



X-Series

大规模软件开发工具集

赫杰辉

X-Series是什么

▲ 一套轻量级的框架

易于使用

易于集成

易于测试

最合适的

▲ 解决大规模软件开发难题

沟通

文档

学习曲线

工欲善其事
必先利其器

开发人员到底想要什么

- ▲ 挑战
- ▲ 根因
- ▲ 需求
- ▲ 解决之道
- ▲ 未来

你不知道你不知道

关于开发的一点 感性认识

期待中的架构



实际的感觉



别人家的程序员



咱。。。。



▲ 大公司开发综合症

文档迷宫

- 文档很难反应最新的需求
- 文档缺乏关键实现细节
- “Show me the code” – with bugs

源代码泥潭

- 冗长的类，冗长的方法，巨大的问题
- 超大，超长，超宽的嵌套条件分支
- 硬编码的对象组装逻辑

工具之痛

- 大多数工具做的是和开发无关的周边管理工作
 - 编译，持续集成，源码管理，发布
- 管理工具越多，开发工作越艰难
 - 为了统一目前过多的解决方案，我们又做了一个类似的
- 对新工具/框架/标准/语言保持清醒
 - 真的有新东西吗？还只是重复解决已经被现有方法解决了的问题？

一边指责问题，一边重复错误

- 如何理解百万，千万代码行级别的系统
- 软件编码的方式50年不变，过去的给现在的挖坑，现在的给将来的挖坑

根因

▲ 开发其实是个翻译的过程

需求 → 设计（有吗？） → 代码

哪里有翻译，哪里就有误解

在抽象层间存在细节的增强和丢失

▲ 需求翻译（理解）

产品经理/商业用户不懂代码

开发人员懂不懂需求很难讲

▲ 设计翻译

对象图的局限

- 显示实体间的关系而不是动作如何完成 [错误答案]

时序图的局限

- 仅能描述特定执行路径，而无法直观表述分支/循环 [不圆满答案]

不幸的是这些图仍然需要进一步翻译，而且图和代码本质上没联系

▲ 代码的局限

任何编程方式都无法解决代码理解问题

需求

▲ 开发并不仅仅意味着写代码

我们解决所有问题都是用同一个原始的手段
事实上不同性质的问题需要不同的手段来解决

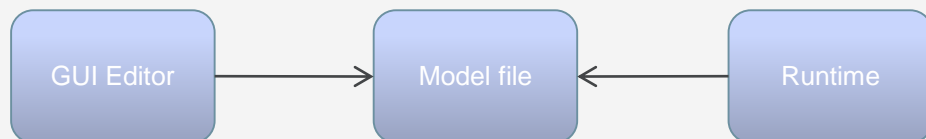
▲ 我们需要人人都懂，无需翻译的媒介

百闻不如一见 — 能可视化系统顶层模型

- 行为
- 决策
- 状态

基于模型而不是代码开发

- 将业务模型和数据模型从代码里面解放出来
- 直接使用模型替代代码生成
- 将模型与代码相关联



不要条件反射式的解决问题

解决之道

▲ Xross unit

专注描述工作如何完成的高层流程

服务级别

▲ Xross Decision tree

为复杂决策建模

模块级别

▲ Xross state

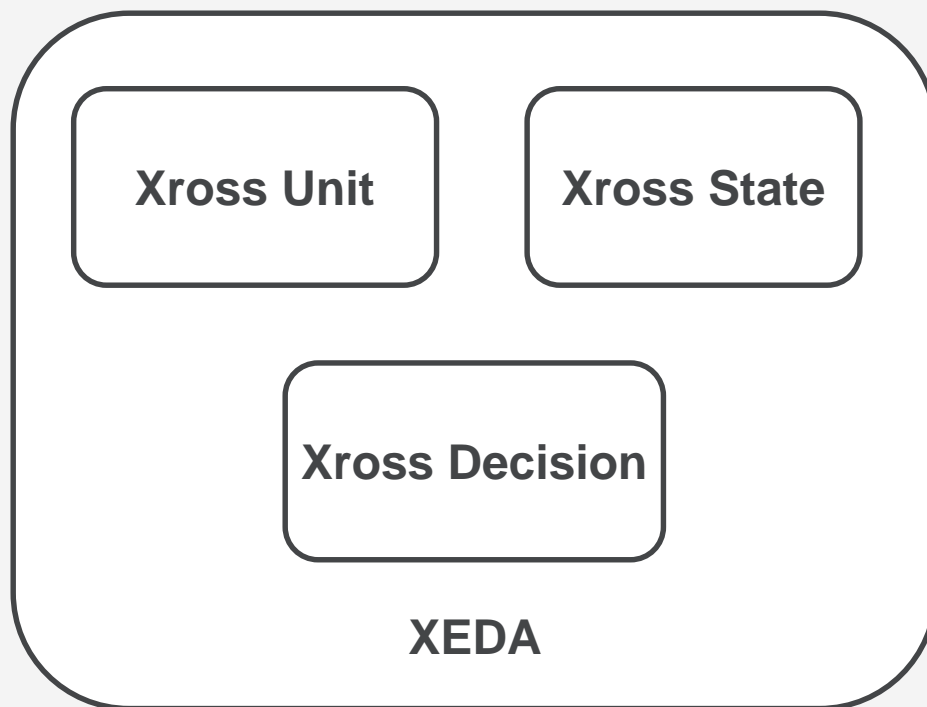
按照状态组织业务流程

领域级别

▲ Xeda

SEDA model

运行平台级别



Xross Unit

Xross Unit

Resource - com.xross.test.new_xross_unit.xunit - Eclipse Platform

File Edit Navigate Search Project Run Downloads Server Window Help

100%

Project Explorer

- src
- (default package)
- com.ebay.xdomain.xcomponent
- com.ebay.xdomain.xunit
- com.ebay.xdomain.xunit.app
- com.ebay.xdomain.xunit.converters
- com.ebay.xdomain.xunit.decorators
- com.ebay.xdomain.xunit.locators
- com.ebay.xdomain.xunit.processors
- com.ebay.xdomain.xunit.validators
- JRE System Library [JavaSE-1.6]
- com.ebay.tools.decisiontree_1.0.0.20120
- JUnit 3

