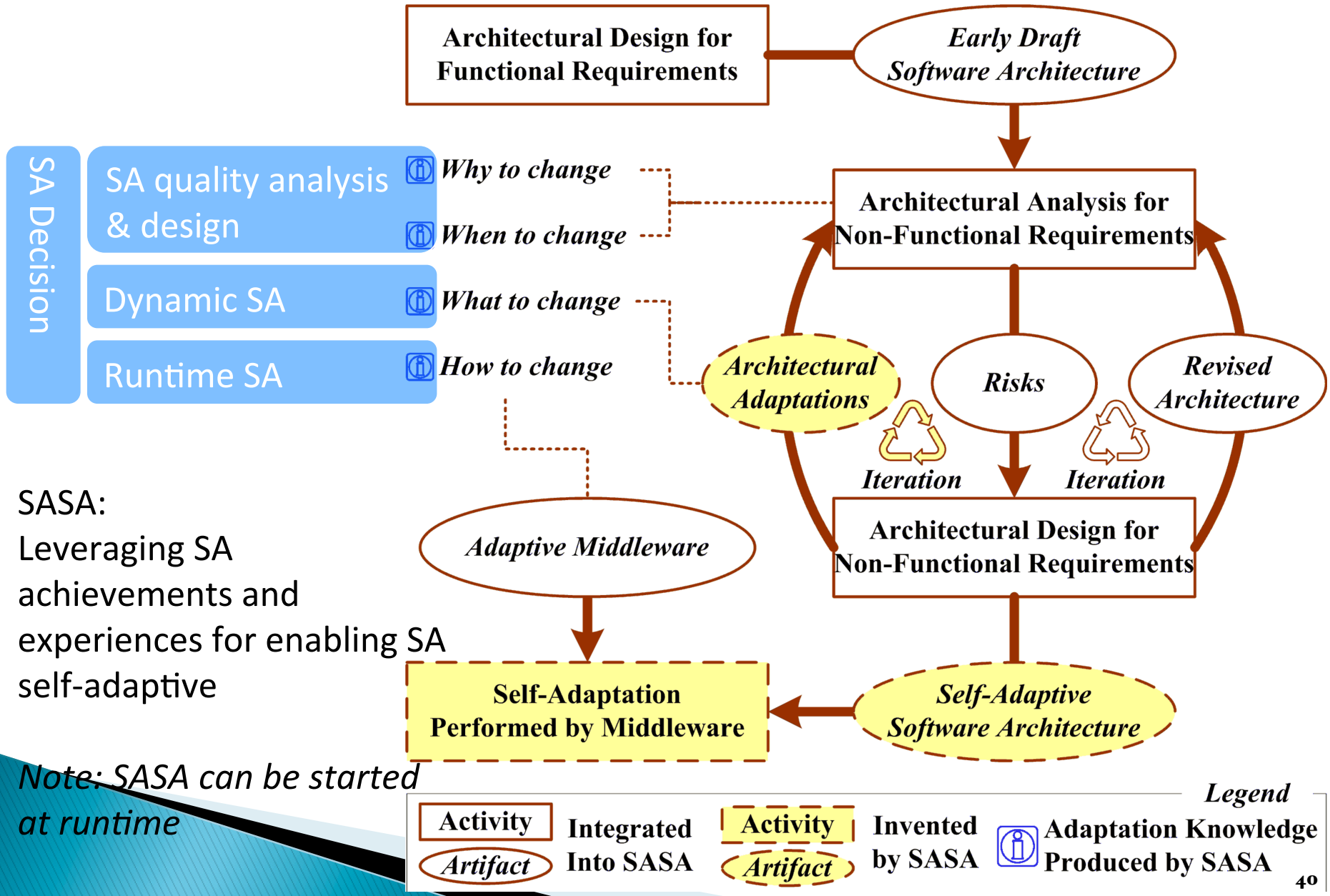
Deploy

SA in Development



Acceptable performance penalty

Self-adaptive Software Architecture

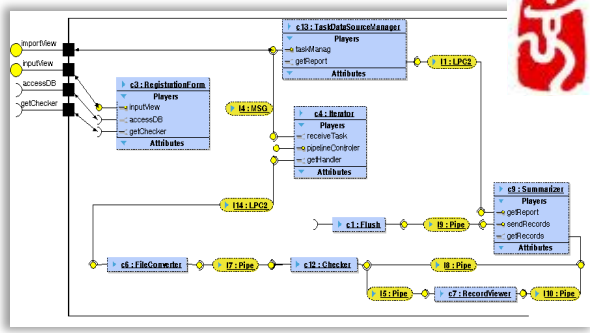


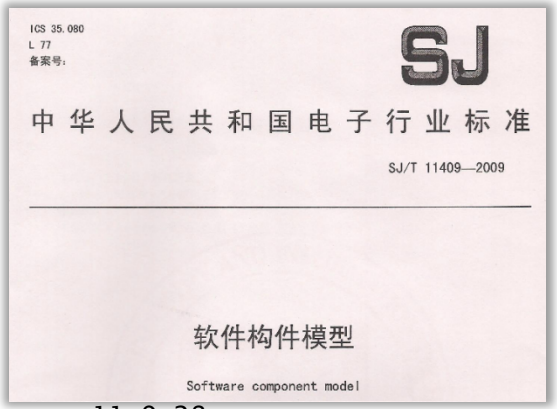
SASA:
Leveraging SA achievements and experiences for enabling SA self-adaptive

