Internetware
Today and Future

Hong Mei

Key Laboratory of High Confidence Software Technologies (PKU), Ministry of Education

Beijing 2015.10
Agenda

• Research on Internetware

• Future of Internetware
At the early 21st century, the Internet is evolving to a global ubiquitous computer.

- Many big and hot trends in IT research and business tried to study such evolution from different perspectives.

**Technical Trend**

- Semantic Web
- Social Computing
- Service Computing
- System of Systems
- Pervasive Computing
- Grid/Cloud Computing
- Internet of Things

**Business Trend**

- Digital Economy
- E-government
- Internet Culture
- Social Network
- Modern Service
- Virtual World
- Smarter Planet
At the early 21\textsuperscript{st} century, the Internet is evolving to a global ubiquitous computer.

- Many big and hot trends in IT research and business tried to study such evolution from different perspectives.
At the early 21st century, the Internet is evolving to a global ubiquitous computer

- Many big and hot trends in IT research and business tried to study such evolution from different perspectives

**Technical Trend**
- Semantic Web
- Social Computing
- Service Computing
- System of Systems
- Pervasive Computing
- Grid/Cloud Computing
- Internet of Things

**Internet as a Computer**

**Trend?**

**Business Trend**
- Digital Economy
- E-government
- Internet Culture
- Social Network
- Modern Service
- Virtual World
- Smarter Planet

How about software perspective on Internet Computer?
Scenarios of Internet Computer (positive)

On Demand Cooperation

1. Contact CDC when suspected H1N1 infectors are detected
2. CDC acquires all customers in the same flight
3. CDC notifies all customers go to the given hotels
4. CDC sends customer info to the hotels
5. CDC notifies the hospitals go to the hotels and take case of the customers

H1N1 Control and Prevention
Scenarios of Internet Computer (positive)

On Demand Cooperation

1. Contact CDC when suspected H1N1 infectors are detected

2. CDC acquires all customers in the same flight

3. CDC notifies all customers go to the given hotels

4. CDC sends customer info to the hotels

5. CDC notifies the hospitals go to the hotels and take care of the customers

H1N1 Control and Prevention
Scenarios of Internet Computer (positive)

On Demand Cooperation
• happens anytime anywhere among anyone
• cooperators may have no or little relations before and after

1. Contact CDC when suspected H1N1 infectors are detected
2. CDC acquires all customers in the same flight
3. CDC notifies all customers go to the given hotels
4. CDC sends customer info to the hotels
5. CDC notifies the hospitals go to the hotels and take case of the customers

H1N1 Control and Prevention
Scenarios of Internet Computer (negative)

On Demand Cooperation

Angela’s Hotel At Shanghai 86-21-87359213

Calling Andy

Andy’s Home at Beijing 86-10-62536041

Call Forwarding: 86-10-62757670

Andy’s Company At Beijing 86-10-62757670
Scenarios of Internet Computer (negative)

On Demand Cooperation

Angela’s Hotel At Shanghai 86-21-87359213

Calling Andy

Andy’s Home at Beijing 86-10-62536041

Call Forwarding: 86-10-62757670

Andy’s Company At Beijing 86-10-62757670

Terminating Call Screening: 86-21-XXXXXXXX
On Demand Cooperation
• right time
• right place
• right ones

1) Feature Interaction:
Terminating the call? Or accepting the call?

2) Multiple widgets may compete for limited HTTP connections in a web browser

Terminating Call Screening:
86-21-XXXXXX

Call Forwarding:
86-10-62757670

Andy's Company At Beijing
86-10-62757670

Andy's Home at Beijing
86-10-62536041

Angela's Hotel At Shanghai
86-21-87359213

Calling Andy
On Demand Cooperation
- happens anytime anywhere among anyone
- cooperators may have no or little relations before and after.

Today’s popular software paradigms cannot enable the on demand cooperation in a natural and cost-effective way.
On Demand Cooperation
• happens anytime anywhere among anyone
• cooperators may have no or little relations before and after.

Today’s popular software paradigms cannot enable the on demand cooperation in a natural and cost-effective way.
On Demand Cooperation
- happens anytime anywhere among anyone
- cooperators may have no or little relations before and after.

Today’s popular software paradigms cannot enable the on demand cooperation in a natural and cost-effective way.