Outline

- a chain
- a decorator
- processor
- a branch
- locator
- key1 : node 1
- not specified : a bibranch
- not specified : while loop
- not specified : do-while loop
- an adapter
- while loop
- validator
- true : do-while loop
- a branch
- locator
- key 1 : a bibranch
- validator
- true : a chain
- a decorator
- a chain
- processor
- a branch
- locator
- not specified : key2 : node 2
- key5 : node 3
- a chain
- a chain
- unit 1
- a branch
- locator
- not specified : an adapte
- conve
- not specified : key 1 : unit 2
- unit 3
- unit 1
- unit 2
- unit 3
- validator

Task List

DelayProcess.java MyUnitApp.java TimeCostMonitor.java test.decisiontree new_xross_unit.xunit

Select

- Marquee
- Processor
- Converter
- Validator
- Locator
- Chain
- If/else
- Branch
- While loop
- Do while loop
- Start
- End
- Decorator
- Adapter

a chain

processor node 1

processor valid node

processor invalide node

processor update sum

processor update sum

processor unit 2

decorator

adapter

while loop

Tasks Properties Console JUnit

Property	Value
Misc	
Description	
Name	rrr
packageId	
Properties	
key	ssss

Start

C:\yide-1.2.0-qa-1\dropl... C:\Temp\decisiontree1 Debug - com.ebay.tools... Editor operation - Tools ... Resource - com.xross...

1:00 AM Thursday

Xross Unit

▲ Xross unit 编辑器是一个灵活的系统构建器

使用流程图构建系统

- 在Eclipse里面所见即所得的方式

提供丰富的行为组件

- 超精简接口 – processor, converter, validator, locator

提供丰富的结构组件

- chain, if-else, branch, while, do while loop, **decorator**, **adapter**

编辑方法自然

- 拖放和对象组合 – E.g. Validator + Unit = if/else structure

可配置

- 可以在应用或构建单元层次上面配置参数

▲ 一图胜千言

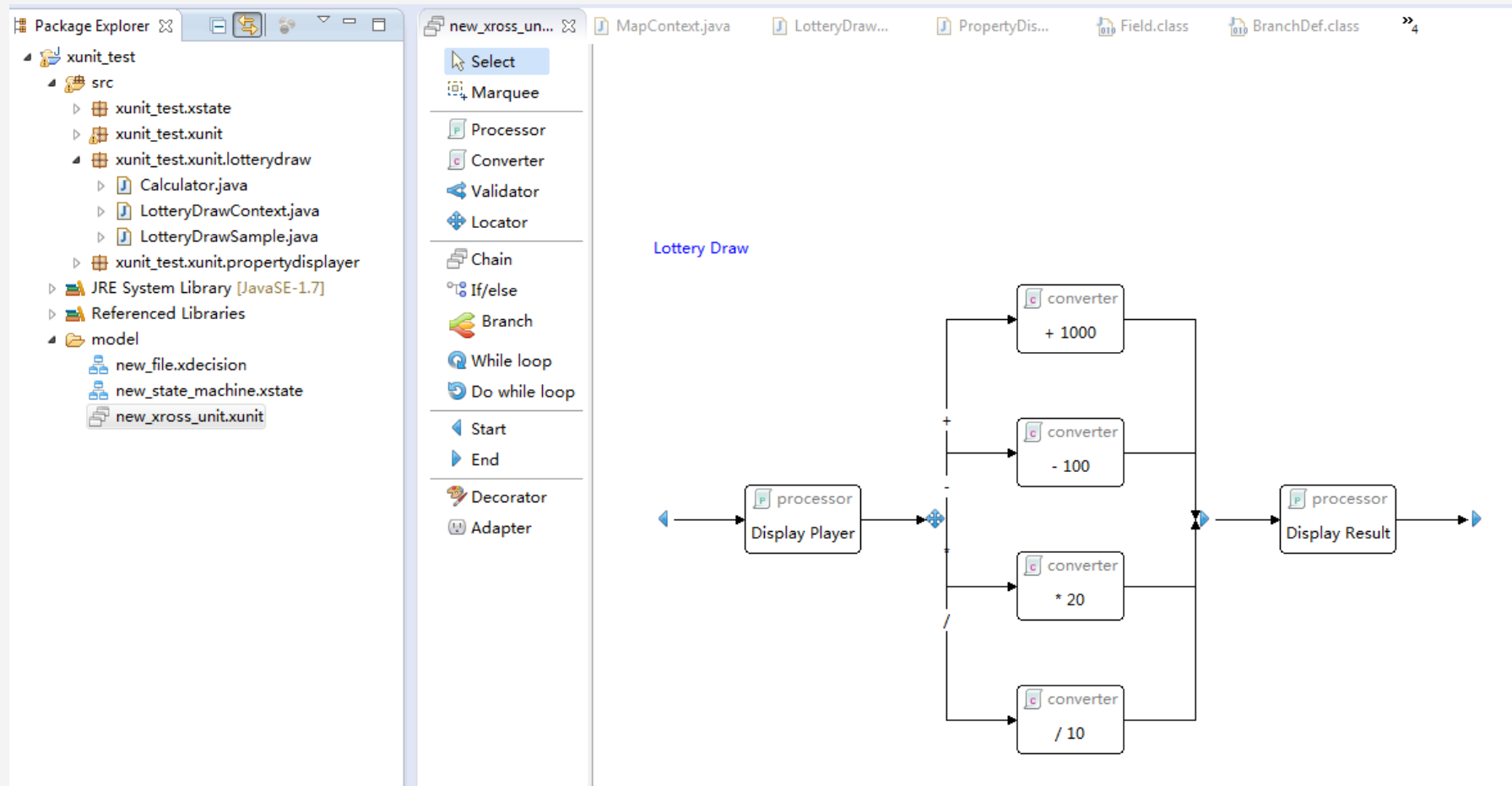
超越传统的开发模式，从代码和配置的汪洋里解脱出来

模型归模型，代码归代码，查看代码仅需双击进入

Xross Unit 示例

▲ 生成系统蓝图

你可以一直和PM, PD, QA一起优化修改讨论



Xross Unit 示例

▲ 创建组件单元

函数式接口易于实现和测试

```
package xunit_test.xunit.lotterydraw;

import java.util.Map;

public class Calculator implements Converter, UnitPropertiesAware {
    private double delta;
    private String operation;

    @Override
    public Context convert(Context arg0) {
        LotteryDrawContext ctx = (LotteryDrawContext)arg0;

        double value = ctx.quantity;

        switch(operation){
            case "+": value+=delta; break;
            case "-": value-=delta; break;
            case "*": value*=delta; break;
            case "/": value/=delta; break;
        }

        ctx.quantity = value;

        return ctx;
    }

    @Override
    public void setUnitProperties(Map<String, String> arg0) {
        delta = Double.parseDouble(arg0.get("delta"));
        operation = arg0.get("operation");
    }
}
```

