Internetware

A Software Paradigm for Internet Computing

Gang Huang
Key Laboratory of High Confidence Software Technologies, Ministry of Education
Institute of Software, Peking University
hg@pku.edu.cn

Agenda

- Internet As A Computer and its Distinguished Software Characteristics
- Challenges Addressed by Internetware
- Internetware Research and Practice

Internet as a Computer (Internet Computer)

- **Internet is evolving to a Global Ubiquitous Computer**
  - Many big and hot trends in IT research and business try 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

Big Trend

Internet as a Computer

Business Trend

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

•Grid/Cloud computing proposes a new model of networked applications from the perspective of resource sharing and management.
•Pervasive computing discusses a new situation of networked applications from the perspective of human computer interaction.
•Service Oriented Computing focuses on a new form of software with emphasis on collaboration and dynamism from the philosophy of software as a service.
•…

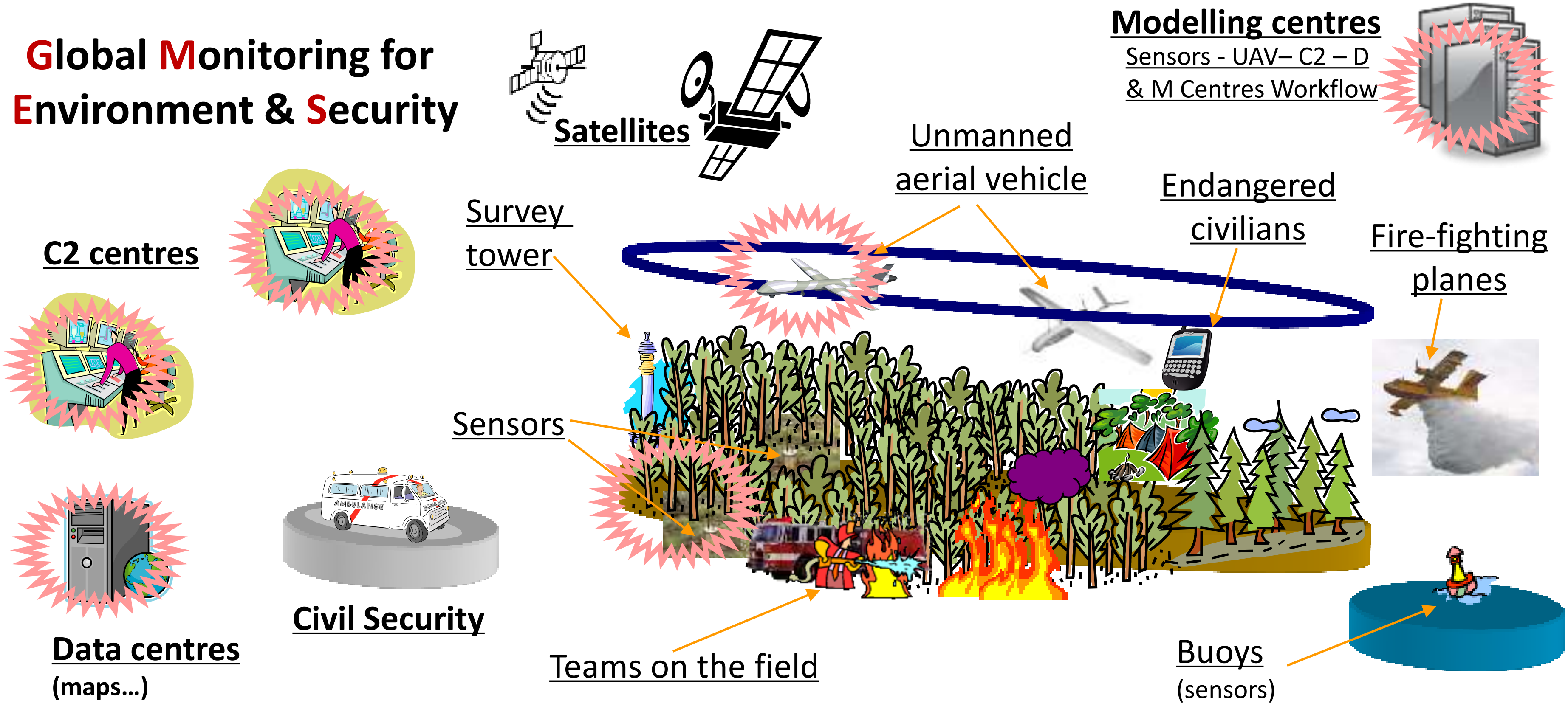
IBM Microsoft Google amazon salesforce.com Success. Not Software.

Scenarios of Internet Computer

Cooperation On Demand (emergent)

happens anytime anywhere among anyone (just like the Internet).
involves partners with no or little relations before and after.

Global Monitoring for Environment & Security



Satellites

Modelling centres
Sensors - UAV– C2 – D
& M Centres Workflow

Unmanned aerial vehicle

Endangered civilians

Fire-fighting planes

Survey tower

C2 centres

Sensors

Data centres
(maps…)

Civil Security

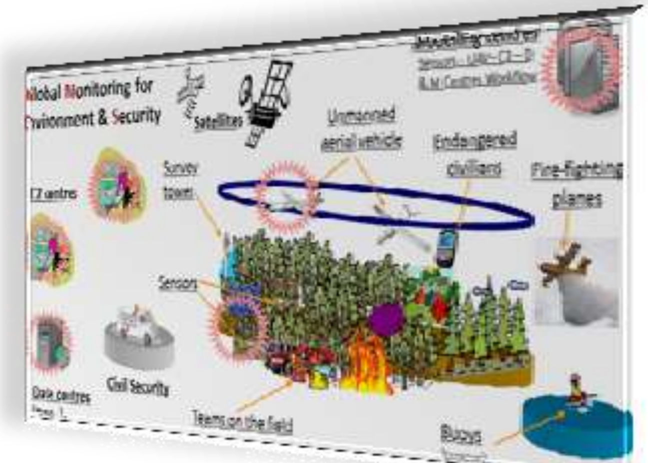
Teams on the field

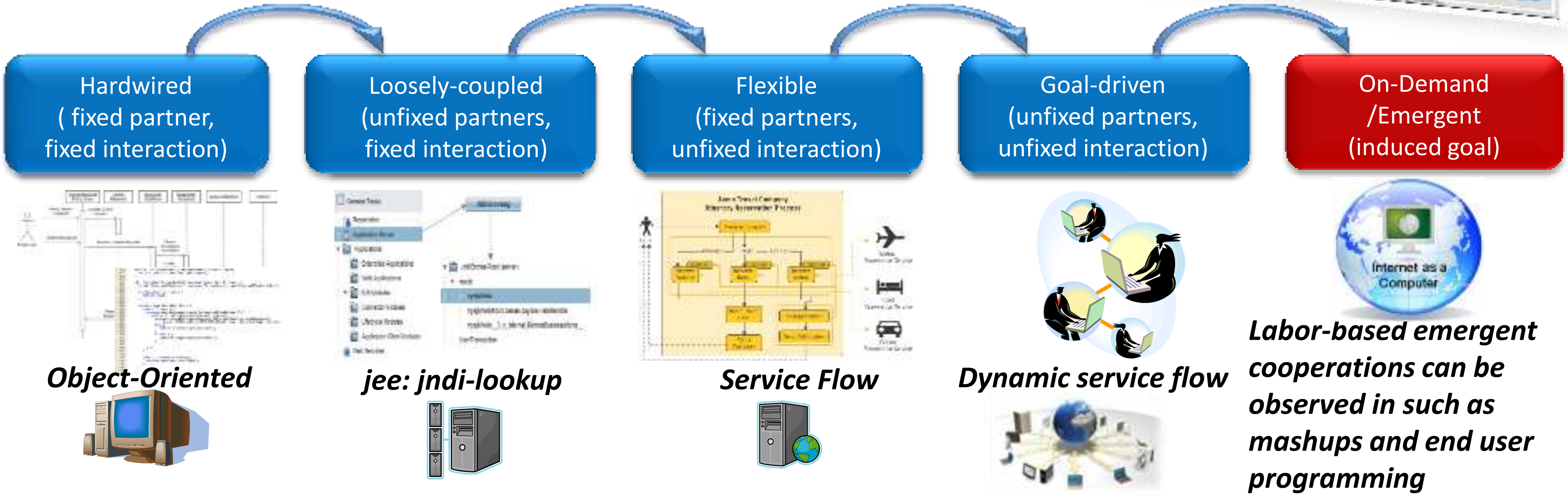
Buoys
(sensors)

Software for Emergent Cooperation

Such emergent cooperations enabled by software are ad hoc, labor-based and un-trusted

- Before “Internet as a Computer”, the goal of cooperation is predefined or predicted while the partners and interactions can be fixed or not
- In the era of “Internet as a Computer”, more and more goals are unclear, unexpected or un-deterministic before the cooperation finally happens (e.g. real-world emergency)





Labor-based emergent cooperations can be observed in such as mashups and end user programming

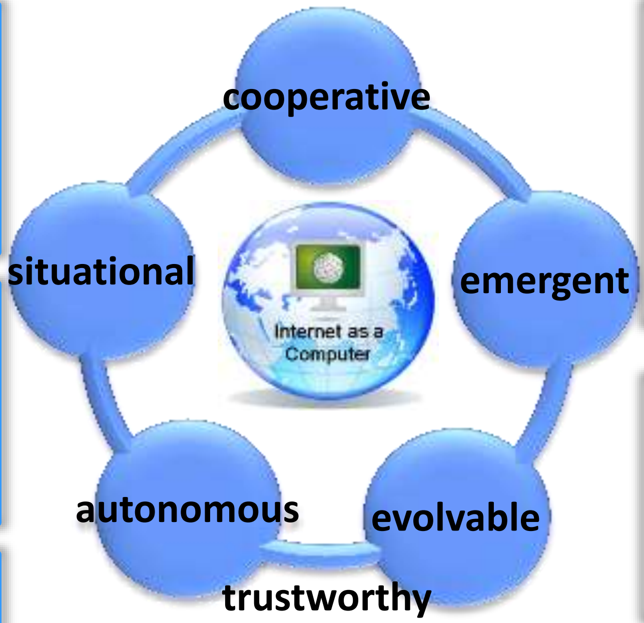
Internet Computer Software Characteristics

"Internet Computer" brings many distinguished characteristics to software systems for implementing new business naturally with new technology.

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

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

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



Emergent: software may have un-designed behaviors or un-expected effects on its runtime instances or interactions with others

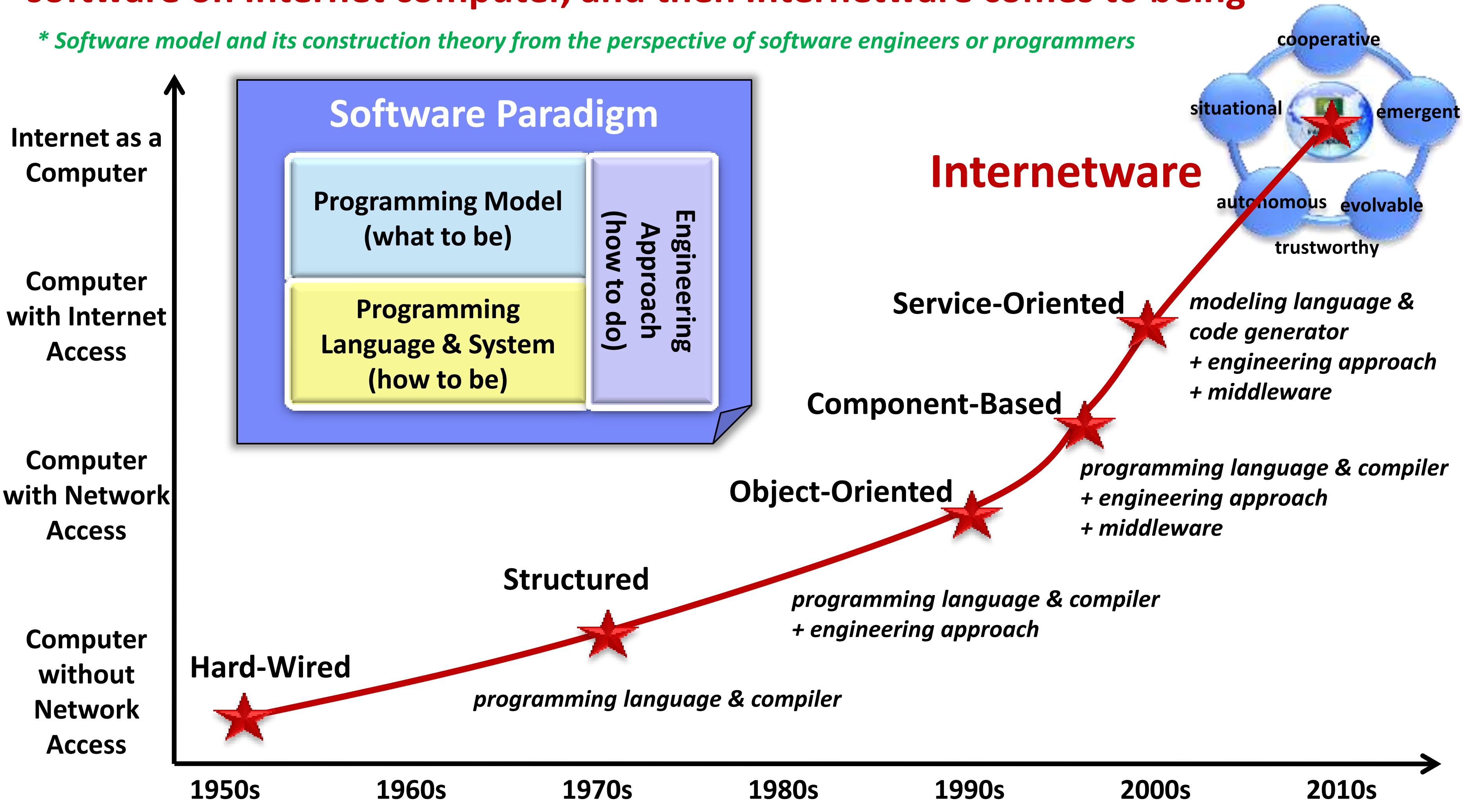
Evolvable: software is easy to add, remove and change its functionalities on-the-fly and just-in-time

Trustworthy: software should promise some kind of tradeoff among process quality, internal system quality, external system quality and usage quality.