- Hardwired (fixed partner, fixed interaction)
- Loosely coupled (unfixed partners, fixed interaction)
- Flexible (fixed partners, unfixed interaction)
- Goal-driven (unfixed partners, unfixed interaction)
On Demand Cooperation
• happens anytime anywhere among anyone
• cooperators may have no or little relations before and after.

Today’s popular software paradigms cannot enable the on demand cooperation in a natural and cost-effective way.

No dominant or popular paradigms support on demand cooperation directly.
Cooperative: software can interact with others in static, dynamic and even on demand manners.

Emergent: software may have undesigned behaviors or unexpected effects on its runtime instances or interactions with others.
Software Characteristics for Internet (2)

**SEECAT features**

**Cooperative**: software can interact with others in static, dynamic and even on demand manners.

**Autonomous**: software is relatively independent of others; it can perform operations as it will and adapt itself when necessary.

**Emergent**: software may have undesigned behaviors or unexpected effects on its runtime instances or interactions with others.

**Situational**

**Evolvable**

**Trustworthy**
Software Characteristics for Internet (3)

SEECAT features

Situational: software is capable of perceiving its runtime context and scenarios

Cooperative: software can interact with others in static, dynamic and even on demand manners

Autonomous: software is relatively independent of others; it can perform operations as it will and adapt itself when necessary

Emergent: software may have undesigned behaviors or unexpected effects on its runtime instances or interactions with others

Situational Awareness

Past
Where was I?
What happened?

Present
Where am I?
What is happening?

Future
Where am I going?
What could happen?

A
utonomous: software is relatively independent of others; it can perform operations as it will and adapt itself when necessary.

C
ooperative: software can interact with others in static, dynamic and even on demand manners.

S
ituational: software is capable of perceiving its runtime context and scenarios.

E
volvable: software is easy to add, remove and change its functionalities on-the-fly and just-in-time.

E
mergent: software may have undesigned behaviors or un-expected effects on its runtime instance or interactions with others.
Software Characteristics for Internet (5)

**SEECAT features**

- **Situational**: software is capable of perceiving its runtime context and scenarios.
- **Evolvable**: software is easy to add, remove and change its functionalities on-the-fly and just-in-time.
- **Emergent**: software may have undesigned behaviors or unexpected effects on its runtime instances or interactions with others.
- **Cooperative**: software can interact with others in static, dynamic and even on demand manners.
- **Autonomous**: software is relatively independent of others; it can perform operations as it will and adapt itself when necessary.
- **Trustworthy**: software should promise some kind of tradeoff among process quality, internal system quality, external system quality and usage quality.
To support these characteristics, a new software paradigm is needed!

- **Structured Paradigm**: Distributed computing middleware / RPC
- **Object-Oriented Paradigm**: Object oriented middleware / CORBA
- **Component-Based Paradigm**: Component based middleware / J2EE
- **Service-Oriented Paradigm**: Service oriented middleware / Web Services
Internetware as a Software Paradigm

- Internetware is a new software paradigm for the Internet Computing
  - Architected like the Internet
  - Developed with the Internet
  - Executed on the Internet
  - Provided as services via the Internet

Internet as a Computer

Computer with Internet Access

Computer with Network Access

Computer without Network Access

- Structured Paradigm
  - Distributed computing middleware / RPC
- Object-Oriented Paradigm
  - Object oriented middleware / CORBA
- Component-Based Paradigm
  - Component based middleware / J2EE
- Service-Oriented Paradigm
  - Service oriented middleware / Web Services
Internetware as a Software Paradigm

• Internetware is a new software paradigm for the Internet Computing
  – Architected like the Internet
  – Developed with the Internet
  – Executed on the Internet
  – Provided as services via the Internet
Internetware: Behavior Mode

Observation and expectation on Internetware
**Internetware: Behavior Mode**

**Observation and expectation on Internetware**

**Trusted Construction/Cooperation On Demand**
- Entity comes and goes autonomously
- Connection changes from one protocol to another
**Self-organizing Domain/Community**

- Entities in cooperation may form an application domain or a community of interest.
- Communities may become big or small, merge or disappear.
Internetware: Engineering Mode