Xross Unit 示例

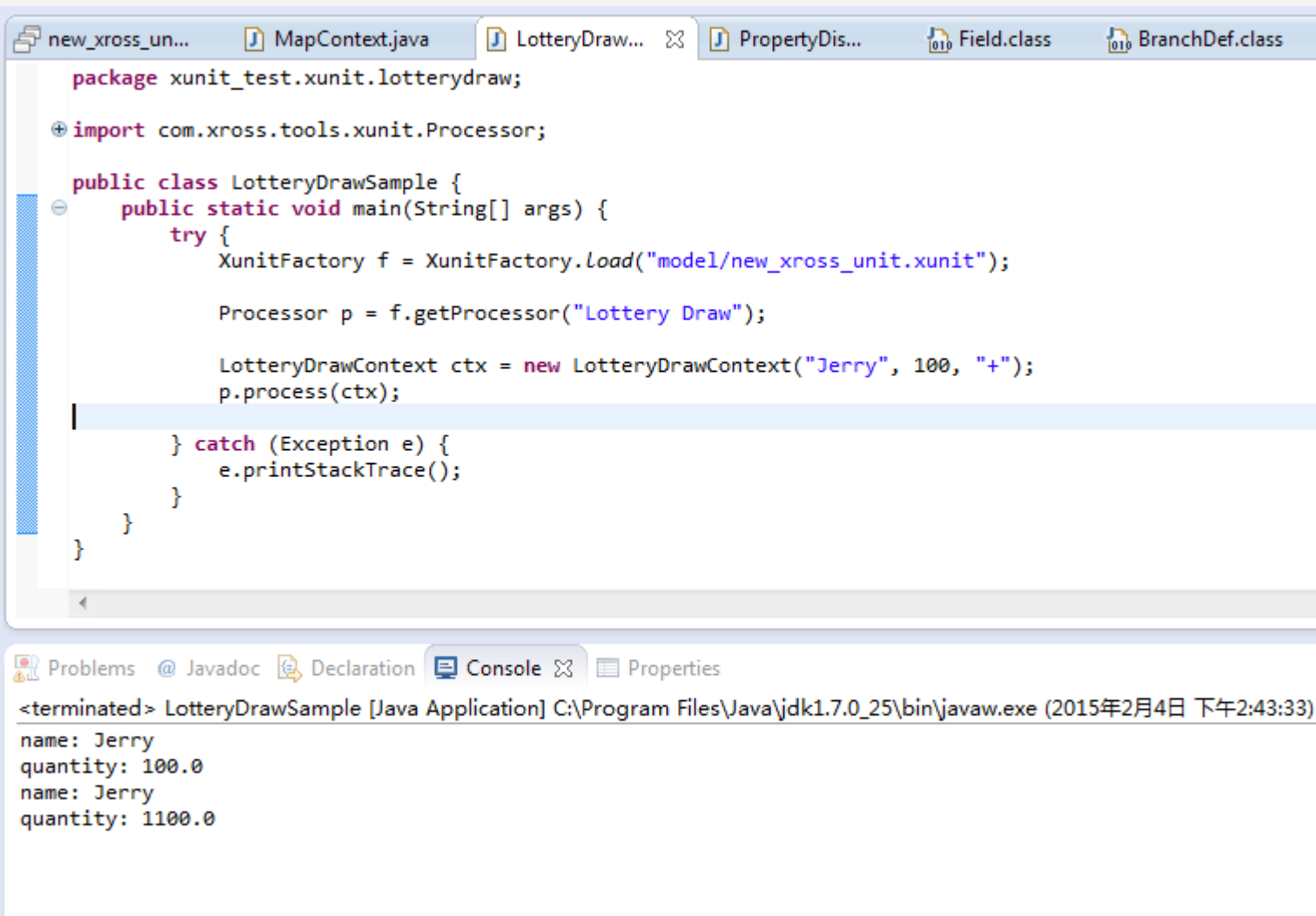
结合代码和系统蓝图，配置参数

The screenshot displays the Xross Unit IDE interface. On the left, a sidebar contains a list of components: Chain, If/else, Branch, While loop, Do while loop, Start, End, Decorator, and Adapter. The main workspace shows a system blueprint titled "Lottery Draw". This blueprint consists of two parallel paths that merge. The top path contains a "converter" block with the value "+ 1000". The bottom path contains a "converter" block with the value "- 100". An "Open Type" dialog box is open in the foreground, prompting the user to "Enter type name prefix or pattern (*, ?, or camel case):". The text "xunit_test.xunit.lotterydraw.Calculator" is entered in the input field. Below the input field, the "Matching items:" section lists "Calculator - xunit_test.xunit.lotterydraw". At the bottom of the dialog, there is a search bar containing "xunit_test.xunit.lotterydraw - xunit_test/src" and an "OK" button. Below the IDE window, a table displays the properties of the selected unit.