Internetware for Internet as a Computer

Existing software paradigms* cannot well support those new characteristics of software on Internet computer, and then Internetware comes to being

** Software model and its construction theory from the perspective of software engineers or programmers*



Internet as a Computer

Computer with Internet Access

Computer with Network Access

Computer without Network Access

Software Paradigm

Programming Model (what to be)

Programming Language & System (how to be)

Engineering Approach (how to do)

Internetware

cooperative

situational

emergent

autonomous

evolvable

trustworthy

Service-Oriented

modeling language & code generator
+ engineering approach
+ middleware

Component-Based

programming language & compiler
+ engineering approach
+ middleware

Object-Oriented

programming language & compiler
+ engineering approach

Structured

programming language & compiler

Hard-Wired

1950s 1960s 1970s 1980s 1990s 2000s 2010s

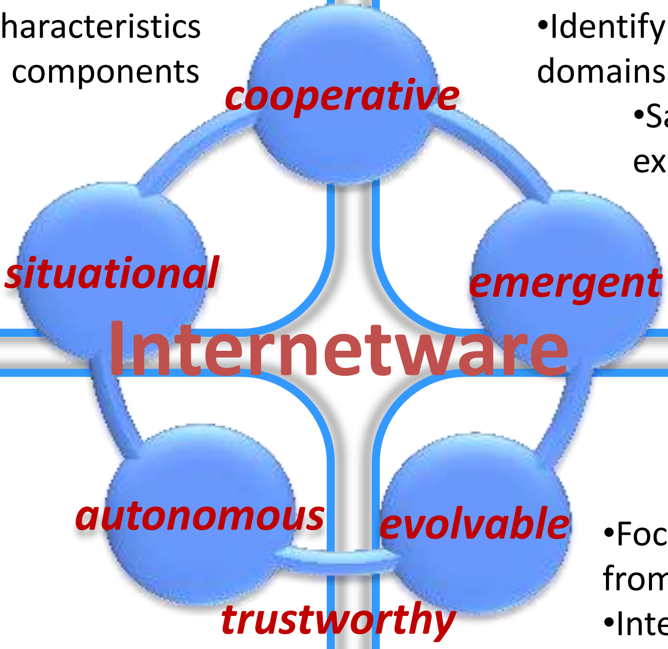
Challenges to Internetware

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



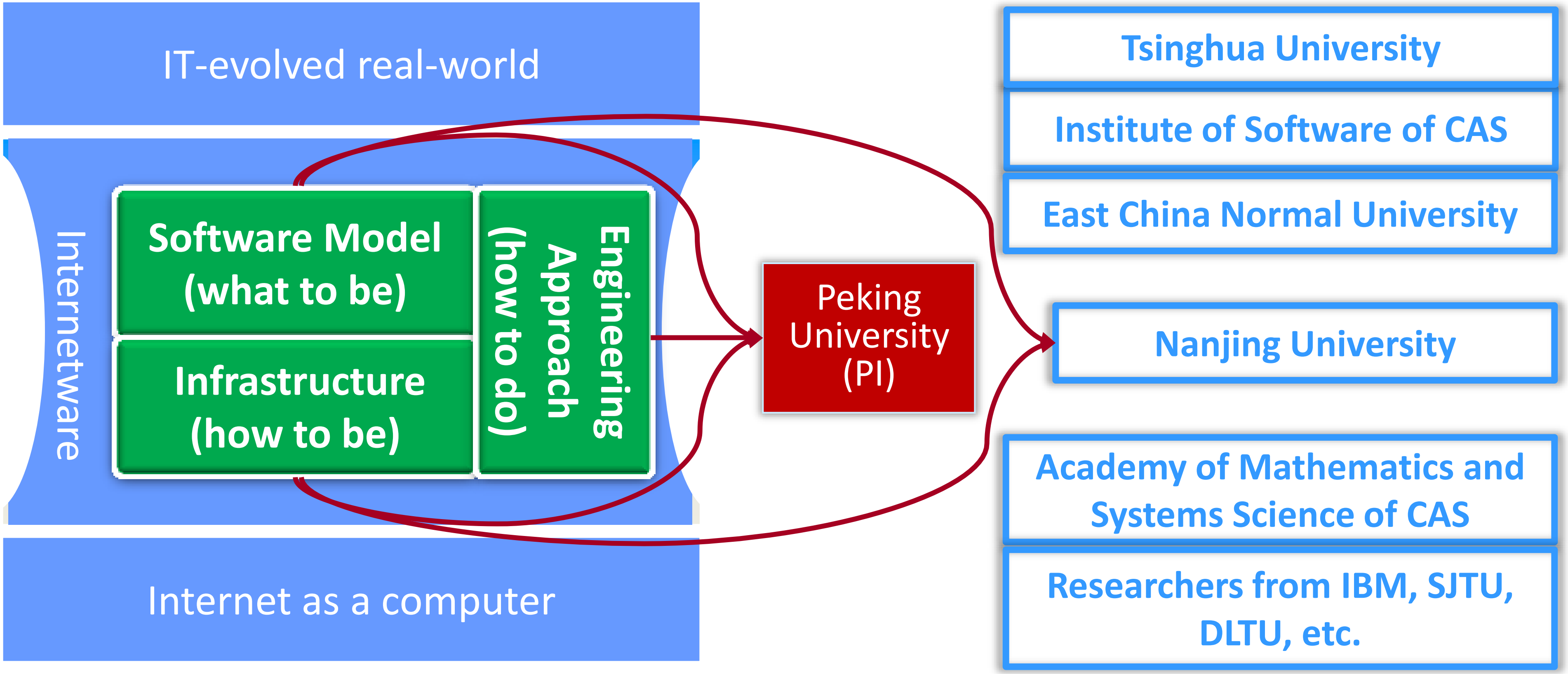
Programming Language & System (how to be)

- incarnates the elements and their relationships of a software model
- Internetware middleware should
- Provide a container for instantiating and operating Internetware components Provide collaboration mechanisms.
- Equip legacy software systems with Internetware characteristics
- Enable context-awareness and reflection

Quality assurance (how to be good enough)

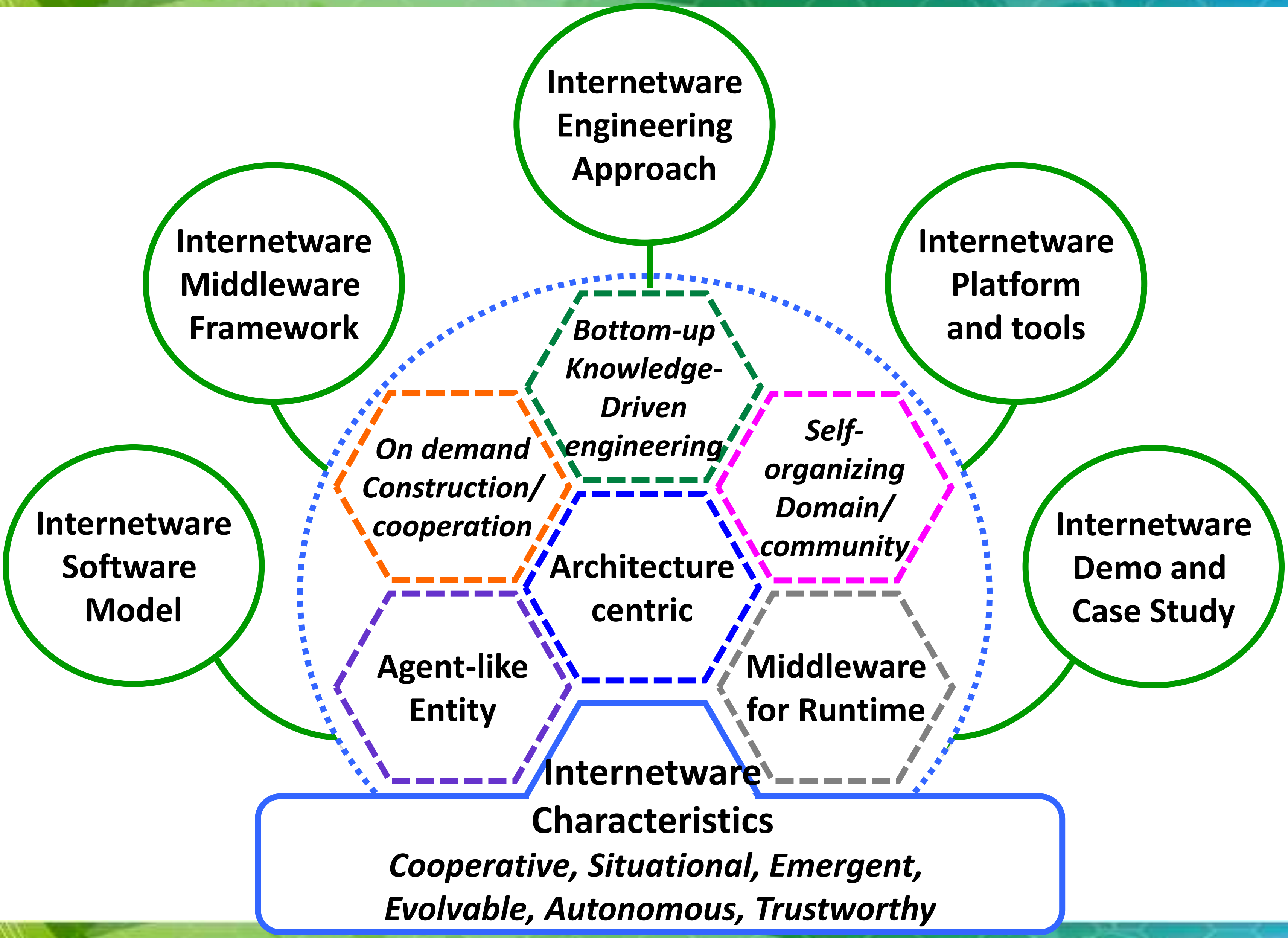
- Focal points of software quality change from system-centric to usage-centric
- Internetware quality assurance should define quantitative and qualitative evaluation framework for quality
- Assure the quality via engineering approach at development time as well as middleware at runtime

Internetware Research in China


IT-evolved real-world
Internetware
Software Model (what to be)
Infrastructure (how to be)
Engineering Approach (how to do)
Internet as a computer
Peking University (PI)
Tsinghua University
Institute of Software of CAS
East China Normal University
Nanjing University
Academy of Mathematics and Systems Science of CAS
Researchers from IBM, SJTU, DLTU, etc.