Observation and expectation on Internetware

Behavior vs. Organization
Micro vs. Macro

Asset and Knowledge

direct output

Asset and Knowledge

macro output
Internetware engineering is reuse-based, knowledge-driven, bottom-up, online composition.
Value of Internetware

Internetware brings new experiences and values to all stakeholders of software, i.e. developers, operators and users (through its distinguished characteristics)

Developers
• Program on physical and local libraries and resources as well as virtual and web-delivered ones
• Implement software could be virtualized and web-delivered naturally (then scalable and dependable)
Internetware brings new experiences and values to all stakeholders of software, i.e. developers, operators and users (through its distinguished characteristics)

Developers
• Program on physical and local libraries and resources as well as virtual and web-delivered ones
• Implement software could be virtualized and web-delivered naturally (then scalable and dependable)

Users
• enjoy software anywhere, anytime, any device in rich and consistent experience e.g. Yahoo Map running on
• IE Browser: unaware of local resources
• Adobe AIR: aware of local resources, e.g. emails
• iPhone: aware of local resources and user contexts, e.g. GPS info, temperature, light.
Value of Internetware

Internetware brings new experiences and values to all stakeholders of software, i.e. developers, operators and users (through its distinguished characteristics)

Developers
- Program on physical and local libraries and resources as well as virtual and web-delivered ones
- Implement software could be virtualized and web-delivered naturally (then scalable and dependable)

Users
- Enjoy software anywhere, anytime, any device in rich and consistent experience e.g. Yahoo Map running on
  - IE Browser: unaware of local resources
  - Adobe AIR: aware of local resources, e.g. emails
  - iPhone: aware of local resources and user contexts, e.g. GPS info, temperature, light.

Operators
- Collaborative management while leveraging the inherent management capabilities of virtualization
- Management capabilities and tasks are virtualized and web-delivered naturally (then scalable and dependable)
Challenges to Internetware

Programming Paradigm (what to be)
- abstracts the elements and their relationships of a software system
- Internetware model should
  - leverage legacy software and new characteristics
  - Enable open collaboration between components
  - Adapt itself for emergent contexts and situations

Engineering approach (how to make)
- Systematically control the software development, deployment, maintenance and evolution
- Internetware engineering should
  - Identify the self-organized communities and domains or facilitate the self-organizations
  - Satisfy requirements via collaborating existing and/or emergent components
  - Involve all stakeholders, especially the actual end users

Programming Language & System (how to be)
- incarnates the elements and their relationships of a software model
- Internetware middleware should
  - Provide a container for instantiating and operating Internetware components
  - Equip legacy software systems with Internetware characteristics
  - Make collaborations runtime entities
  - Enable context-awareness and reflection

Quality assurance (how to be good enough)
- Focal points of software quality change from system-centric to usage-centric
- Internetware quality assurance should
  - Define quantitative and qualitative evaluation framework for quality
  - Assure the quality via engineering approach at development time as well as middleware at runtime
Internetware Research in China

- Initially sponsored by 973
- Now by all national R&D programs

- Major National S&T Major Projects
- Key National Key Technology R&D Program
- National High Technology R&D Program
- National Basic Research Program
- National Natural Science Foundation

For social and Economical Development Technology R&D Fundamental Research
Internetware R&D Outputs

Internetware Engineering Approach

- Internetware Middleware Framework
- Internetware Software Model
- Internetware Tools and IDE
- Internetware Demo and Case Study

Internetware Characteristics

Cooperative, Situational, Emergent, Evolvable, Autonomous, Trustworthy
Autonomous Component Model

- First of all, the entity of Internetware is a component so that it can be deployed, executed and evolved independently.
- **The Open Collaboration Model** aims to provide a new interpretation for the core of OO model, the object-message structure, according to the open Internet. The separation between software entities and their collaborations make the collaborating logic become explicit and independent entities based on software architectures.
- **The Context Driven Model** aims to build up a model to capture the features and changing modes of the context of the open collaboration model.
- **The Intelligent trustworthy model** aims to solve the problems derived from the open environment, such as the trustworthiness, personalization, self-evolution, etc. by combining the trustworthy computing framework and artificial intelligence on the basis of the context driven model.
Internetware Entity: Reference Implementation