Property	Value
▲ Misc	
Class	xunit_test.xunit.lotterydraw.Calculator
Description	
Name	+ 1000
Reference unit	
▲ Properties	
delta	1000
key	income

Xross Unit 示例

▲ Rock'n Roll



The screenshot shows an IDE with several open files: `new_xross_un...`, `MapContext.java`, `LotteryDraw...`, `PropertyDis...`, `Field.class`, and `BranchDef.class`. The `LotteryDraw...` file is active and contains the following Java code:

```
package xunit_test.xunit.lotterydraw;

import com.xross.tools.xunit.Processor;

public class LotteryDrawSample {
    public static void main(String[] args) {
        try {
            XunitFactory f = XunitFactory.load("model/new_xross_unit.xunit");

            Processor p = f.getProcessor("Lottery Draw");

            LotteryDrawContext ctx = new LotteryDrawContext("Jerry", 100, "+");
            p.process(ctx);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The console output at the bottom shows the execution results:

```
<terminated> LotteryDrawSample [Java Application] C:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2015年2月4日 下午2:43:33)
name: Jerry
quantity: 100.0
name: Jerry
quantity: 1100.0
```

Xross Unit

趁手的工具是原则保证的利器

▲ 快速组建系统

自顶向下分解，组件化设计，流水线式开发

最优化设计复用

快速切换开发焦点

▲ 高内聚，低耦合

通过名字描述功能

通过配置调整行为

通过Context限定数据

每个unit仅仅完成明确描述的功能