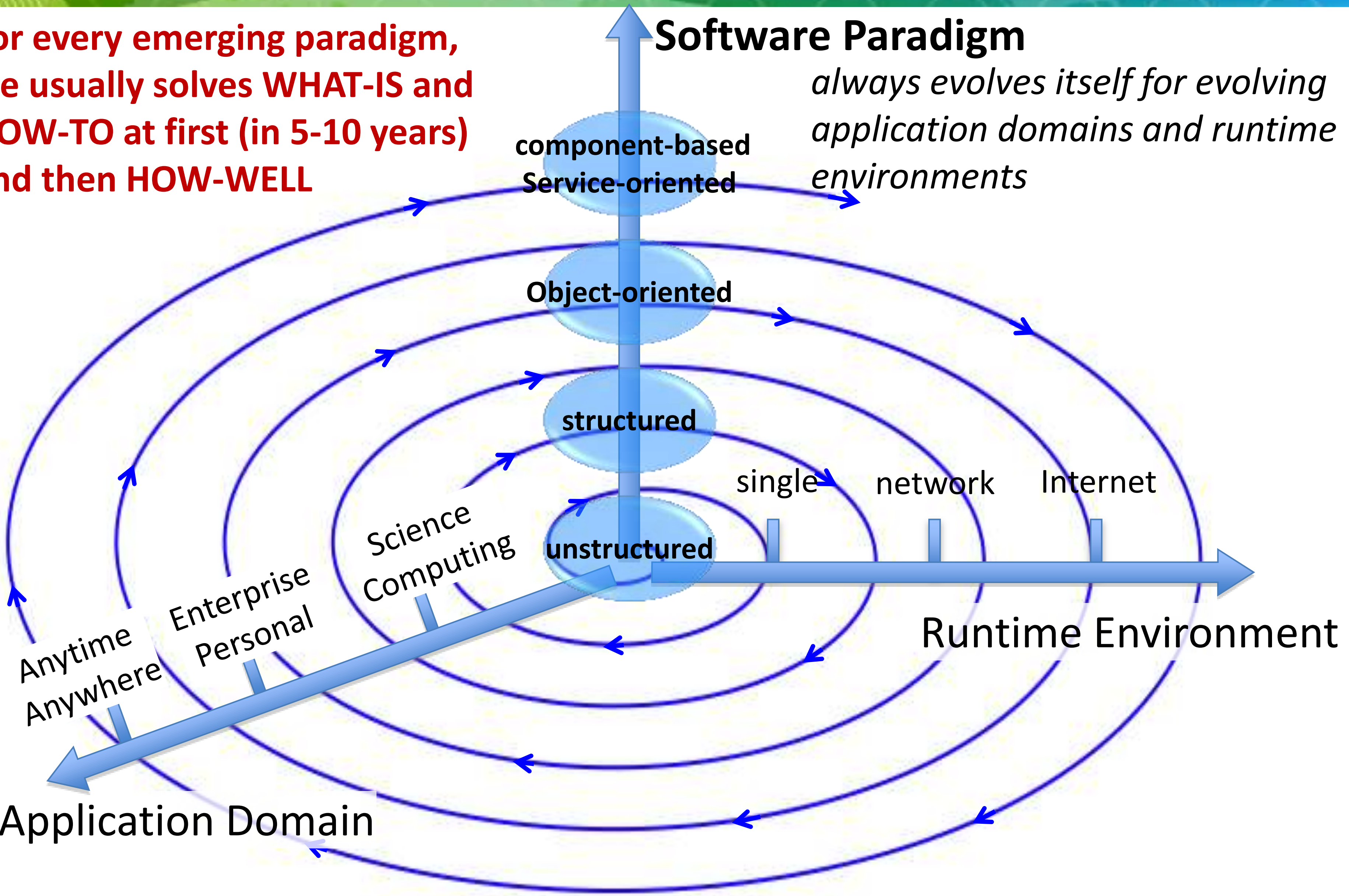
- **"Theory and Methodology of Agent-based Middleware on Internet Platform"**
  - The first national basic research program (973) project on software
  - From 2002~2008; > 80 faculty; 17 post-doc ; 88 Ph.D; 364 Master
- **"High Confidence of Internetware"**
  - IBM joined as the first foreign company in 973 program
  - From 2009~2013; > 100 faculty

Internetware R&D Outputs


Internetware Engineering Approach
Internetware Middleware Framework
Internetware Platform and tools
Internetware Software Model
Internetware Demo and Case Study
Bottom-up Knowledge-Driven engineering
On demand Construction/ cooperation
Self-organizing Domain/ community
Architecture centric
Agent-like Entity
Middleware for Runtime
Internetware Characteristics
Cooperative, Situational, Emergent, Evolvable, Autonomous, Trustworthy

High Confidence of Internetware

For every emerging paradigm, we usually solves WHAT-IS and HOW-TO at first (in 5-10 years) and then HOW-WELL



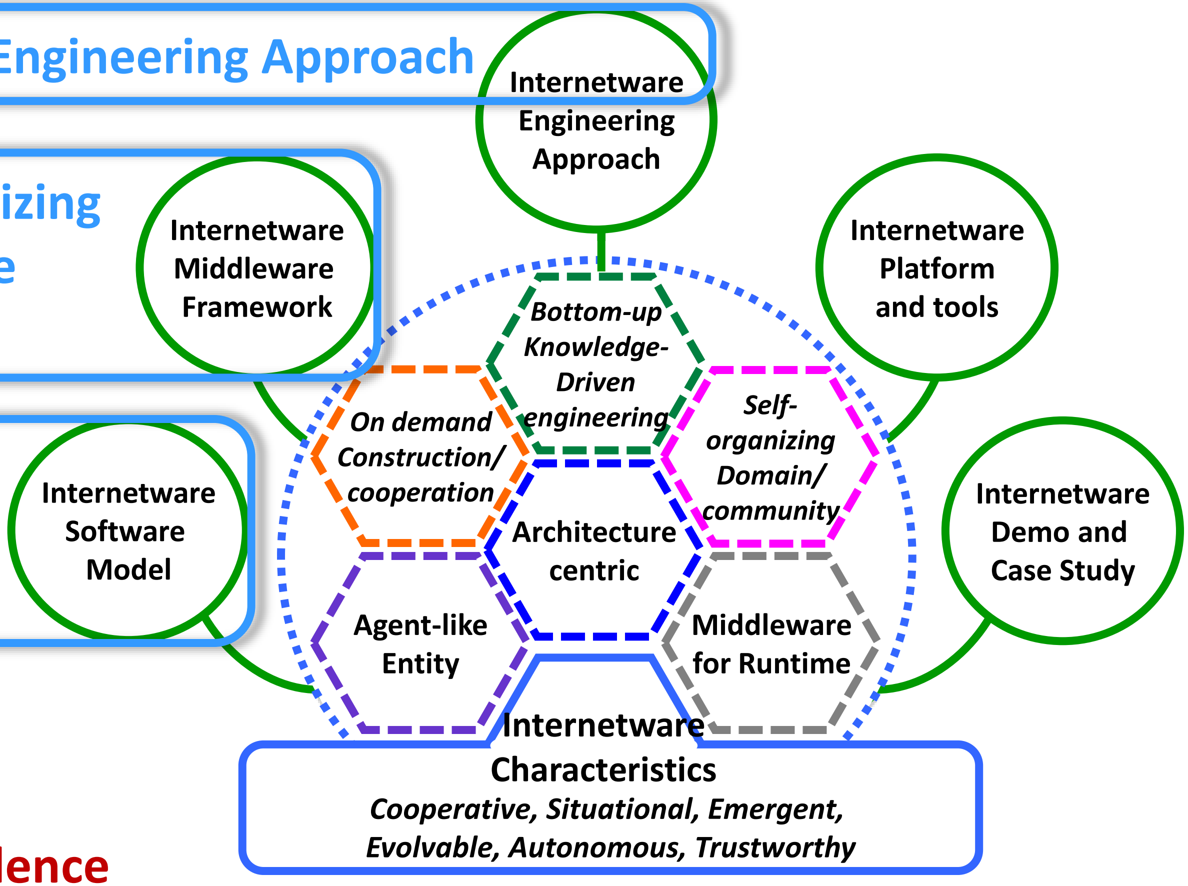
always evolves itself for evolving application domains and runtime environments

Roadmap of High Confidence Internetware

Quality Aware Internetware Engineering Approach

Self-Adaptive and Self-Organizing Internetware Middleware (via Models at Runtime)

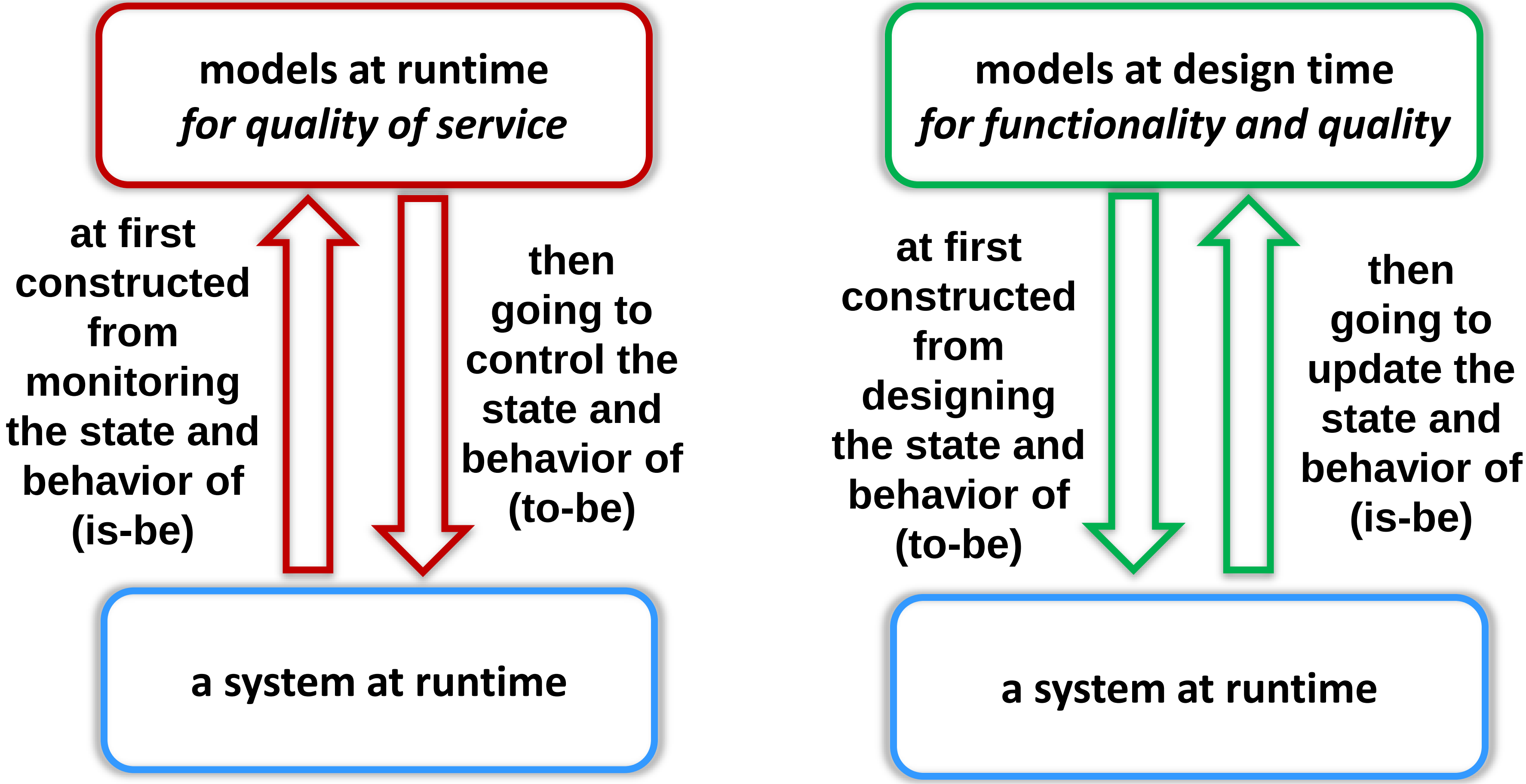
Internetware Quality Model and Evaluation Approach


Internetware Engineering Approach
Internetware Middleware Framework
Internetware Platform and tools
Internetware Software Model
Internetware Demo and Case Study
Bottom-up Knowledge-Driven engineering
On demand Construction/ cooperation
Self-organizing Domain/ community
Architecture centric
Agent-like Entity
Middleware for Runtime
Internetware Characteristics
Cooperative, Situational, Emergent, Evolvable, Autonomous, Trustworthy

Extend existing outputs with focus on killer applications of Internetware and High Confidence in emerging computing and application paradigms, like cloud computing, mobile Internet, internet of things, cyber-physical systems, social networking, etc.