**Entity** (assembled by multiple entities)

- Entity (leveraging object-oriented mutations)
  - Interoperation as Service
  - Behavior as Agent
  - Package as Component

**Built on**
- Basic Component Model
- Open Collaboration Model
- Context Driven Model
- Intelligent trustworthy Model

**Implemented by**
- Computing
- Controlling
- Connecting

**Agent-based Implementation**
- Packaged as components
- Behaving as agents
- Interoperating as services
- Collaborating in a structured manner
- Mainly for new system construction
Internetware Entity: Reference Implementation

Implemented by

Reflective Component Based Implementation

- A set of reflective points is defined in the typical component model and architecture model
- These points instrument Internetware capabilities dynamically
- Able to transform legacy systems
Internetware Infrastructure: Conceptual Framework (1)

Conceptual Architecture of Internetware Middleware

- Provides a runtime space for component
Internetware Infrastructure: Conceptual Framework (2)

Basic Component Model
- Business Interface: computing, controlling, connecting
- Reflective Interface: computing, controlling, connecting

Open Collaboration Model
- entity

Context Awareness Model
- context

Data-driven Quality Model
- Quality
- Data Mining
- Machine Learning

Container
- provides a runtime space for component

Connector
- mediates different protocols between entities

Sensor
- helps to monitor the environment

Analyzer
- Performs data-driven quality assessment

Conceptual Architecture of Internetware Middleware
Internetware Infrastructure: Conceptual Framework (3)

**Architecture at design**
for functionality and predictable quality

guide

**a system at runtime**

**Architecture at runtime**
for adaptability and unpredictable quality

causally connected

**a system at runtime**

---

Conceptual Architecture of Internetware Middleware

Runtime Software Architecture

Container

Connector

Sensor

Analyzer

entity

context

Machine Learning
Internetware Infrastructure: Conceptual Framework (4)

Basic Component Model
- Business Interface
  - Computing
  - Controlling
  - Connecting
- Reflective Interface
  - Computing
  - Controlling
  - Connecting

Open Collaboration Model
- Context Awareness Model
- Data-driven Quality Model
  - Quality
  - Data Mining
  - Machine Learning

Runtime Software Architecture
- Container
- Connector
- Sensor
- Analyzer

Self-Evolution
- Self-Organization
- Self-Adaptation

Conceptual Architecture of Internetware Middleware
Internetware Infrastructure: Reference Model

Runtime Support for Autonomous Component Model

- Incarnation of basic Internetware entities
- Autonomic management
- Incarnation of open collaborations
  - Micro kernel for Componentization
  - Middleware for Internetware
  - entity
  - container
  - Intelligent facilities
  - Trust mechanisms
  - Autonomic loop
  - Reflective framework
  - Runtime software architecture
  - Interoperability protocols
  - Constraint services
  - collaboration
### Entity Container

- **Customize Autonomous Component capabilities on-the-fly**

**The container** is the runtime space of an Internetware entity, which is responsible for managing the entity’s lifecycle (e.g. the class loading, instantiation, caching, release, etc), the entity’s runtime context (e.g. the invocation context and database connections), and the binding between entity’s business functions and quality policies (e.g. interoperability, security, transaction, persistence and rule reasoning).

For a normal java class which implements some business functions and is deployed into a container, it can be bound with various quality policies to implement various advanced features on demand at runtime.
Internetware Infrastructure: Collaboration Incarnation

- **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
  - Implemented in Java EE AS (PKUAS, JonAS), Web Services (Axis), Reflective Component Framework (Fractal), Mobile Middleware (PLASTIC)
Internetware Infrastructure: Autonomic Management

Architecture-Centric Self-Adaptation and Self-Organization

**Analyze** runtime models:
Enriching runtime models by querying and transforming the original models

**Plan** expected models:
Generating TO-BE models by transforming original runtime models

**Read** runtime system:
Runtime system states are organized by customer-defined models just-in-time

**Write** runtime system:
Runtime system states are changed according to the TO-BE customer-defined models

**Monitoring**
**Executing**
**Reasoning**
**Dynamism**
**Trustworthiness**
**Reflection**

Software Architecture

Trustworthiness
Dynamism
Reflection
Reasoning
Analyzing
Planning
Internetware Infrastructure: Reference Implementation

- Fully Supported
  - PKUAS (JEE AS)
  - OnceAS (JEE AS)
  - FlowMe (Mobile)

- Partial Supported
  - JOnAS (JEE AS)
  - OnceSE/BPEL (SOA)
  - OncePortal

- Demo
  - Fractal (Reflective Component Model)
  - PLASTIC (Mobile)
  - Jboss (JEE AS)
ABC (Architecture-Based component Composition) is a systematic engineering approach for developing and evolving Internetware systems.