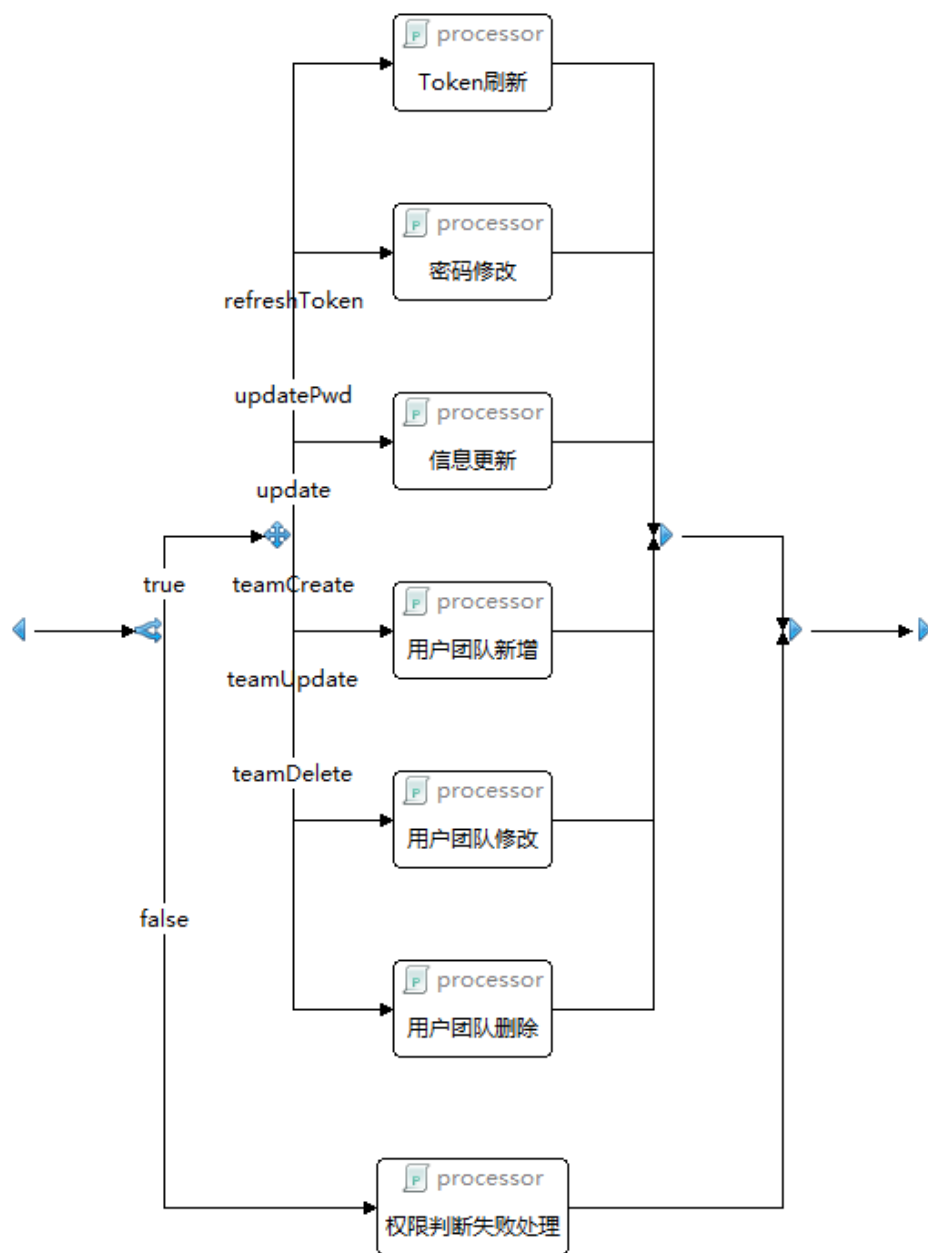
▲ 易于单元化测试

单接口设计，无选择，无歧义的实现

通过构造Context，轻松模拟测试数据

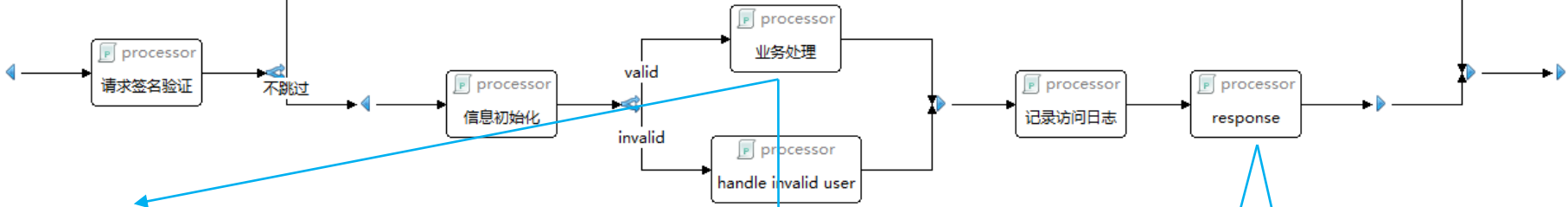
实际案例

security check chain

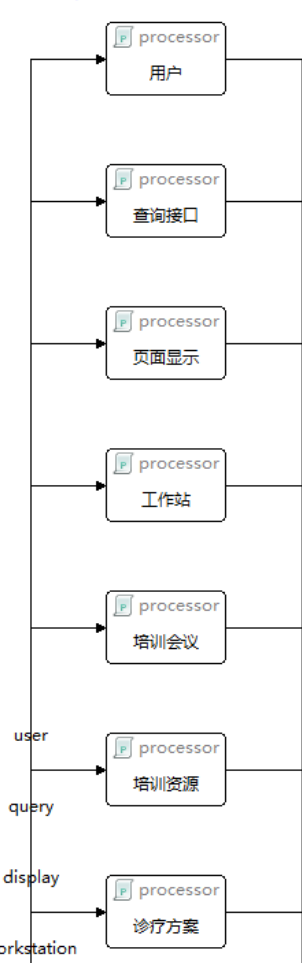


同一模型文件的子图引用

main



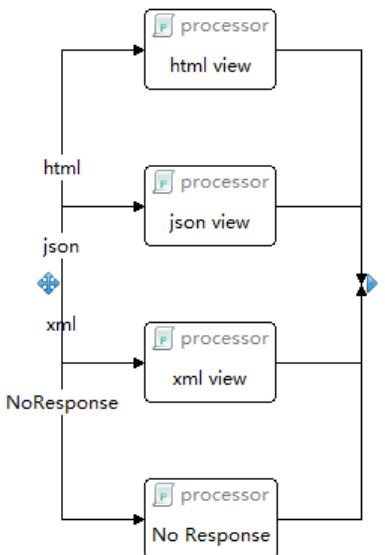
module process branch



内置功能

Property	Value
Class	
Description	
Name	response
Reference unit	view branch

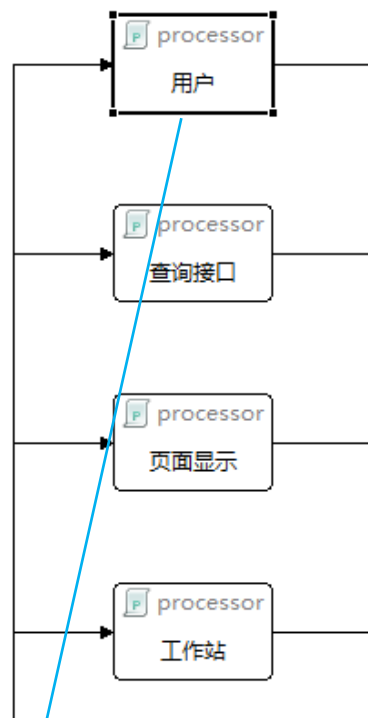
view branch



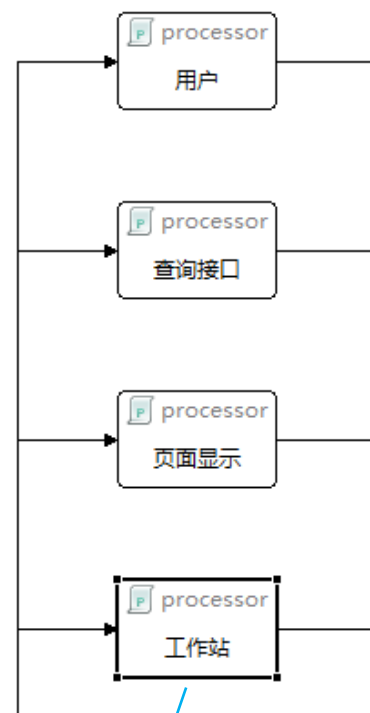
Property	Value
Class	
Description	
Name	业务处理
Reference unit	module process branch

不同模型文件的子图引用

module process branch



module process branch



@ Javadoc Proper... Declar... Servers Search Console Progress History

Property	Value
▲ Misc	
Class	com.medicalmaster.web.control.entry.DistributeProcessor
Description	
Name	用户
Reference unit	
▲ Properties	
module	user
processor	user process

用户的创意用法

Declar... Servers Search Console Progress History Pr

Value
com.medicalmaster.web.control.entry.DistributeProcessor
工作站
workstation
workstation process

关于 Xross Unit更多信息

▲ 不是又一个Spring

Spring: 从整体如何由局部构成的观点构建系统

Xunit: 从请求如何被处理的行为观点构建系统

▲ 不是工作流

工作流处理多角色在多请求之间的任务/路径管理

Xunit 管理一个请求的响应路径/处理单元

▲ 不是一个可视化的编程语言

可视化的编程语言解释和生产代码

Xunit 在业务层组装行为和结构单元

Workflow

Xunit

Spring

Visual
Language

关于 Xross Unit更多信息