Definition of Models at Runtime (M@RT)

- **Models@Runtime are models causally connected to the state and behavior of a runtime system**



models at runtime
for quality of service

at first constructed from monitoring the state and behavior of (is-be)

then going to control the state and behavior of (to-be)

a system at runtime

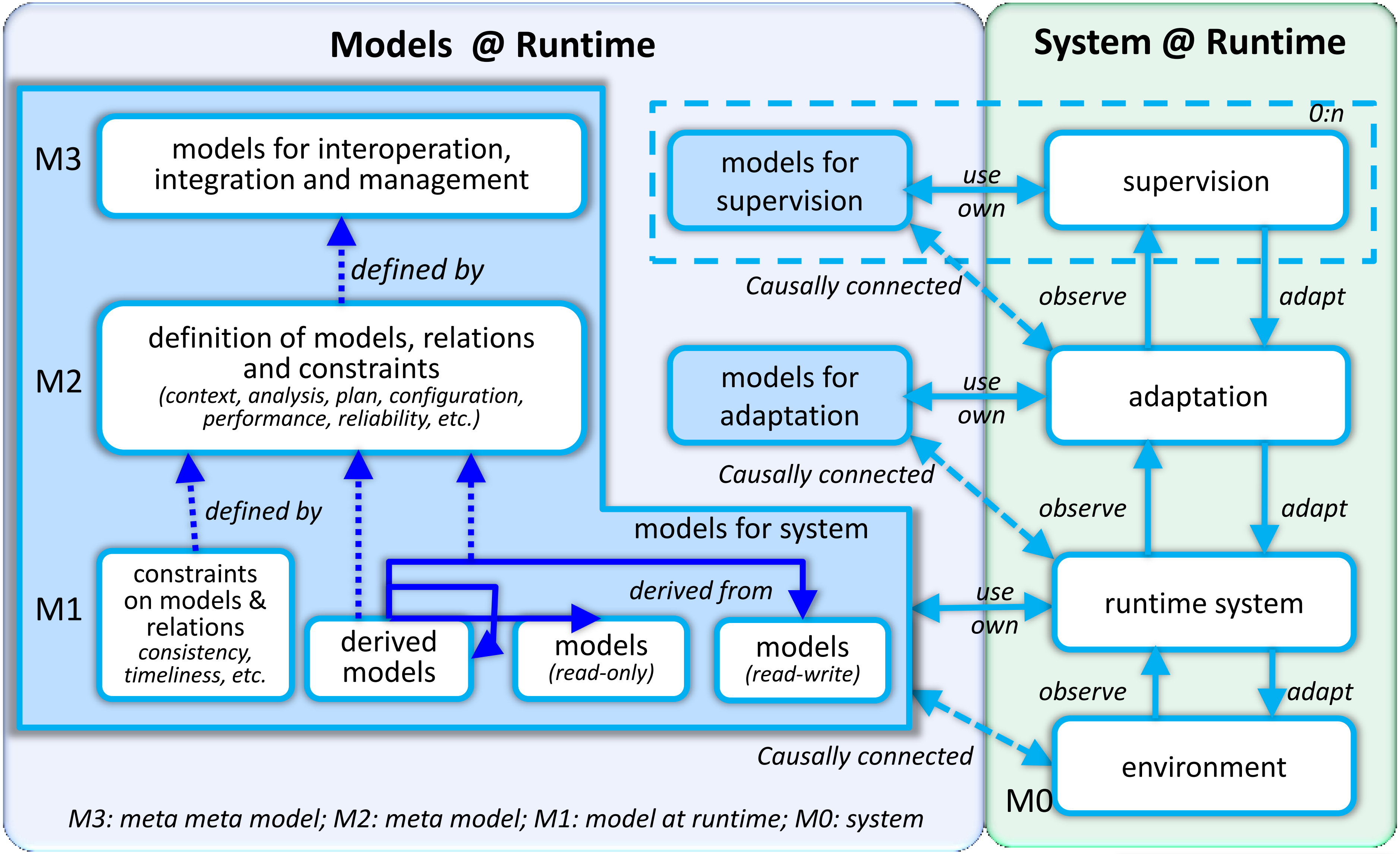
models at design time
for functionality and quality

at first constructed from designing the state and behavior of (to-be)

then going to update the state and behavior of (is-be)

a system at runtime

Reference Architecture of M@RT



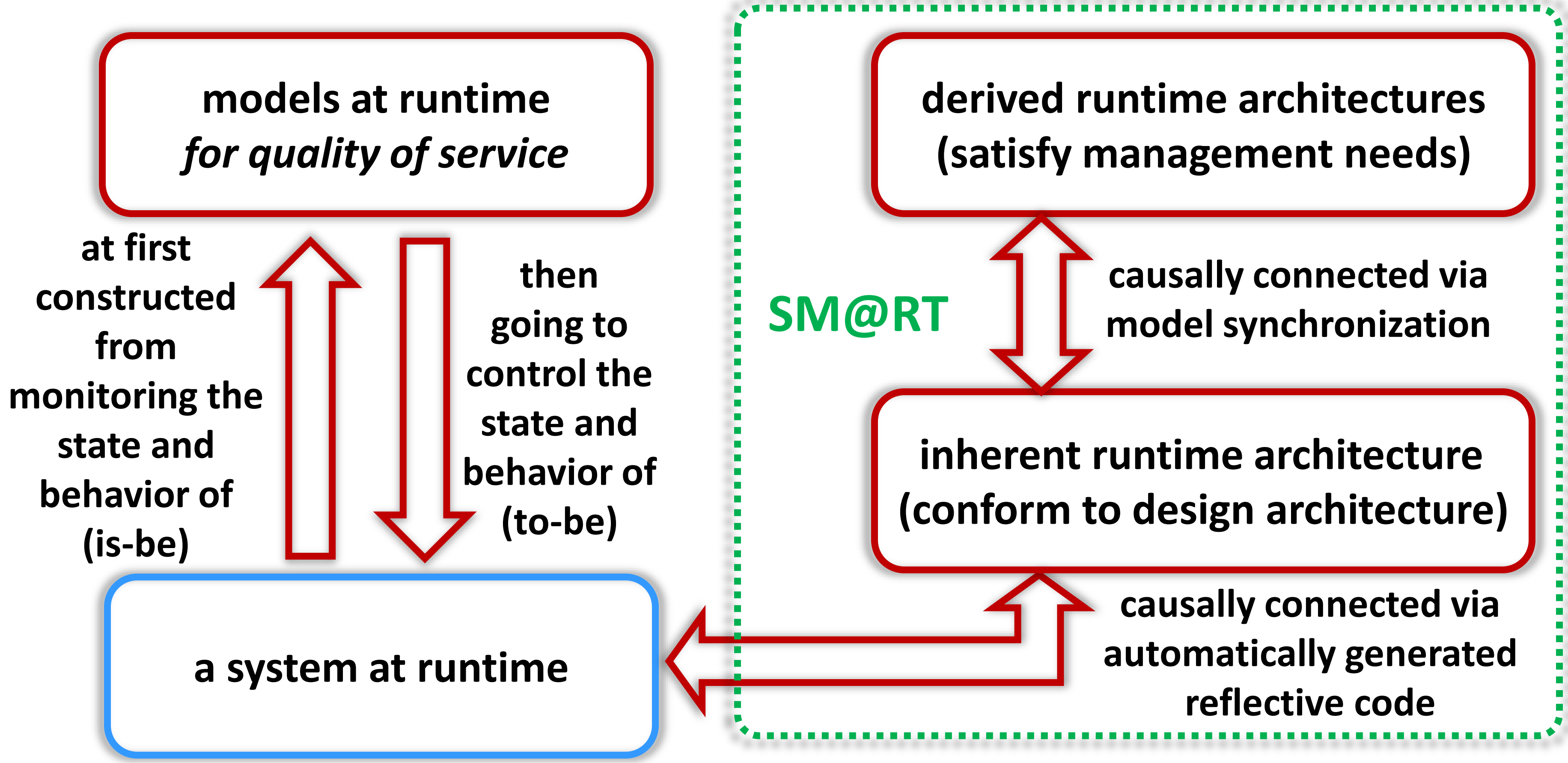
M3: meta meta model; M2: meta model; M1: model at runtime; M0: system

InternetWare

SCHLOSS DAGSTUHL Leibniz-Zentrum für Informatik

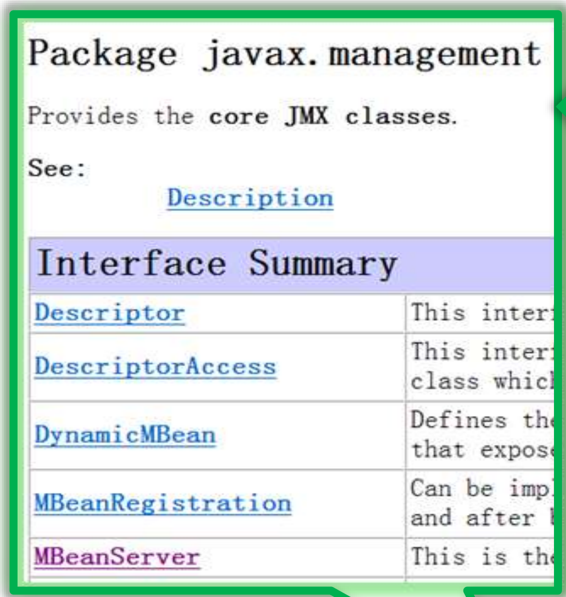
SM@RT: Supporting Models at Runtime

- **SM@RT: A model-driven framework for constructing the causal connection between the architectural models and runtime systems in an automated manner (using MOF/QVT standards)**


models at runtime
for quality of service
at first constructed from monitoring the state and behavior of (is-be)
then going to control the state and behavior of (to-be)
a system at runtime
SM@RT
derived runtime architectures (satisfy management needs)
causally connected via model synchronization
inherent runtime architecture (conform to design architecture)
causally connected via automatically generated reflective code

Construction of Inherent Runtime Architecture

3) define QVT-based mapping between the meta model and system manageability



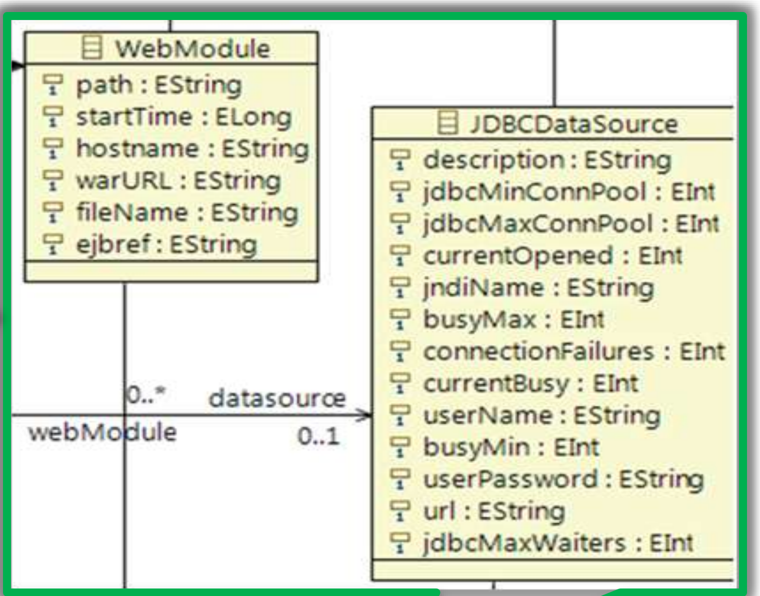
Package javax.management

Provides the core JMX classes.

See:
Description

Interface Summary	
Descriptor	This inter
DescriptorAccess	This inter class whic
DynamicMBean	Defines the that expos
MBeanRegistration	Can be imp and after
MBeanServer	This is the

```
pping SysElement::xAdd(prop:Property,v:Object):
 result:Object
en(self,type=MBeanServer; prop=ejb)
var fileName=v.fileName;
var server:=self.parent.auxiliary.server;
\{ String[] signature={"java.lang.String"};
  String[] params=new String[$fileName];
  String deployedName=(String)mgmt.invoke($server,
    "deployJar", params, signature);
 [result = ObjectName.newInstance(deployedName); }] }
```



1) define MOF-based meta model of

2) select the manageability of

inherent runtime architecture (conform to design architecture)

a system at runtime

causally connected via automatically generated reflective code

Construction of Inherent Runtime Architecture

3) define QVT-based mapping between the meta model and system manageability

What operation of meta model

Under what meta model's condition

for which model elements

```
mapping SysElement::getX(prop:Property): result:Object
when{self.type=Any and prop=Any}
{ var aux:=self.parent.auxiliary;
  \[ Management mgmt = $aux.getMainEntry();
     $result = mgmt.getAttribute($self.core, $prop.name); \] }
```