- Introduces software architecture into the whole software lifecycle.
- Almost all activities in lifecycle stages are performed on software architectures.
- Provides a set of tools and mechanisms for architecture-centric development and evolution.
- Used in 2008 Beijing Olympic Games Information System Modeling.

**ABC Approach to Internetware Engineering**

**Domain Modeling for self-organizing community**

**Domain Model**

**Application Tailoring**

**Requirements Specification**

**Application requirements**

**Software architecture in the whole lifecycle**

- Analysis
- Design
- Composition
- Evolution
- Maintenance
- Deployment
Automated Domain Modeling

Web Services: 10,131
Applications (Mashups): 9,995

(1) Tag as knowledge

Tag Cloud

(2) Domain construction by tag clustering based on given key tags
-- can be refined by changing clustering parameters or manually

(3) Tag as abstract service if similar actual services exist
-- similarity matching for the given tag

(4) Domain with candidate entities for travel is constructed automatically

(5) Refinement and Customization with ABC Feature Modeling Tool
Feature Model to Software Architecture

Features (& relations)
→ responsibilities (& relations)
→ components (& relations)

The Requirement Level

The Specification Level
Refining Software Architecture

- Dynamic and Adaptive Software Architecting
  - Developing a totally new Internetware system
  - Transforming an existing software system to an Internetware system
  - Decision-driven

Legend:
- Activity Artifacts
- Integrated into SASA
- Activity Artifacts
- Invented by SASA
- Adaptation Knowledge Produced by SASA
Incarnating Architecture at Runtime

- **SM@RT**: enable runtime architecture in Internetware
  - **Synchronizer**: causally connecting MOF-compliant models to runtime systems so that changes on one side will cause corresponding changes on the other side just-in-time
  - **Synthesizer**: integrating, orchestrating and applying multiple user-defined model analysis methods on runtime models

Runtime data exists in enterprise systems and embedded systems

MOF and QVT compliant, Non-intrusive, Automated

Feed is-be models to analysis & decision tools

Change data according to to-be models

Effectuate & integrate analysis results

query, transform & synthesize

Visualize
Trustworthy integrated environment

---Trustie Architecture

Trustworthiness evaluation framework and services

Trustie software resource repository is implemented as a sharing service

Trustie collaborative development platform is implemented as a collaborative service
References of Internetware

A Survey paper published on IEEE Computer Special Issue of “Computing in Asia”

Internetware: A Software Paradigm for Internet Computing. IEEE Computer, June 2012

IEEE Software Special Issue “Software Engineering for Internet Computing: Internetware and Beyond” (Jan/Feb 2015)
• Research on Internetware

• Future of Internetware
A New Era: Internet of Everything

- Manufacture Industry
- IT Industry
- Service Industry

- Big Data
  - Mobile & Social Computing
  - Cloud Computing
  - Cyber Physical Computing

- Internet
  - Internet
  - Internet of Things

- computers
- storages
- networks
- things
With the Internet’s upgrade and extension to the mobile Internet, IoT, (1) It is becoming a new platform to connect everything in one space. (2) It motivates some new computing and application modes.
As an emerging phenomena induced by sustained reduction of IT cost and ubiquitous IT applications, Big Data brings new challenges and opportunities for IT and its applications.
Challenges and Opportunities

**New runtime infrastructures** are needed to
- manage the distributed and heterogeneous resources
- enable “Everything-as-a Service” provisioning
- support the high quality of user experiences of “situation-aware” apps running on smartphones, tablet computers, and wearables

**New engineering approaches, techniques, and tools** are needed to
- support the effective development of high-confidence, adaptive software systems for the IoE
- support open-source/crowd-sourcing/swarm intelligence
New **engineering** approaches, techniques, and tools are required for the IoE.