▲ 为什么使用单元来完成代码也能做的事情？

因为问题的大小决定手段的选择，想象下面工作的复杂度

- “Hello World”
- 一个Web Service
- 一个小的Web App
- 一个淘宝，ebay，ctrip规模的网站

▲ 为什么不用现有的命令框架

缺乏管理单元的内部细节表示

- Servlet – Command at URL level
- JEE: Session Bean, Entity Bean, Message Bean – Command at bean id level

尽管有大量的小的仅仅只有一页代码的command

但是还是会有少量但是非常重要的command是非常的复杂[80/20原则]

弥补开发层次中缺失的一环



Xross Decision

Decision Tree

▲ 什么是decision tree

商业智能领域常用的决策工具

利用树形模型表达复杂的决策制定过程

▲ Decision Tree 编辑器可以让开发者

生成decision tree

- 以所见即所得的方式

依据模型生成单元测试的验证代码

- 所有决策路径全覆盖

▲ 优势

纯模型，无代码

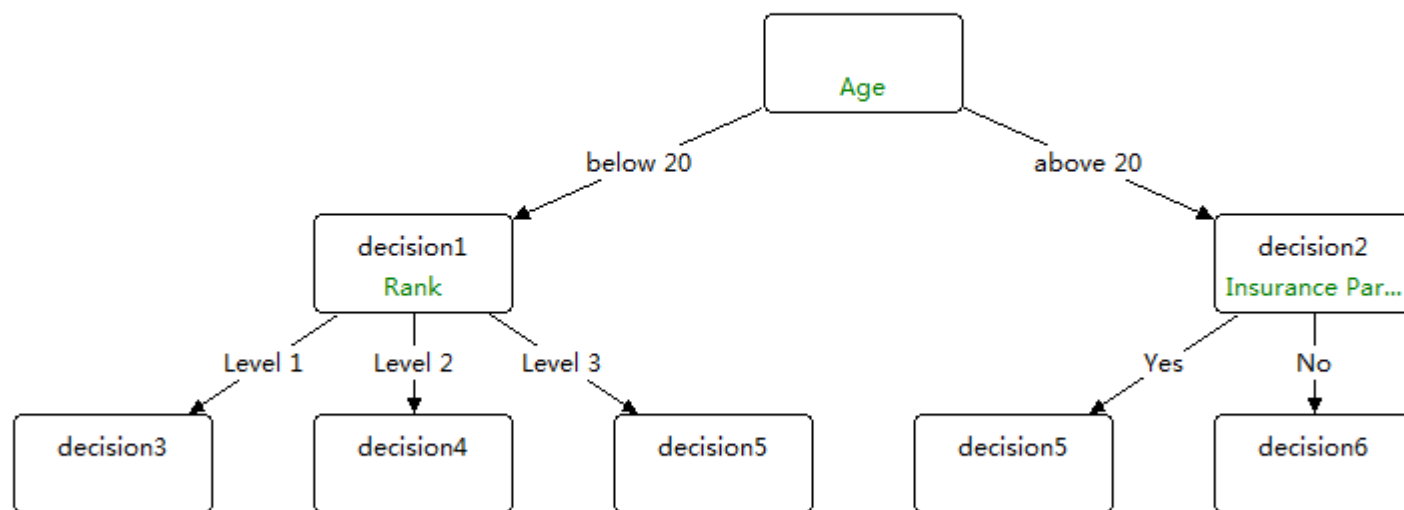
方便重用

替代if/else，极大的简化代码

复杂逻辑无法用条件判断表达

Decision Tree

▲ 直观的表达



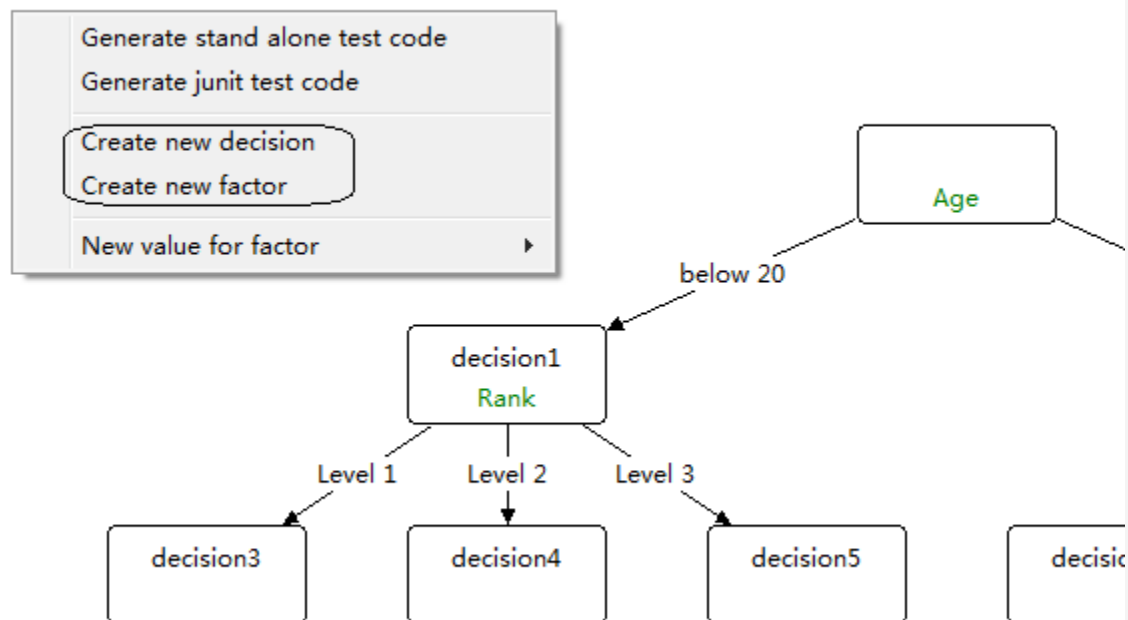
Decision Tree建模

▲ 定义决策的考虑因素

可以有多个取值的变量

▲ 定义决策

代表特定决策的标志符



Decision Tree测试

▲ 生成单元测试

检验模型是否正确运行，示范如何使用工具

The screenshot shows the Xross tools IDE interface. At the top, a menu is open with options: "Generate stand alone test code", "Generate junit test code" (highlighted), "Create new decision", "Create new factor", and "New value for factor". Below the menu, a decision tree diagram is visible with a node labeled "decision3". A status bar indicates "Finished after 0.061 seconds" and "Runs: 1/1", "Errors: 0", "Failures: 0". The bottom panel shows the generated JUnit test code for the decision tree.