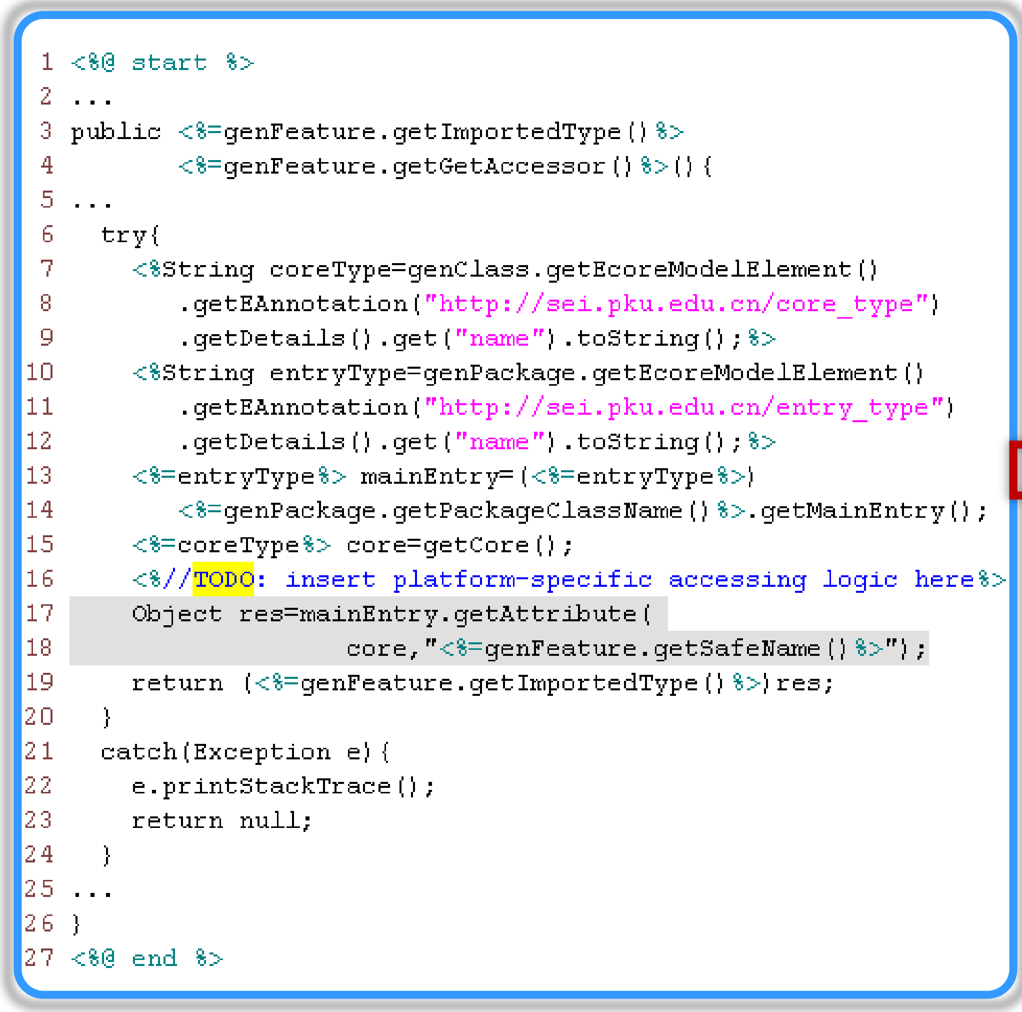
Actual invocation to the manageability

```
mapping SysElement::addX(prop:Property,v:Object):
    result:Object
when{self.type=MBeanServer; prop=ejb}
{ var fileName=v.fileName;
  var server:=self.parent.auxiliary.server;
  \[ String[] signature={"java.lang.String"};
     String[] params=new String{$fileName};
     String deployedName=(String)mgmt.invoke($server,
        "deployJar", params, signature);
     $result = ObjectName.newInstance(deployedName); \] }
```

when get/set the value of a model element, if necessary, invoke the corresponding management APIs

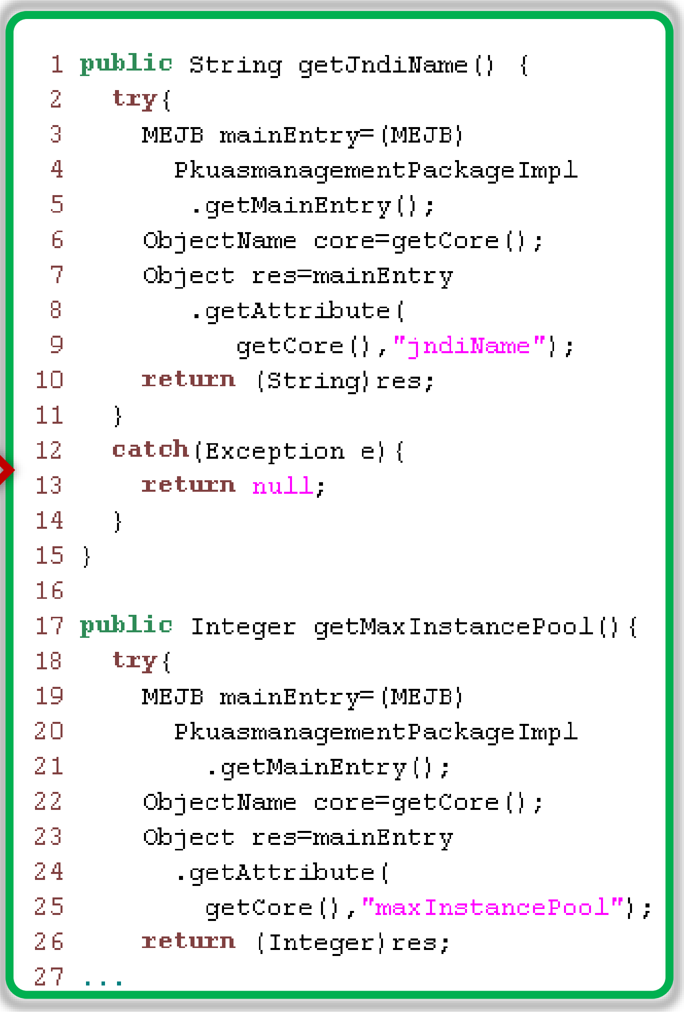
Construction of Inherent Runtime Architecture

4) generate the code based on Eclipse M2M and JET



```
<%@ start %>
...
public <%=genFeature.getImportedType()%>
       <%=genFeature.getGetAccessor()%>(){
...
  try{
    <%String coreType=genClass.getEcoreModelElement()
       .getEAnnotation("http://sei.pku.edu.cn/core_type")
       .getDetails().get("name").toString();%>
    <%String entryType=genPackage.getEcoreModelElement()
       .getEAnnotation("http://sei.pku.edu.cn/entry_type")
       .getDetails().get("name").toString();%>
    <%=entryType%> mainEntry=(<%=entryType%>)
       <%=genPackage.getPackageClassName()%>.getMainEntry();
    <%=coreType%> core=getCore();
    <%//TODO: insert platform-specific accessing logic here%>
    Object res=mainEntry.getAttribute(
                  core,"<%=genFeature.getSafeName()%>");
    return (<%=genFeature.getImportedType()%>)res;
  }
  catch(Exception e){
    e.printStackTrace();
    return null;
  }
...
}
<%@ end %>
```

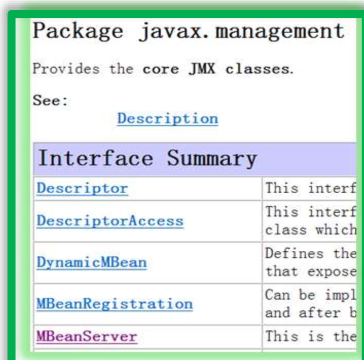
JET template for the getX methods



```
public String getJndiName() {
  try{
    MEJB mainEntry=(MEJB)
      PkuasmanagementPackageImpl
       .getMainEntry();
    ObjectName core=getCore();
    Object res=mainEntry
       .getAttribute(
          getCore(),"jndiName");
    return (String)res;
  }
  catch(Exception e){
    return null;
  }
}

public Integer getMaxInstancePool(){
  try{
    MEJB mainEntry=(MEJB)
      PkuasmanagementPackageImpl
        .getMainEntry();
    ObjectName core=getCore();
    Object res=mainEntry
      .getAttribute(
        getCore(),"maxInstancePool");
    return (Integer)res;
...
```

generated code for getJndiName

Construction of Inherent Runtime Architecture



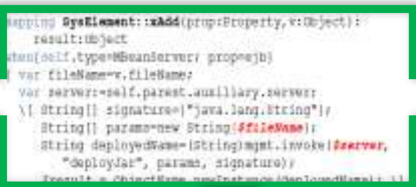
Package javax.management

Provides the core JMX classes.

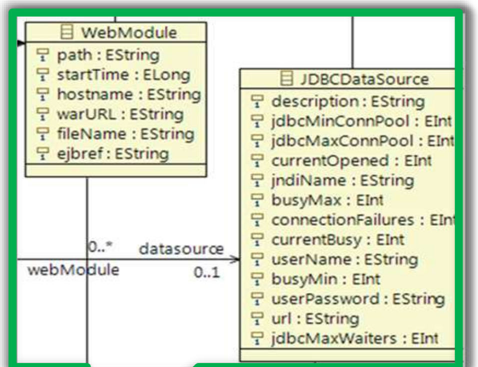
See:
Description

Interface Summary	
Descriptor	This interf
DescriptorAccess	This interf class which
DynamicMBean	Defines the that expose
MBeanRegistration	Can be impl and after b
MBeanServer	This is the

3) define QVT-based mapping between the meta model and system manageability



```
mapping SysElement::xAdd(prop:Property,v:Object):
    result:Object
when{self.type=MBeanServer; prop=ejb}
{ var fileName=v.fileName;
  var server:=self.parent.auxiliary.server;
  \[ String[] signature={"java.lang.String"};
     String[] params=new String{$fileName};
     String deployedName=(String)mgmt.invoke($server,
        "deployJar", params, signature);
   [result = ObjectName.newInstance(DeployedName); ]]
```

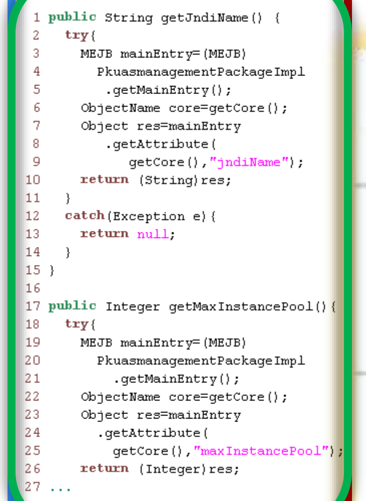


2) select the manageability of

4) generate reflective code

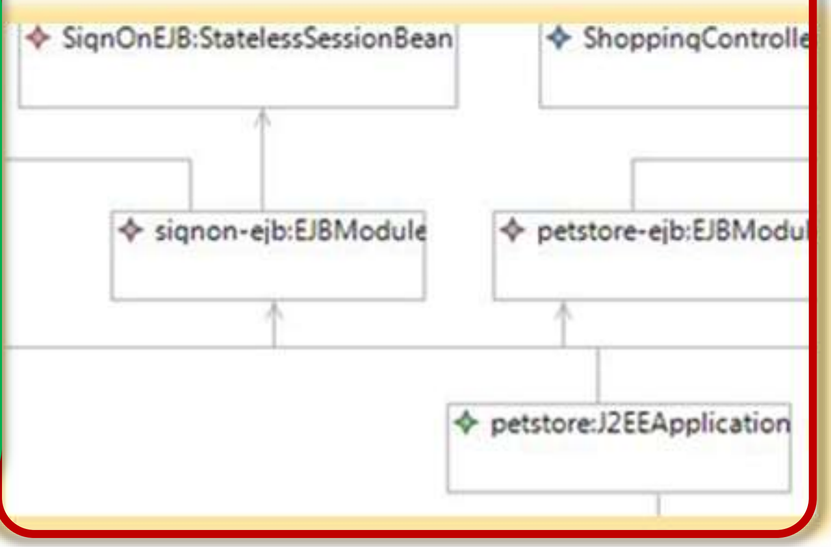
1) define MOF-based meta model of

a system at runtime

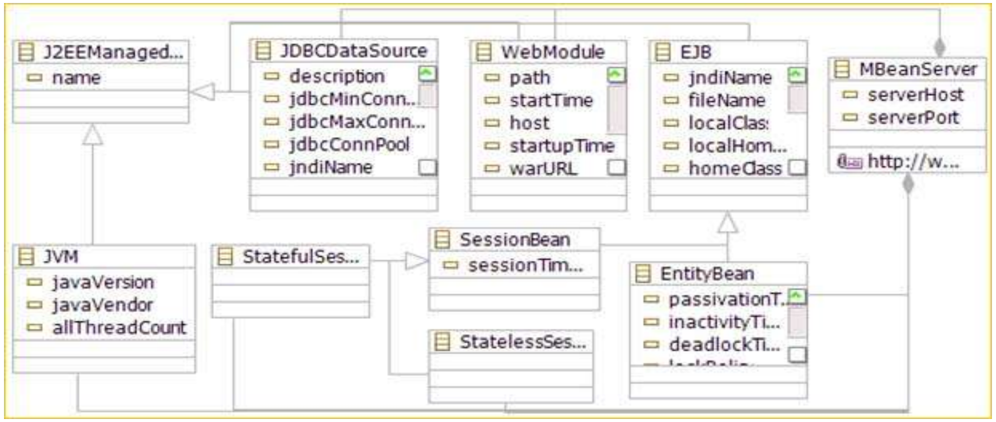


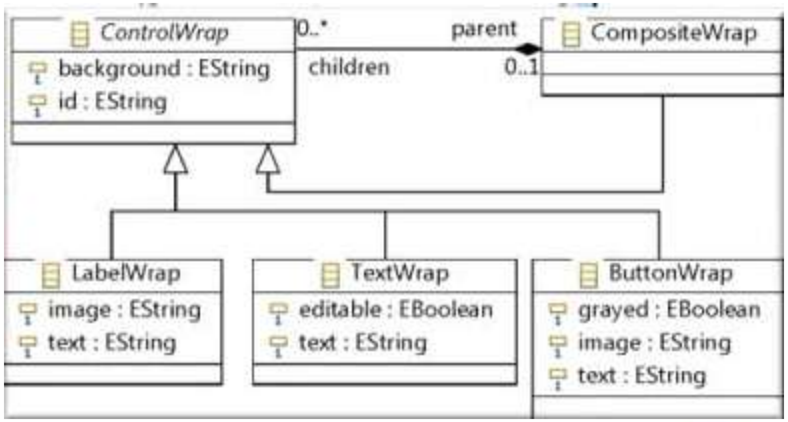
```
public String getJndiName() {
  try{
    MEJB mainEntry=(MEJB)
      PkuasmanagementPackageImpl
       .getMainEntry();
    ObjectName core=getCore();
    Object res=mainEntry
       .getAttribute(
          getCore(),"jndiName");
    return (String)res;
  }
  catch(Exception e){
    return null;
  }
}

public Integer getMaxInstancePool(){
  try{
    MEJB mainEntry=(MEJB)
      PkuasmanagementPackageImpl
        .getMainEntry();
    ObjectName core=getCore();
    Object res=mainEntry
      .getAttribute(
        getCore(),"maxInstancePool");
    return (Integer)res;
...
```