The big data changes the traditional data management and analytic ecosystem, and needs the runtime infrastructure to be more efficient to data collection, storage, processing, mining, and so on.

The software “big” data such as OpenSource and Q&A and advanced AI techniques make the software development highly automated and even intelligent.

New **runtime** infrastructures are needed for the IoE.
Operating System for Internet of Everything: A Trend

Manufacture Industry
IT Industry
Service Industry

Big Data

Mobile & Social Computing
Cloud Computing
Cyber Physical Computing

Operating System

Mobile Internet
Internet
Internet of Things

computers
storages
networks
things
Operating System for Internet of Everything: A Trend

Single Machine OS

- High difficulty in managing low-level hw such as CPU/RAM and allocating resources among processes

Networked OS

- Management of diverse networked and distributed resources, and difficulties reduce by facilities such as MW

Operating System

Mobile & Social Computing

Mobile Internet

computers

Internet

storages

Cyber Physical Computing

networks

Big Data

things

Cloud Computing

Manufacture

Industry

Service Industry

IT Industry

Industry
Operating System for Internet of Everything: A Trend

Single Machine OS

High difficulty in managing low-level hw such as CPU/RAM and allocating resources among processes

Networked OS

Management of diverse networked and distributed resources, and difficulties reduce by facilities such as MW

IoE OS

Resource management should be highly “application-aware”. The IoE OS takes charge of the flexible management of diverse, heterogeneous HW, SW, MW, and promises on-demanding services provisioning.

Resources allocation should be quite programmable, customizable, and situational, e.g., lightweight resource management for IoT and WSN.
Operating System for Internet of Everything: A Trend

Resource management should be highly “application-aware”. The IoE OS takes charge of the flexible management of diverse, heterogeneous HW, SW, MW, and promises on-demanding services provisioning.

Resources allocation should be quite programmable, customizable, and situational, e.g., lightweight resource management for IoT and WSN.

Software-defined should be a promising solution!
Operating System for Internet of Everything: Challenges

New applications and services emerge in various domains in the interconnected cyber space.
Operating System for Internet of Everything: Challenges

New applications and services emerge in various domains

More powerful supports to manage distributed and heterogeneous resources
Operating System for Internet of Everything: Challenges

- Manufacture Industry
- IT Industry
- Service Industry

- Big Data
- Mobile & Social Computing
- Cloud Computing
- Cyber Physical Computing

- Operating System
- Mobile Internet
- Internet
- Internet of Things

New applications and services emerge in various domains

Well structured and autonomic for ubiquitousness

More powerful supports to manage distributed and heterogeneous resources
Increased Big data, including Software Engineering Data

- How to support development of Big Data applications
- How to collect, store, analyze and use SE data
- Data driven Software Engineering
- ......
Increased Variety and Complexity

- New applications and services emerge in various domains and run on different devices
- New engineering methodology
- New tools and platform based on Internet
- New quality measurement and metrics
- ……

Increased Big data, including Software Engineering Data

- How to support development of Big Data applications
- How to collect, store, analyze and use SE data
- Data driven Software Engineering
- ……
Software Complexity Increases

- How to cope with the continually increased complexity of software from the engineering viewpoint?
Software Complexity Increases

• How to cope with the continually increased complexity of software from the engineering viewpoint?

Internet-of-Intelligence

Internet-of-Everything

Internet-of-Computers

CI on the Internet:

Wikipedia
CI for knowledge gathering

reCAPTCHA
CI for text digitalization

FoldIt
CI for biology research
Software Complexity Increases

• How to cope with the continually increased complexity of software from the engineering viewpoint?

How to harvest the collective intelligence (CI) in software engineering

Internet-of-Intelligence

Internet-of-Everything

Internet-of-Computers

CI on the Internet:

Wikipedia
CI for knowledge gathering

reCAPTCHA
CI for text digitalization

FoldIt
CI for biology research
Usage of “Big Data” for Software Development

- Various large repositories of software data are available over the Internet;
- Programmers actually often study and reuse these data (e.g., source code) to help improve productivity.
Usage of “Big Data” for Software Development

- Various large repositories of software data are available over the Internet;
- Programmers actually often study and reuse these data (e.g., source code) to help improve productivity.
Ongoing work on Internetware

• More Powerful Runtime Support:
  – OS for Internetware

• Software Engineering with Internet
  – Collective-Intelligence based Software Engineering