```
package com.ebay.xdomain.xcomponent;

import org.junit.Test;

import static org.junit.Assert.*;

import com.xross.tools.xdecision.MapFacts;
import com.xross.tools.xdecision.XDecisionTree;
import com.xross.tools.xdecision.XDecisionTreeFactory;

public class SimpleDecisonTest {
    @Test
    public void testGetDecision(){
        XDecisionTree<String> tree;
        try {
            tree = XDecisionTreeFactory.create(
                "model/SimpleDecison.xdecision");
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }

        //Verify tree
        MapFacts test;

        test = new MapFacts();
        test.set("Age", "below 20");
        assertEquals("decision1", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        assertEquals("decision2", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 1");
        assertEquals("decision3", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 2");
        assertEquals("decision4", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        test.set("Insurance Participation", "Yes");
        assertEquals("decision5", tree.get(test));
    }
}
```

```
package xunit_test.xdecision;

import org.junit.Test;

public class SimpleDecisonTest {
    @Test
    public void testGetDecision(){
        XDecisionTree<String> tree;
        try {
            tree = XDecisionTreeFactory.create("model/SimpleDecison.xdecision");
        } catch (Exception e) {
            e.printStackTrace();
            fail();
            return;
        }

        //Verify tree
        MapFacts test;

        test = new MapFacts();
        test.set("Age", "below 20");
        assertEquals("decision1", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        assertEquals("decision2", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 1");
        assertEquals("decision3", tree.get(test));

        test = new MapFacts();
        test.set("Age", "below 20");
        test.set("Rank", "Level 2");
        assertEquals("decision4", tree.get(test));

        test = new MapFacts();
        test.set("Age", "above 20");
        test.set("Insurance Participation", "Yes");
        assertEquals("decision5", tree.get(test));
    }
}
```

Xross State

Xross State

状态机是系统的核心

▲ 一个允许开发人员创建状态机的编辑器

状态机用处极其广泛

- 订单，用户，任务

通用直观的解决方案

▲ 结合模型和代码

可以创建仅包含状态和变迁的状态机

也可以提供状态变迁时的触发器

▲ 模型可以被工具用于在运行时触发状态转移

所见即所得

Xross State基本模型

Select

Marquee

State Machine

State Node

Start Node

End Node

Transition

```
stateDiagram-v2
    [*] --> create : create
    create --> paused : create
    create --> paid : paid
    create --> highPriority : paid
    paused --> create : create
    paused --> paid : screated
    paid --> paused : screated
    paid --> shipped : shipped
    shipped --> paid : shipped
    shipped --> end : shipped
    highPriority --> paid : shipped
    end --> [*] : 
```

Find

Connect Mylyn
Connect to your task and

Outline

create

paid

shipped

created

paid

start

End node

Paused

High Priority

statemachine xxxbcc

Statemachine1ww

Problems

Javadoc

Declaration

Console

Properties

Property	Value
Entry action	
Exist action	
Id	created
Reference state machine	

Xross State扩展元素

▲ 状态转移触发器

EntryAction

ExitAction

TransitionAction

▲ 状态转移校验

TransitionGuard

```
package xunit_test.xstate;

import com.xross.tools.xstate.EntryAction;

public class TestAction implements EntryAction, ExitAction, TransitAction, TransitionGuard {
    public void transit(String sourceStateId, String targetStateId, Event event) {
        System.out.println(String.format("Transit from %s to %s on %s", sourceStateId, targetStateId, event.getId()));
    }

    public void exit(String sourceStateId, Event event) {
        System.out.println(String.format("Exit from %s on %s", sourceStateId, event.getId()));
    }

    public void enter(String targetStateId, Event event) {
        System.out.println(String.format("Enter into %s on %s", targetStateId, event.getId()));
    }

    public boolean isTransitAllowed(String sourceId, String targetId, Event event) {return true;}
}
```

Xross State 示例

```
package xunit_test.xstate;

import com.xross.tools.xstate.Event;

public class StateTest {
    public static void main(String[] args) {
        try {
            StateMachineFactory f = StateMachineFactory.load("model/new_state_machine.xstate");
            StateMachine sm = f.create("Statemachine1ww");

            print(sm);
            sm.notify(new Event("e1"));

            print(sm);
            sm.notify(new Event("e2"));