Note: SASA can be started at runtime

ABC Dissemination

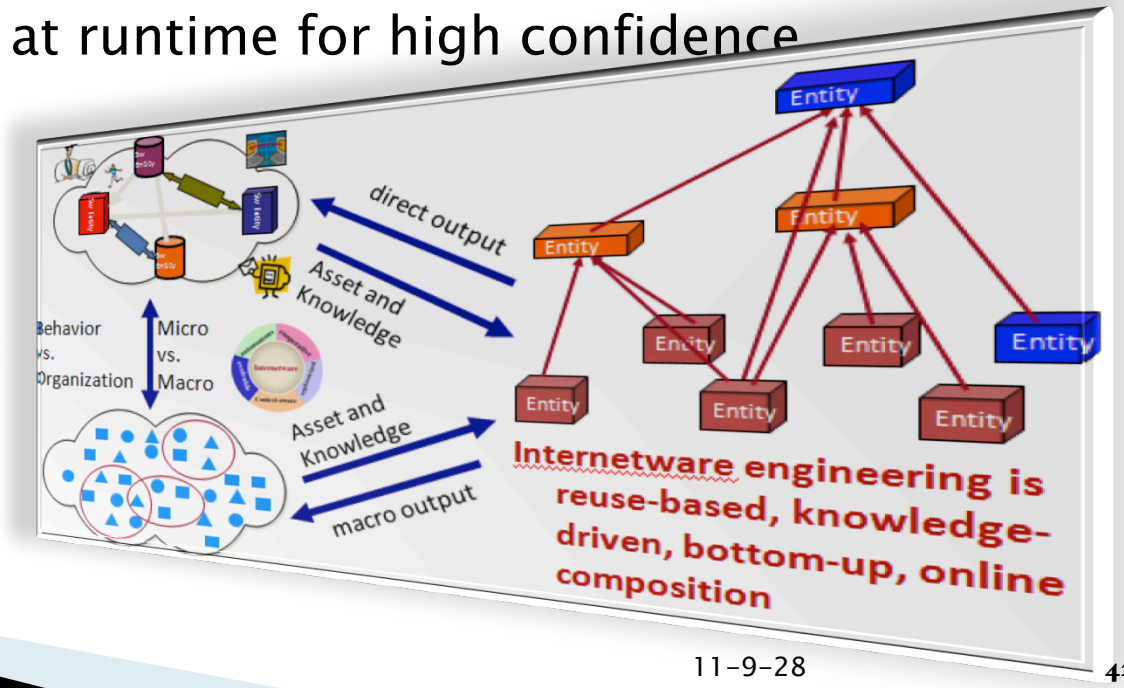
- Real Applications
  - Information System Modeling of Beijing Olympic Games 2008
  - Credit Management Modeling of China XXX Bank
- Training & Education
  - Reuse Oriented Requirements Modeling (book)
  - Design and Implementation of Component-based Systems (book)
  - Solution Engineering (courseware with IBM)
- Standardization
  - Leading Componentization Standard Series of China IT Industry from 2005
  - Release Software Component Model Specification in 2009





ABC Future

- Open Source as Eclipse Plug-ins
  - An extensible and customizable integrated engineering environment
- Support Internetware
  - A new software paradigm of Internet as a Computer
  - Design decision in architecting
  - Software architecture documentation with rich knowledge
  - Software architecture at runtime for high confidence



Representative Research Groups (4/5)

Fudan University

- Focus: adaptive architecture
- Typical project
  - Research of Confidence requirement model based adaptive fault-tolerant software architecture
- Website: http://www.se.fudan.edu.cn
- Main publications:

ICSE, ICSR, QSIC, RE, SEKE, COMPSAC, CSMR, Journal of Software, ….