inherent runtime architecture



Case Studies of Inherent Runtime Architecture



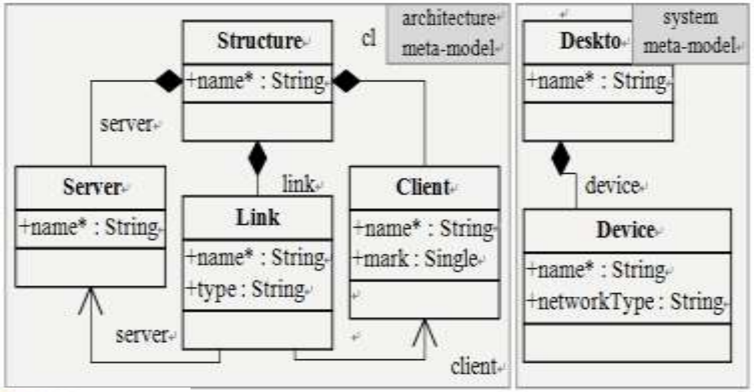


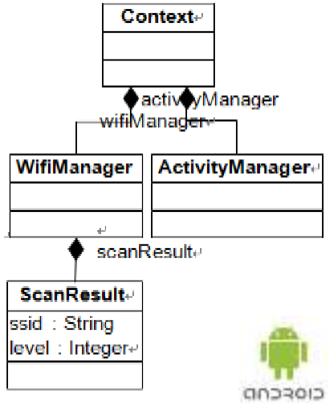
JEE (JonAS/PKUAS, Apusic) Inherent RSA MM

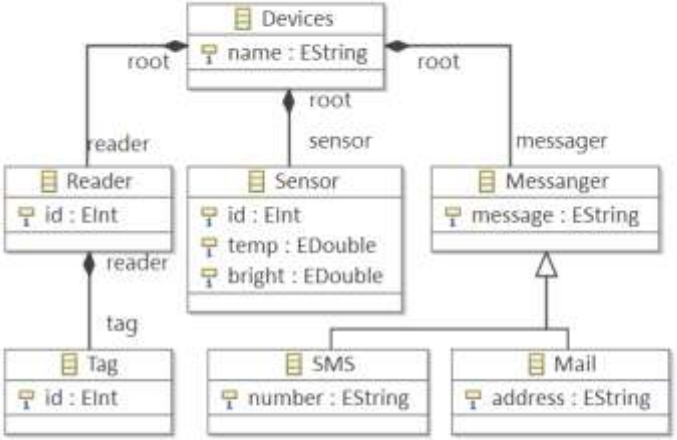
305*ele*+28*map*+310*loc*=22151*loc*

Eclipse SWT

19*ele*+23*map*+178*loc*=11209*loc*







PLASTIC

6*ele*+13*map*+547*loc*=9126*loc*

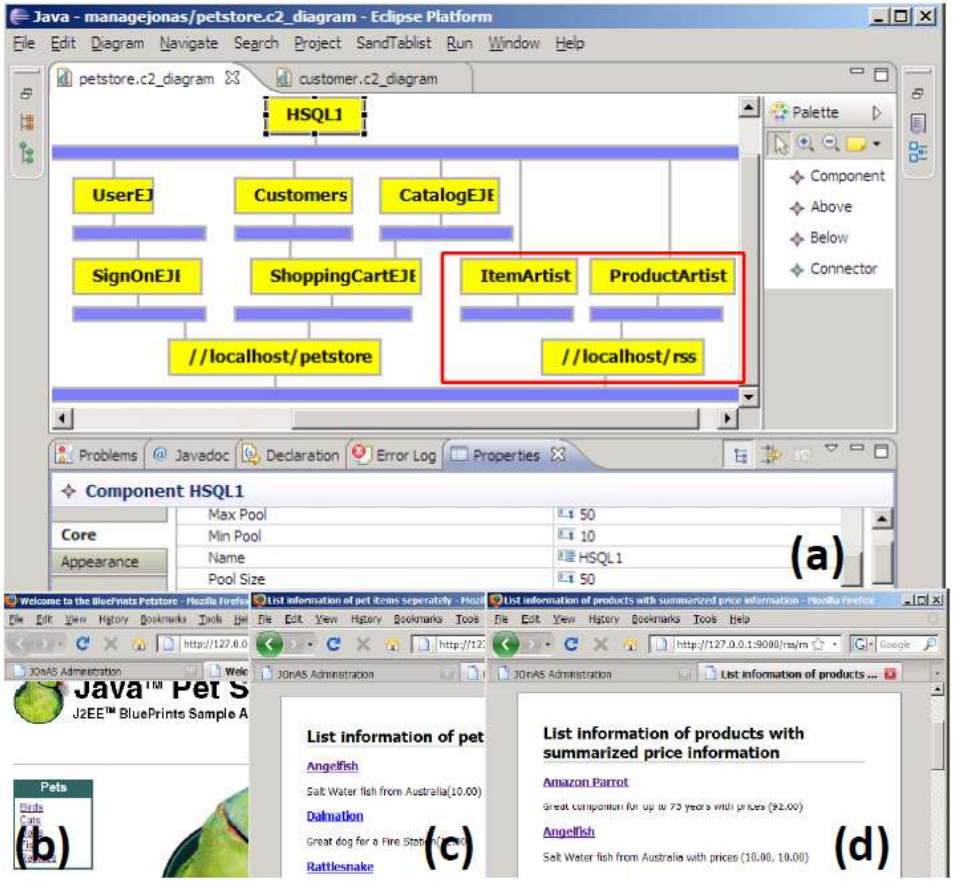
Android

87*ele*+95*map*+431*loc*=21732*loc*

IoT Devices

29*ele*+15*map*+267*loc*=8732*loc*

Construction of Derived Runtime Architecture


Java - managejonas/petstore.c2_diagram - Eclipse Platform
File Edit Diagram Navigate Search Project SandTablist Run Window Help
petstore.c2_diagram
customer.c2_diagram
HSQL1
UserEJ
Customers
CatalogEJE
SignOnEJE
ShoppingCartEJE
ItemArtist
ProductArtist
//localhost/petstore
//localhost/rss
Palette
Component
Above
Below
Connector
Problems
Javadoc
Declaration
Error Log
Properties
Component HSQL1
Core
Appearance
(a)
Welcome to the BluePrints Petstore - Mozilla Firefox
List information of pet items seperately - Mozilla Firefox
List information of products with summarized price information - Mozilla Firefox
JOnAS Administration
Java™ Pet S
J2EE™ BluePrints Sample A
Pets
(b)
(c)
(d)

Max Pool	50
Min Pool	10
Name	HSQL1
Pool Size	50

List information of pet

Angelfish

Salt Water fish from Australia(10.00)

Dalmation

Great dog for a Fire Station(...)

Rattlesnake

List information of products with summarized price information

Amazon Parrot

Great companion for up to 75 years with prices (92.00)

Angelfish

Salt Water fish from Australia with prices (10.00, 10.00)

if the JEE administrators prefer C2-style architecture for runtime evolution, then they have to derive C2 RSA from the inherent one

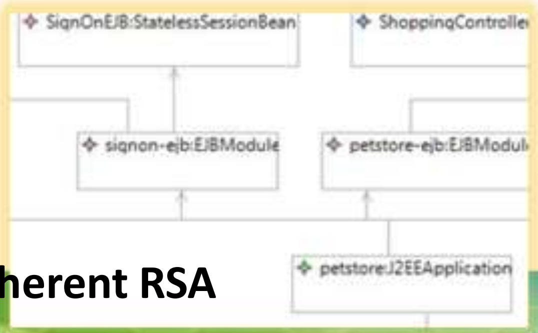
derived runtime architectures (satisfy management needs)

causally connected via model synchronization

inherent runtime architecture (conform to design architecture)

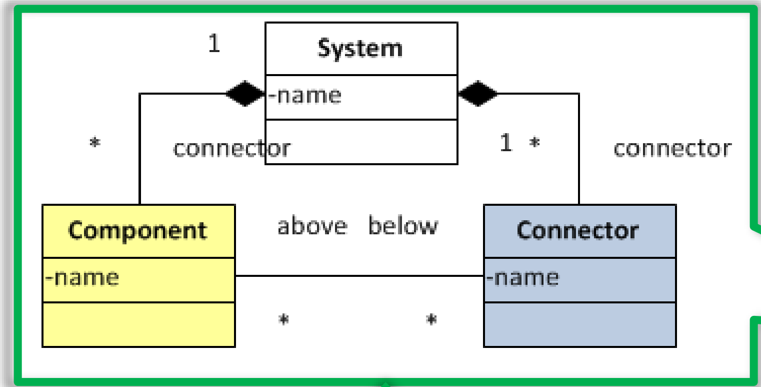
a system at runtime

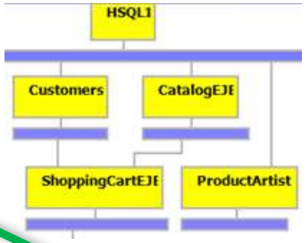
causally connected via automatically generated reflective code



inherent RSA

Construction of Derived Runtime Architecture





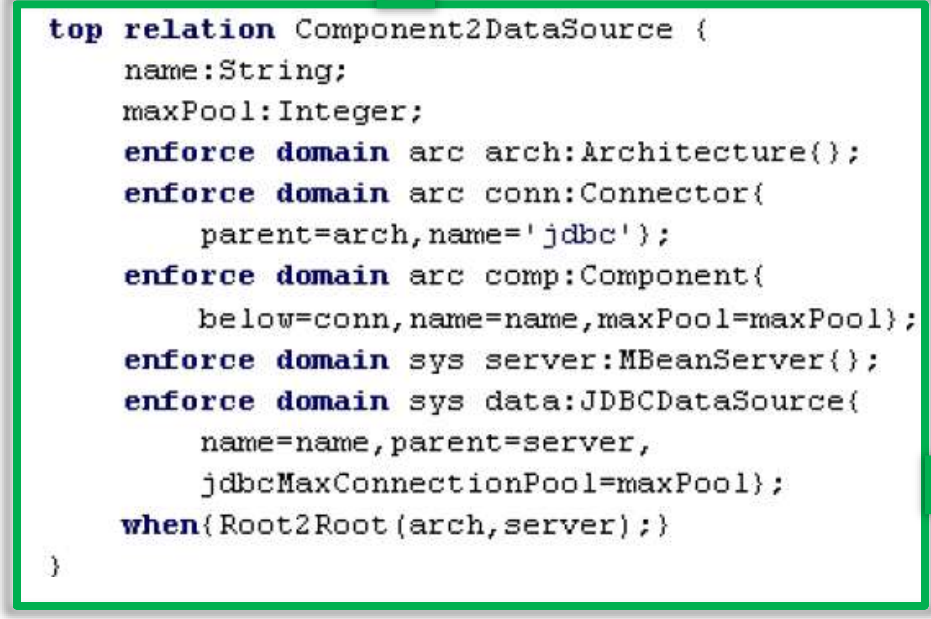
1) define MOF-based meta model of

acquired runtime architectures (satisfy management needs)

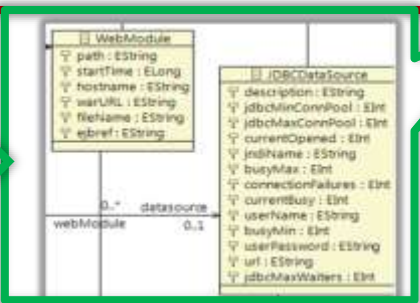
causally connected via model synchronization

inherent runtime architecture (conform to design architecture)

3) define QVT-based mapping between the two meta models



```
top relation Component2DataSource {
    name:String;
    maxPool:Integer;
    enforce domain arc arch:Architecture{};
    enforce domain arc conn:Connector{
        parent=arch,name='jdbc'};
    enforce domain arc comp:Component{
        below=conn,name=name,maxPool=maxPool};
    enforce domain sys server:MBeanServer{};
    enforce domain sys data:JDBCDataSource{
        name=name,parent=server,
        jdbcMaxConnectionPool=maxPool};
    when{Root2Root(arch,server);}
}
```