• Data-driven Software Engineering
  – Intelligent Recommendation
  – Software Automation
Ongoing work on Internetware

• More Powerful Runtime Support:
  – OS for Internetware

• Software Engineering with Internet
  – Collective-Intelligence based Software Engineering

• Data-driven Software Engineering
  – Intelligent Recommendation
  – Software Automation
Internetware OS: Overview

- Apps & Data
  - e-Gov
  - Smart City
  - Campus

- Framework
  - Enterprise Computing
  - Mobile Computing

- System Call
  - Virtualized Resource Management and Programming Framework

- Resource Management
  - Hardware Resource Virtualization and Management

- Devices
  - Compute
  - Storage
  - Network
  - Peripheral
  - PC
  - Phone
  - Tablet
  - Wearable
Internetware OS: Overview

On-Demand Cloud Service Provisioning

Virtualized Resource Management and Programming Framework

Adaptive Resource Management

Apps & Data

System Call

Resource Management

Framework

e-Gov

Smart City

Campus

Compute
Storage
Network
Peripheral

Devices

PC
Phone
Tablet
Wearable

IaaS, PaaS, SaaS, DaaS & KaaS
YanCloud: Internetware OS for IaaS

- **IaaS Cloud Management**: A Software-Defined Way
  - Customization of Cloud Resource Allocation
  - Application-Aware Cloud Service Provisioning

**Private Cloud (e-Gov) / Hybrid Cloud (Campus)**

- MaaS Lib (Reliability/Security)
- Control Pane (Web UI)
- Customization (MOF/QVT/Java)

**Software-Defined Resource Management**

**MaaS+SM@RT**

**API Design, Integration, and Orchestration**

**Native Management Facility**

- Vmware/HyperV/Xen/KVM/PowerVM, Vsphere/Openstack/Cloudstack, HyperiC

**Hardware**
- Server, Storage, Network, Peripheral

**Software**
- OS, DB, MW, Application
YanCloud: Internetware OS for IaaS

Based on SM@RT, a runtime model specifies the causality between applications and status of underlying platform resources like CPU, RAM, network, VM, MW, DB, AppServer...

Automated generation of management API from SM@RT, whereas Cloud admins and developers could program and customize the management solutions in DIY fashion.
Define a new programming abstraction, called Service-Model-View-Controller, to refactor the mobile apps for adaptive resource consumption spanning client and cloud, e.g., computation/data offloading, hybrid composition.
Define a new programming abstraction, called Service-Model-View-Controller, to refactor the mobile apps for adaptive resource consumption spanning client and cloud, e.g., computation/data offloading, hybrid composition.
YanPhone: Internetware OS for Mobile-Cloud

API generation from the programming abstraction to enable the adaptive resource sharing and utilization, such as computation, storage, network, display, sensors,....
YanPhone: Internetware OS for Mobile-Cloud

Automated refactoring legacy Web applications for mobile browsing and enabling composition with other apps

China Computer Federation Portal
http://www.ccf.org.cn/sites/ccf/

Official Android Apps:
3 developers + 5 month, not real-time (3+ days news latency), no integration

YanPhone for CCF
1 senior undergraduate, 2 workdays, real-time, fast page load, 16 APIs for external composition
Ongoing work on Internetware

- **More Powerful Runtime Support:**
  - OS for Internetware

- **Software Engineering with Internet**
  - Collective-Intelligence based Software Engineering
    Use the collective intelligence of large-scale software developers on the Internet to support the development of software with continually-increased complexity.

- **Data-driven Software Engineering**
  - Intelligent Recommendation
  - Software Automation
Collective-Intelligence based SE (CISE) — The Current Practice

Open Source Software Development
- Attract people to participate in software development through open source

Crowdsourcing Software Development
- Outsource software development tasks or problems to the crowd on the Internet

Application Store
- Use the market mechanism to guide the development of a large set of applications

How to utilize collective intelligence to support the development of Internetware?
While there are many successful instances and insightful post analyses of CISE, the design of CISE systems is still an exploratory experiment.