Self-Adaptive Architecture

- Self-Adaptive Systems: systems that can adapt their structures and behaviors at runtime for
  - changing requirements
  - changing environments
- Architecture-based Self-Adaptation
  - reconfigurable architecture as the basis of adaptation decision making and execution
    - architecture model as knowledge base for adaptation decision
    - architectural reflection keeps the causal connection between meta-level (model) and base-level architecture (runtime architecture)
  - Separation of Concern: business logic and adaptation
  - flexible adaptation rules

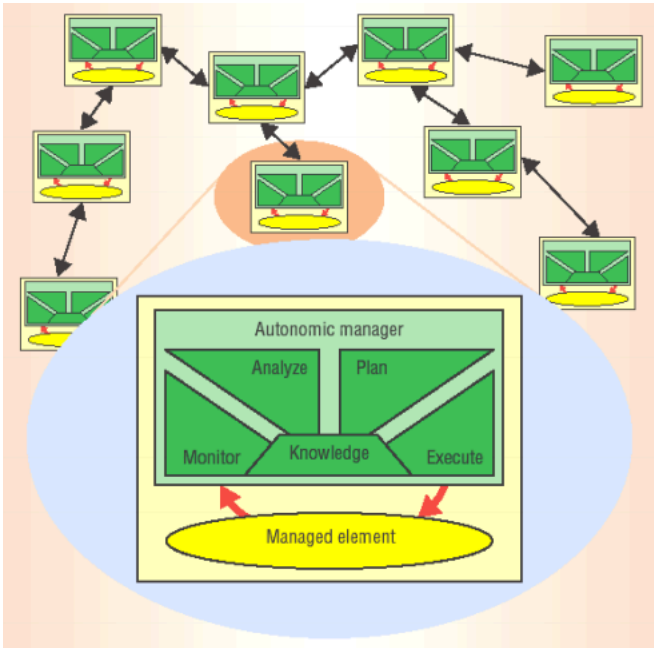
Self-Management and Control Structure of Self-Adaptive Systems

- Self-*: *systems shall managing themselves.*
  - Self-tuning – performance
  - Self-configuring – flexibility
  - Self-healing – dependability
  - Self-protecting – security/privacy

MAPE Control Loop for Self-Adaptive System

Monitoring
Analyzing +
Planning
Execution

Sensing
Actuating + ***knowledge***



Self-Adaptive Architecture

The vision of autonomic computing (Kephart and Chess, 2003)

MAPE Control Loop for Self-Adaptive Architecture

- Monitor
  - collect architecture-level events and parameters
- Analysis
  - aggregate and reason about events/parameters
- Plan
  - generate reconfiguration plans based on adaptation rules
- Execution
  - architecture reconfiguration

Component-based Middleware for Self-Adaptive Architecture

Architecture Reconfiguration

- Component replacement
  - with similar or identical interfaces
  - dynamic service selection and composition in service-oriented systems
- Structure reconfiguration
  - add/remove components
  - add/remove links between components/connectors
- Behavior reconfiguration
  - behavior of individual components
  - interaction behavior among components

Representative Research Groups (5/5)

Nanjing University

- Focus: dynamically reconfigurable systems
- Typical project

  ARTEMIS : Agent-oRiented TEchnology and Methodology for Internet Software
- Website: http://moon.nju.edu.cn/cgi-bin/twiki/view/ICSatNJU/ARTEMIS
- Main publications:

  FSE, ICWS, Science in China, Software: Practice & Experience, Journal of Computers, Journal of Software…

Smart wire

- Enriched component wires based on mobile agent technology
- Wiring logic
  - Group Table: Combination of available components.
  - Location Table: Locations of the components in the GT.
- The GT and LT are translated into a mobile agent.
- The components are wired up by the agent.