2) select the target meta model of

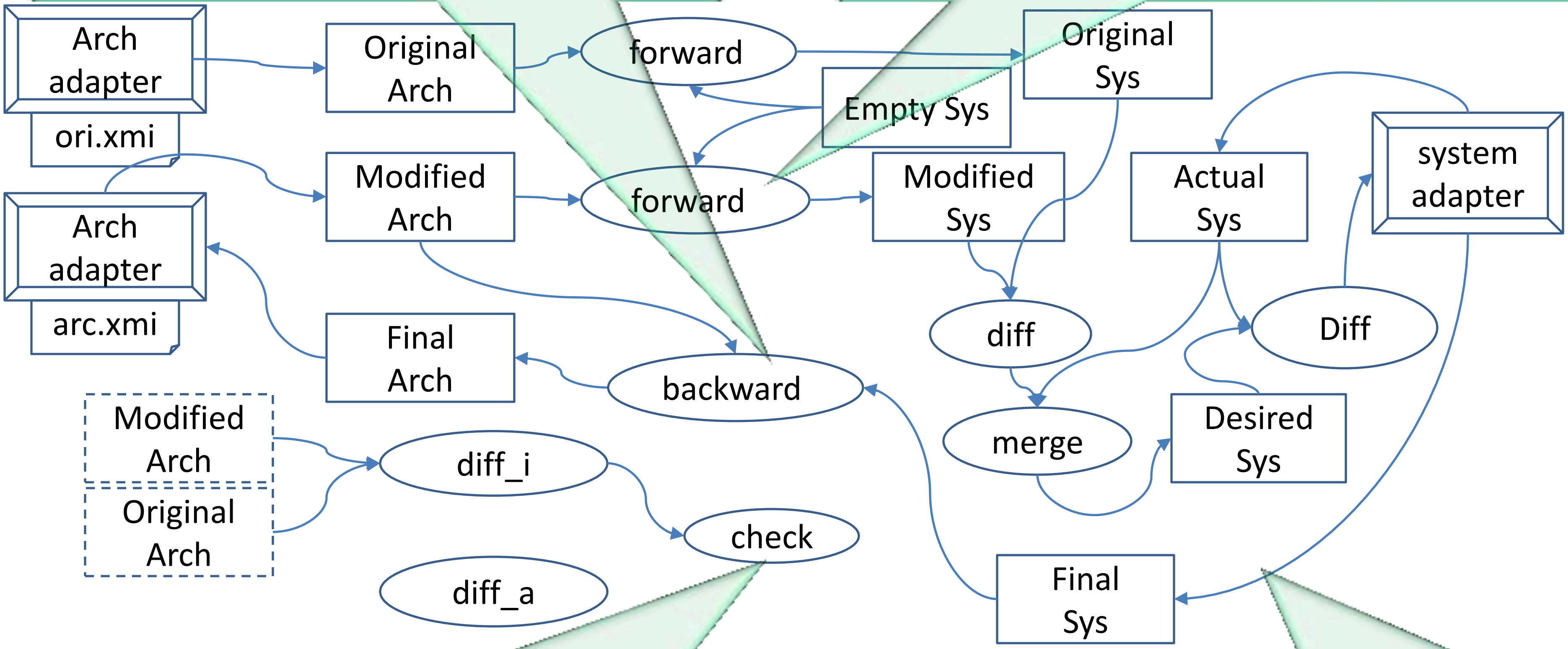
JOnAS

Construction of Derived Runtime Architecture

4) synchonize two models via incremental bi-directional model transformation

3) Backward transformation to feed the sync result back to the view

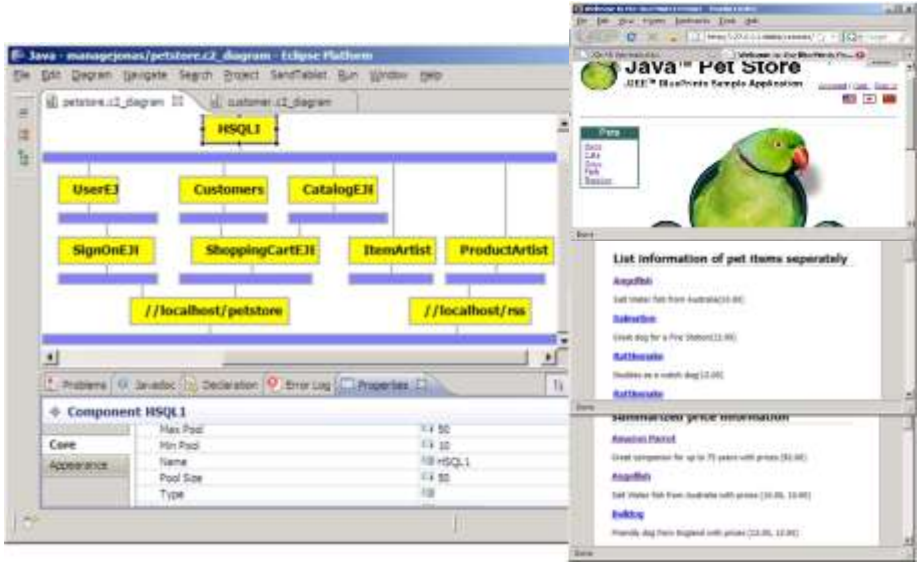
1) Forward transformation to calculate the meaning of view changes



4) Recheck if all the original view modifications have been propagated

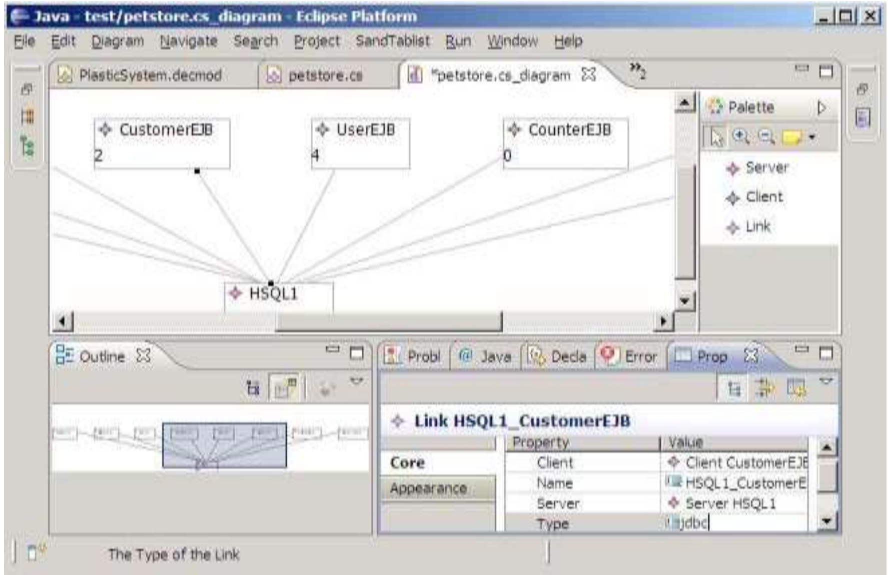
2) CVS-like three-way comparison to filter out conflicts

Case Studies of Derived Runtime Architecture



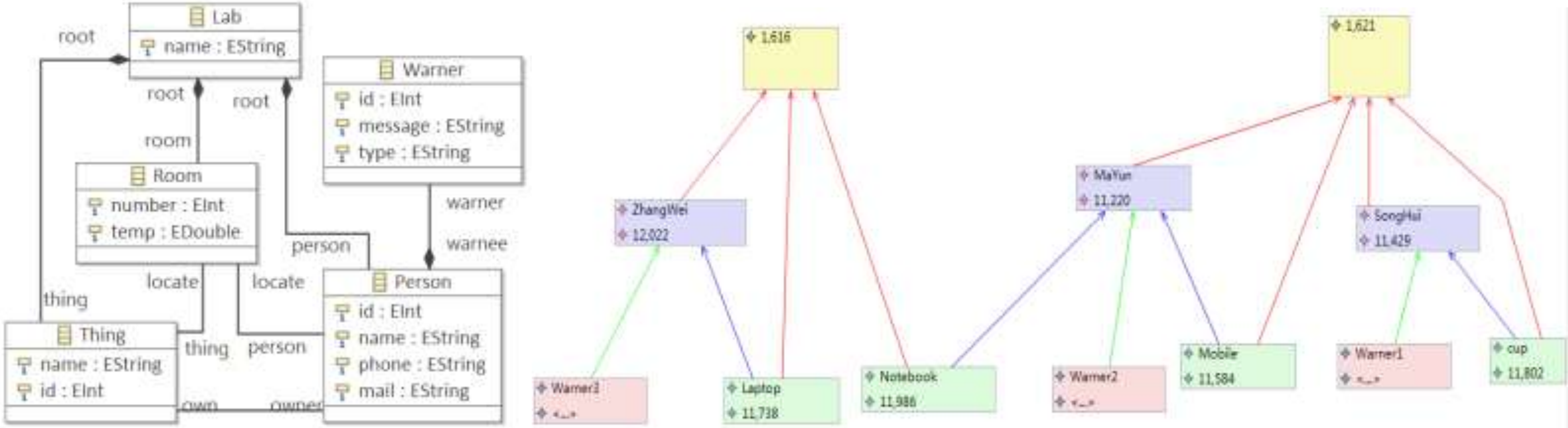
Acquired C2 RSA on JEE

29*ele*+157*map*



Garlan's C/S RSA on JEE

17*ele*+73*map*

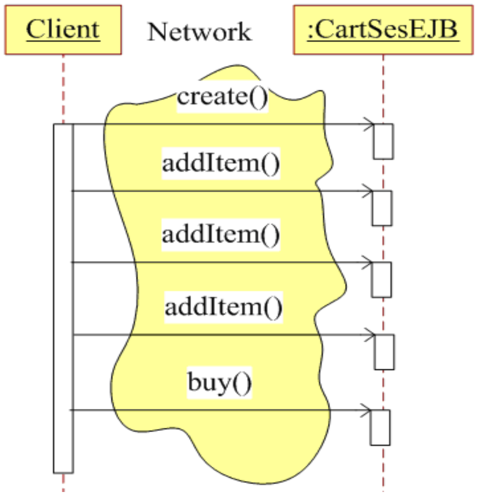


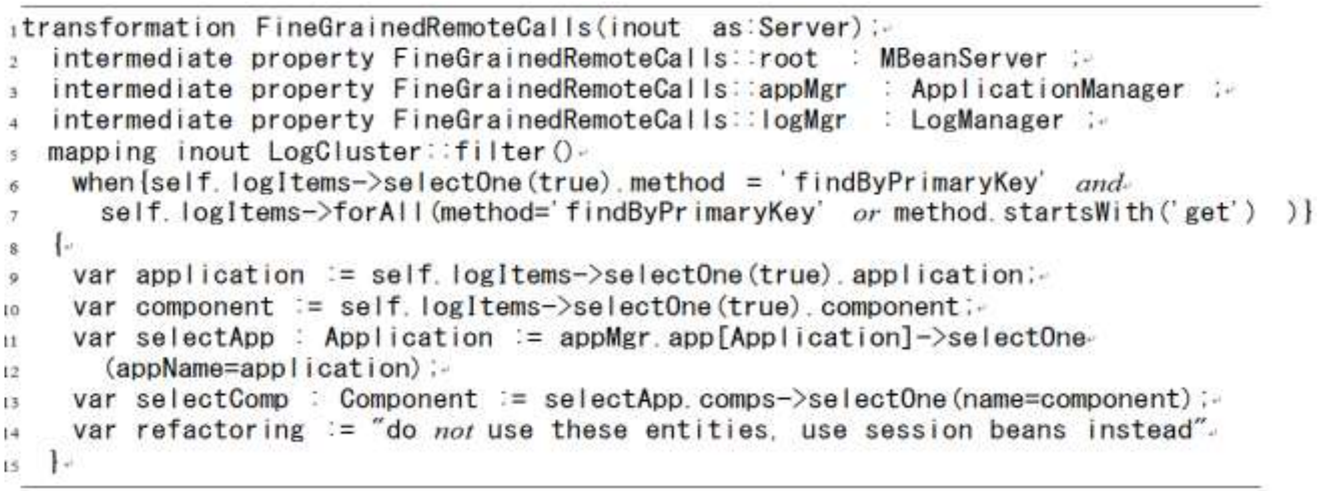
SM@RT Lab on IoT

36*ele*+20*map*

QVT-based RSA Manipulation Language

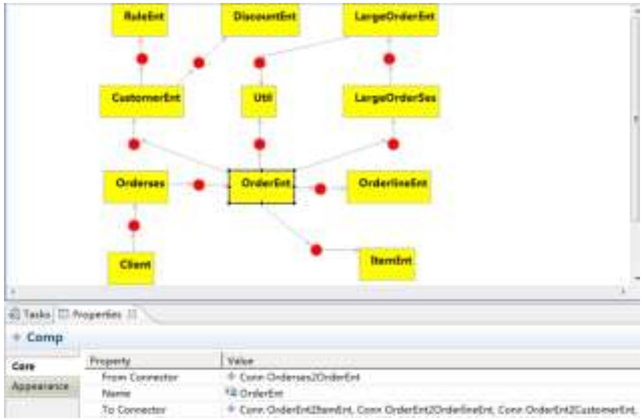
detect 38 anti-patterns in 3 JEE systems: 8/ECperf, 7/Petstore, 5/Rubis





```
transformation FineGrainedRemoteCalls(inout as:Server);
 intermediate property FineGrainedRemoteCalls::root  : MBeanServer ;
 intermediate property FineGrainedRemoteCalls::appMgr  : ApplicationManager ;
 intermediate property FineGrainedRemoteCalls::logMgr  : LogManager ;
 mapping inout LogCluster::filter()
  when{self.logItems->selectOne(true).method = 'findByPrimaryKey' and
    self.logItems->forAll(method='findByPrimaryKey' or method.startsWith('get') )}
 {
  var application := self.logItems->selectOne(true).application;
  var component := self.logItems->selectOne(true).component;
  var selectApp : Application := appMgr.app[Application]->selectOne
    (appName=application);
  var selectComp : Component := selectApp.comps->selectOne(name=component);
  var refactoring := "do not use these entities, use session beans instead"
 }
```