Two difficulties causing this situation:

① The complexity of software development
   • Software development involves different phases, different artifacts, and complex relationships between these elements;
   • Each of them may require different mechanisms for them to be collectively conducted/constructed/maintained

② The complexity of human collective intelligence
   • Human collective intelligence is not purely technical, but also involves attributes of psychology, sociology, and economics;
   • These attributes should be managed appropriately to enable the emergence of collective intelligence.
CISE: Issues and Challenges

- Driving forces of the forming of human collectives on the Internet
- Massively decomposing mechanisms for different software development problems
- Nature of human collective intelligence
- Measurement and control of collective intelligence
CISE: Our Research Plan

1.1 Analyze the CI mechanism of artificial CI phenomena

1.2 Identify the CI mechanism of natural CI phenomena

2 Develop theories and methods for CISE

3.1 Develop technical platform to support CISE

3.2 Experiments and Practice
Current Progress (1): Mining Open-Source Software (OSS) Repositories

We keep tracking various open source projects.
- Level 0: raw data
- Level 1: structured data
- Level 2-n: filtered data

Data Gathering (80T)

Data Analyzing

Question: How long does it take for an average developer to become fluent in an open-source software project?

Result 1: Developers’ productivity plateaus within 6-7 months in small and medium projects and it takes more than 12 months in large projects.

Result 2: It takes developers at least 3 years to become fluent in large projects.

https://passion-lab.org
Stigmergy

- The process that leads to collective intelligence in social insects
- Information that one leaves in the environment is perceived by others and stimulates them to leave new information, which forms the positive feedback loop among different individuals’ behavior.

Based on stigmergy, we develop a mechanism called **Incremental Graph Superimposition** (IGS).

With IGS, a large collective of developers can collaboratively construct software artifacts without any direct interaction between each other.
Ongoing work on Internetware

• More Powerful Runtime Support:
  – OS for Internetware

• Software Engineering with Internet
  – Collective-Intelligence based Software Engineering

• Data-driven Software Engineering
  – Intelligent Recommendation
  – Software Automation
Data-driven Software Engineering

- Many successful stories on Big Data, such as
  - Business intelligence, Intelligent transportation, Natural language processing, Automatic question-answering system.

- Massive software engineering data are available in many software companies and over the Internet, including
  - various large repositories of source code, documents, comments, tags, files, etc.

- “Big Data” can make software engineering smarter!
Current Progress: Data-driven Intelligent Recommendation

Knowledge based **intelligent** recommendation

- development aid: bug fixing/code completion
- assets
- tools
- experts

Knowledge Library

- Domain specific
- Library of domain knowledge
- Knowledge library management system
- QA system
- Search and browse
- Intelligent recommendation

Intelligent DE

Local knowledge library

Library of domain knowledge

Requirement tools
- requirement

Modeling tools
- analysis

Testing tools
- design

Maintenance tools
- coding

Project analysis tools
- testing
- maintenance
- evolution
Can Big Data bring new opportunities to software automation?

Can we develop a data-driven approach to automatically synthesizing real large programs?

Two possible application scenarios

• Scenario one
  – Extracting the code for implementing functionality X from existing software repositories and patching it to the program under development to add functionality X

• Scenario two
  – Extracting the code for implementing a series of functionalities from existing software repositories to form a new program with these functionalities
Decompose the user’s requirement description by natural language analysis.

Find reusable code snippets by search in big code data.

Generate glue code by learning the big code data.

Fix code bugs automatically by learning existed bug fix records.

A framework for realizing scenario one
Current Progress(1): Some Preliminary Experiments

Experiment 1: Automatic Bug Fixing based on History Data

Step 1: Q&A page extraction

Step 2: Edit script generation

Step 3: patch generation and filtering
**Current Progress(1): Some Preliminary Experiments**

**Experiment 1: Automatic Bug Fixing based on History Data**

- Automatically fix 8 out of 24 bugs in the Android projects on GitHub;
- Acceptable overhead: 76.5 seconds for each.

---

**Step 1: Q&A page extraction**  
**Step 2: Edit script generation**  
**Step 3: patch generation and filtering**
Experiment 2: Program Generation

Model
• Recurrent neural networks

Learning
• Problem & Example Code

Generating
• With problem description, a program is generated, with four tiny errors.

Problem Description:
find the max number and the sec-max number.

Solution:
Thanks

Internetware
Today and Future
Hong Mei