Multi-mode interaction

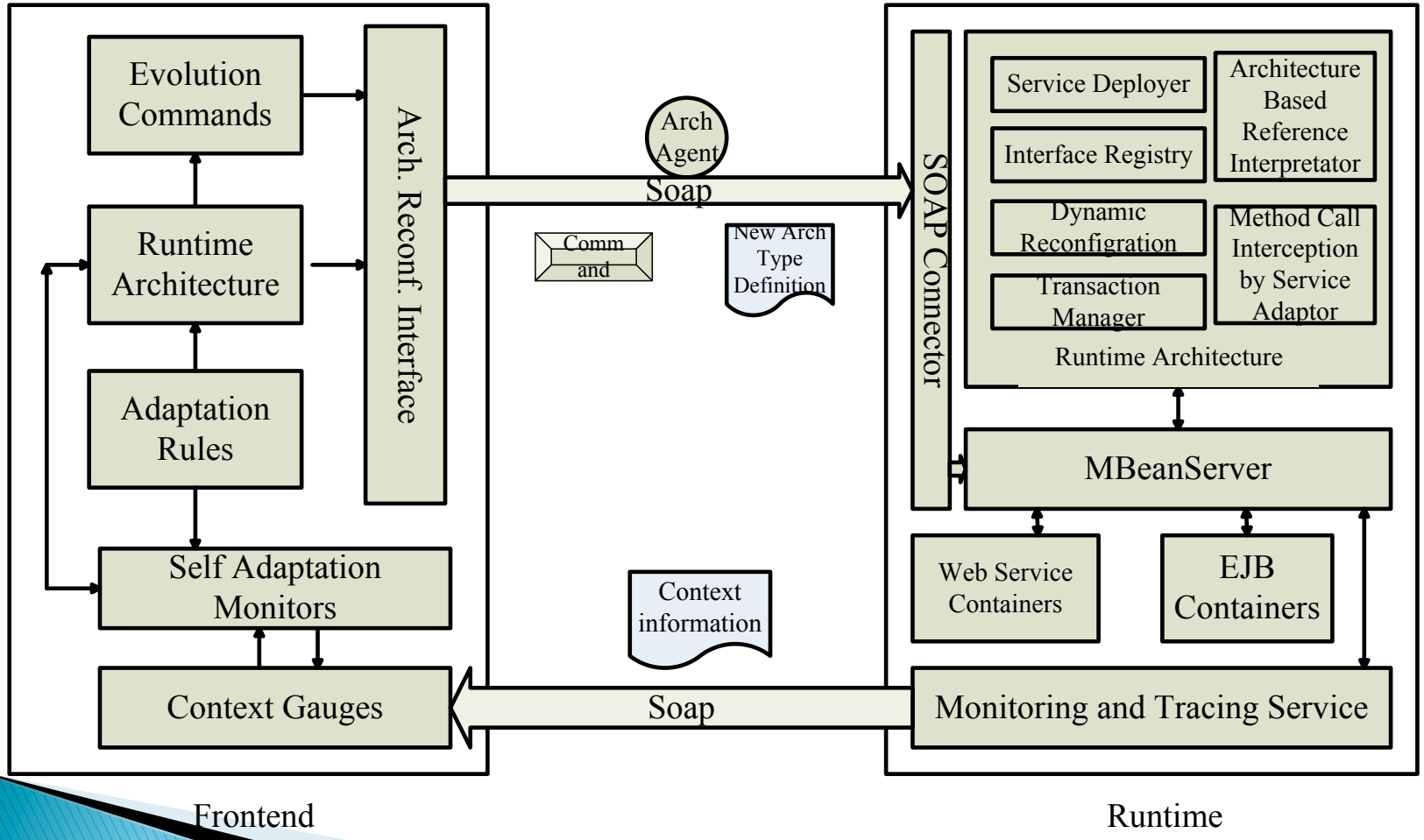
- "Factorize" the Interaction modes into basic interaction "factors"
- "Synthesize" the factors to form different interaction modes
- Artemis-M³C system:
  - Software service integration and interaction mode configuration
  - Runtime support for multi-mode interaction

Dynamic architecture

- Program the architecture as an distributed shared object.
- Express the architectural style with the type of the object;
- Constrain the architectural evolution with the type and inheritance mechanism of OOPL.

- Support the dynamic reconfiguration
  - With programmed behavior of the architecture object --- Planned
  - With polymorphic replacement of the architecture object --- Unplanned

System Structure



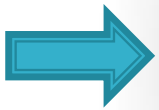
Agenda

- Introduction
- Founding Support
- Basic Research and Program
- Representative Research Groups
- Conclusion

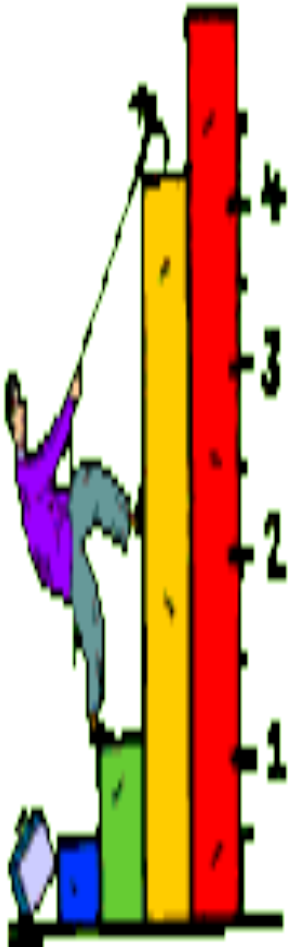
Main problems in China

- Several reasons caused many problems of software development in China

Custom Requirement	Theory Supervise
Complexity Increasing	Developer heterogeneity
Developer Instability	Scale Increasing



Challenges of Software



- Legacy system
  - reasonable cost to maintain and update the system
  - integrate the important business information and services in the system
- Heterogeneous system
  - Require new technology for development to run the developed software on different hardware platforms and system environments
- High confidence requirements for software development
  - How to ensure the high confidence for software development and running?
- Changing of software development
  - Research the architecture and development model of distributed software, explore the corresponding strategy of software engineering

Software research in China

- The NSFC major research project named ‘Basic Research on Trustworthy Software’ from 2007
- Research of major colleges and universities in the future

Thanks for your attention!!
Q & A