calculate component's impact on system reliability using SBRA

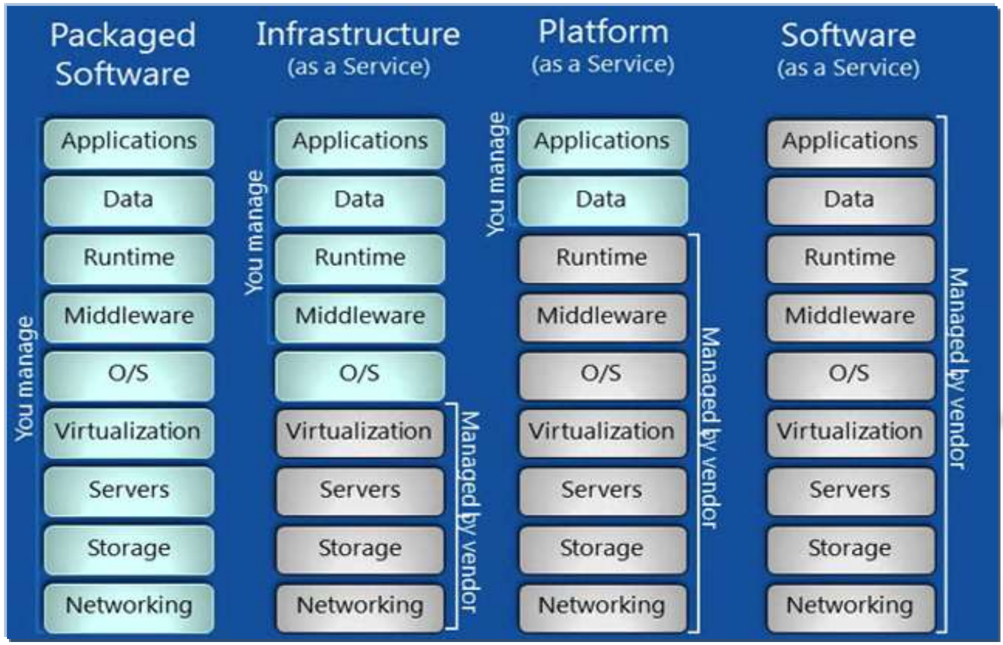


Property	Value
From Connector	Conn Orderses2OrderEnt
Name	OrderEnt
To Connector	Conn OrderEnt2ItemEnt, Conn OrderEnt2OrderlineEnt, Conn OrderEnt2CustomerEnt,

```
transformation Locate_Key_Comp(in jonas:JOnAS)   main(){
 var components := jonas.rootObjects[Server].selectOne(true).map toComponents;
 var scenarios := components.map toScenarios;
 var cdg := new SBRA().Create_CDG(components,scenarios);
 var compNames:=components.collect(name);
 var key='‘;  var maxReliability:= new SBRA().Algo();
 compNames->forEach(curr){   var cdg_cp := cdg.clone();
    var comp:=cdg_cp.components->SelectOne(name=curr);
    comp.reliability:=0.5 + comp.reliability/2; //increasing the reliability of a given component
    var reliability:=new SBRA().Algo(cdg_cp); //re-calculate the system reliability
    if(reliability>maxReliability){maxReliability:=reliability; key:=curr;  }
 } return curr;  }
```

Future Work: SM@RT Cloud (as killer app)

If system management is the core of business, it may be a killer app of SM@RT.

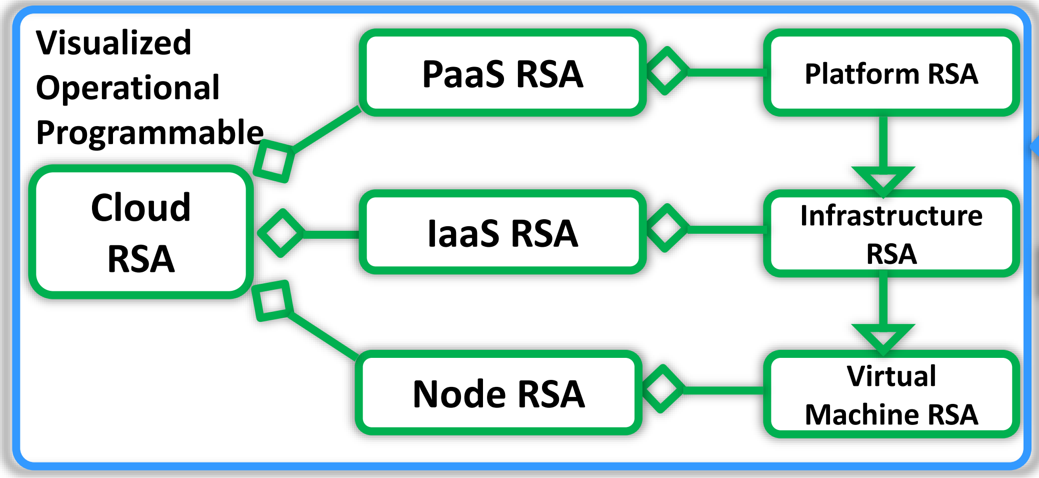


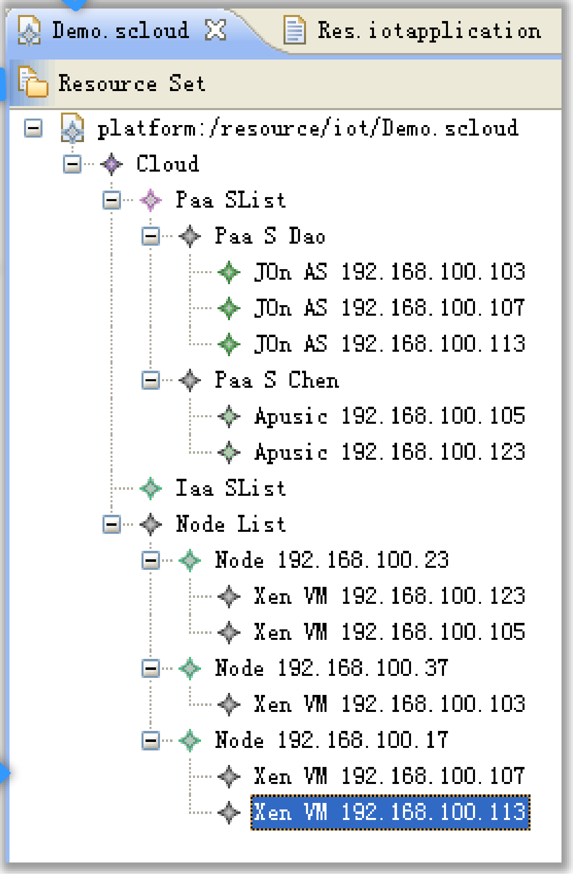
model analysis and planning

model @runtime

abstraction

meta model



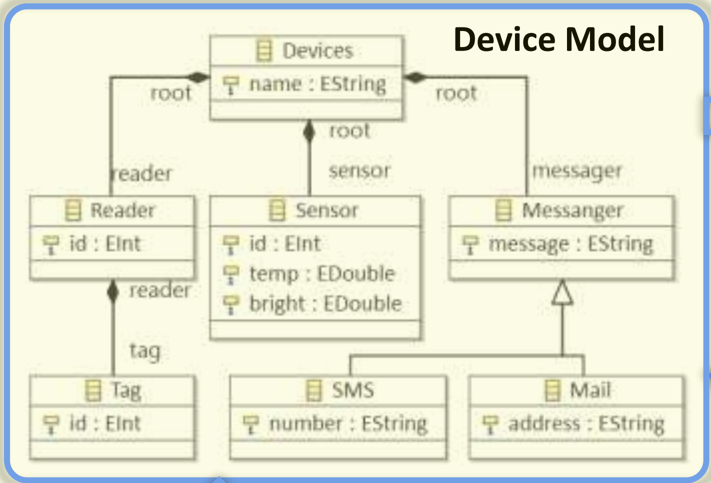


```
platform:/resource/iot/Demo.scloud
  Cloud
    PaaSList
      PaaS Dao
        JOnAS 192.168.100.103
        JOnAS 192.168.100.107
        JOnAS 192.168.100.113
      PaaS Chen
        Apusic 192.168.100.105
        Apusic 192.168.100.123
    IaaSList
    Node List
      Node 192.168.100.23
        Xen VM 192.168.100.123
        Xen VM 192.168.100.105
      Node 192.168.100.37
        Xen VM 192.168.100.103
      Node 192.168.100.17
        Xen VM 192.168.100.107
        Xen VM 192.168.100.113
```

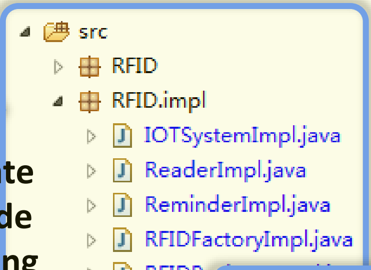
Future Work: SM@RT IoT (as killer app)

If online evolution is the core of business, it may be a killer app of SM@RT.

Device Model



generate the code accessing the devices



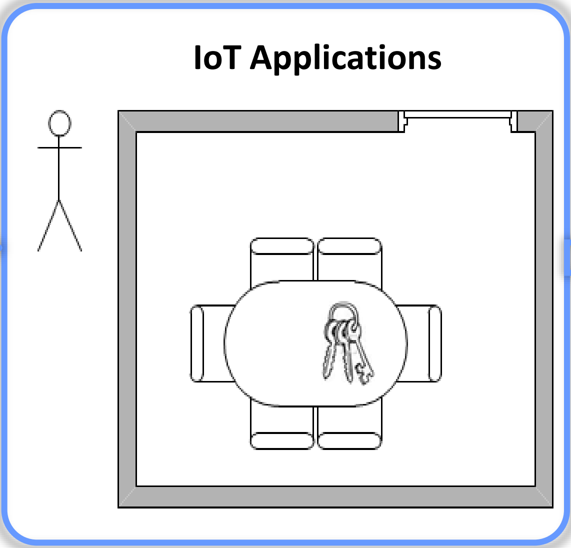
- src
  - RFID
  - RFID.impl
    - IOTSystemImpl.java
    - ReaderImpl.java
    - ReminderImpl.java
    - RFIDFactoryImpl.java
    - RFIDPackageImpl.java
    - Sensor
    - TagImp
  - RFID.util

Auto synchronize the app models and device models and access the devices if necessary

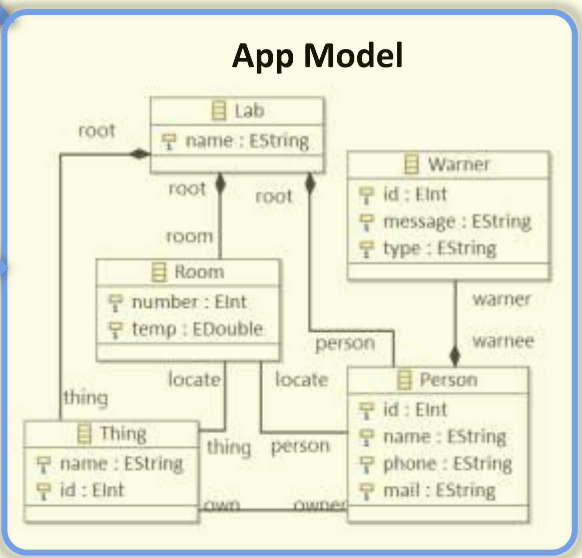
App-Device Mapping

```
top relation RTHP{
 rid:Integer; tid:Integer; rs:RFIDRoot; ra:LabRoot;
 sys reader:Reader{id=rid,root=rs};
 sys tag:Tag{id=tid,reader=reader};
 app house:Room{number=rid,root=ra};
 app person:Person{id=tid,root=ra,locate=house};
 when{RR(rs,ra) and ra.person.id->includes(tid);}  }
```

IoT Applications



App Model



Thank You!