            print(sm);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static void print(StateMachine sm) {
        System.out.println("Current state Id: " + sm.getCurrentState().getId());
        System.out.println("Is ended: " + sm.isEnded());
    }
}
```

Problems @ Javadoc Declaration Console Properties

<terminated> StateTest [Java Application] C:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2015年2月6日 下午12:27:35)

Current state Id: start

Is ended: false

Current state Id: S1

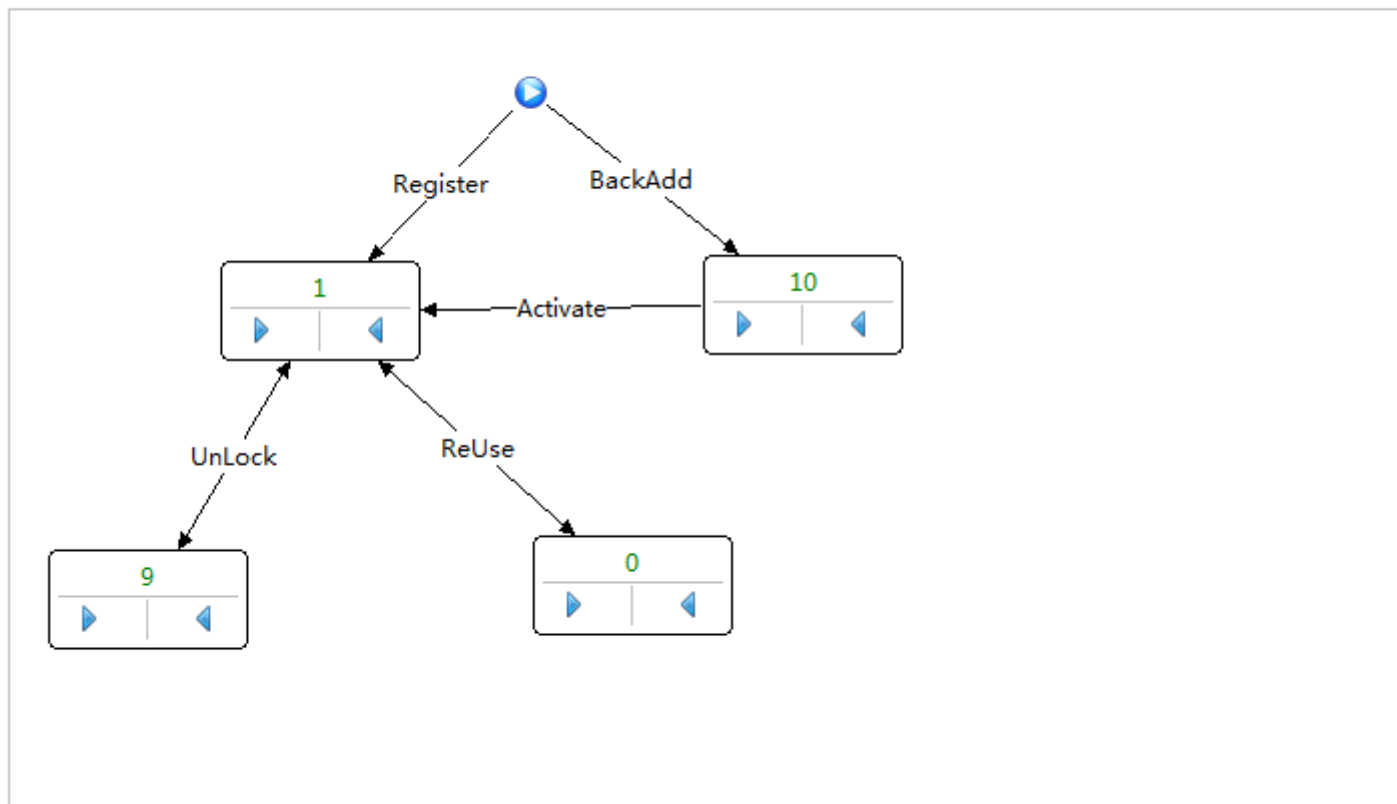
Is ended: false

Current state Id: end

Is ended: true

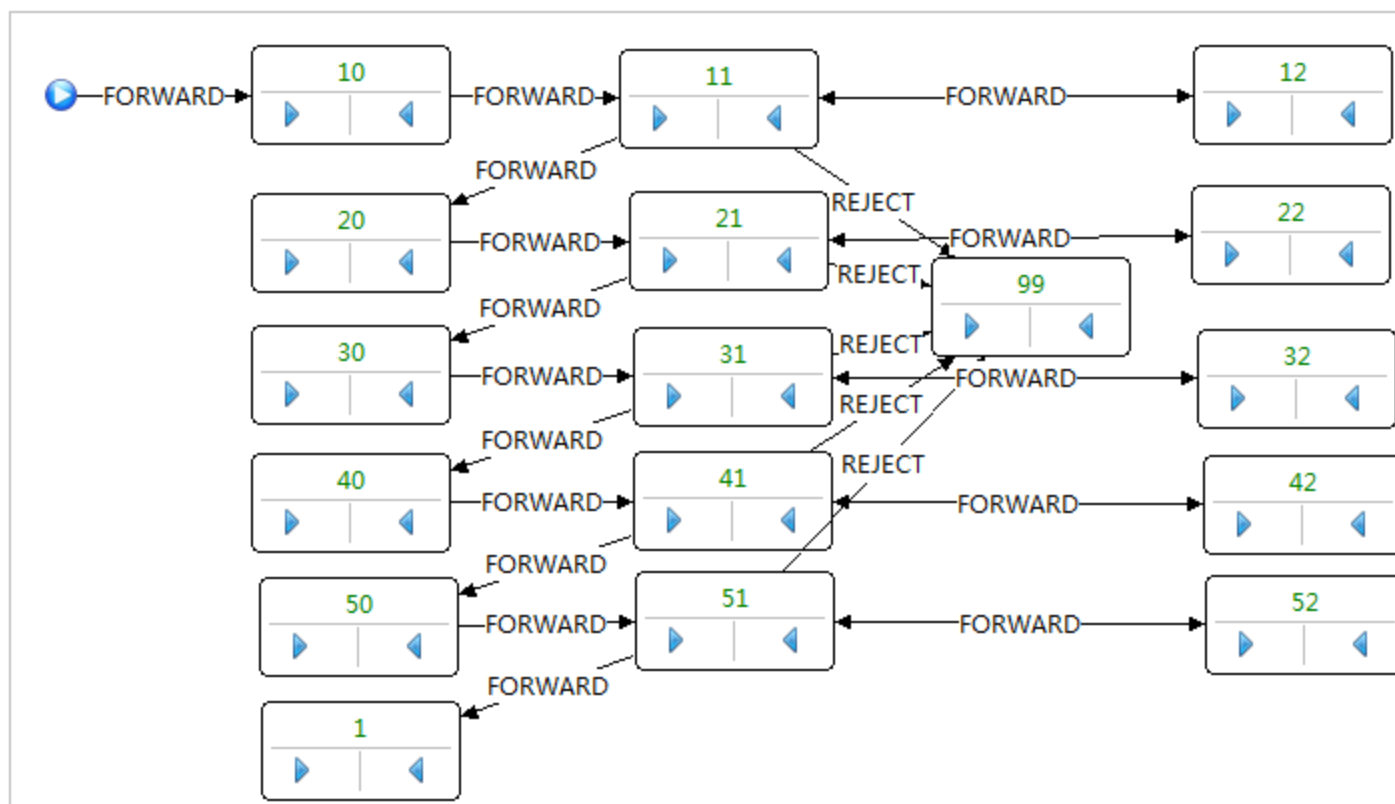
实际案例

user status



实际案例

researchJoinStatus



XEDA

The Next

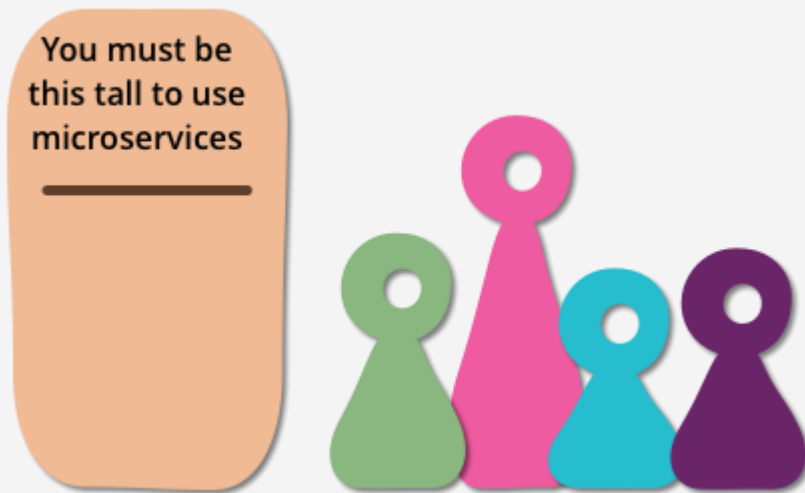
▲ XEDA

SEDA/Microservice implementation

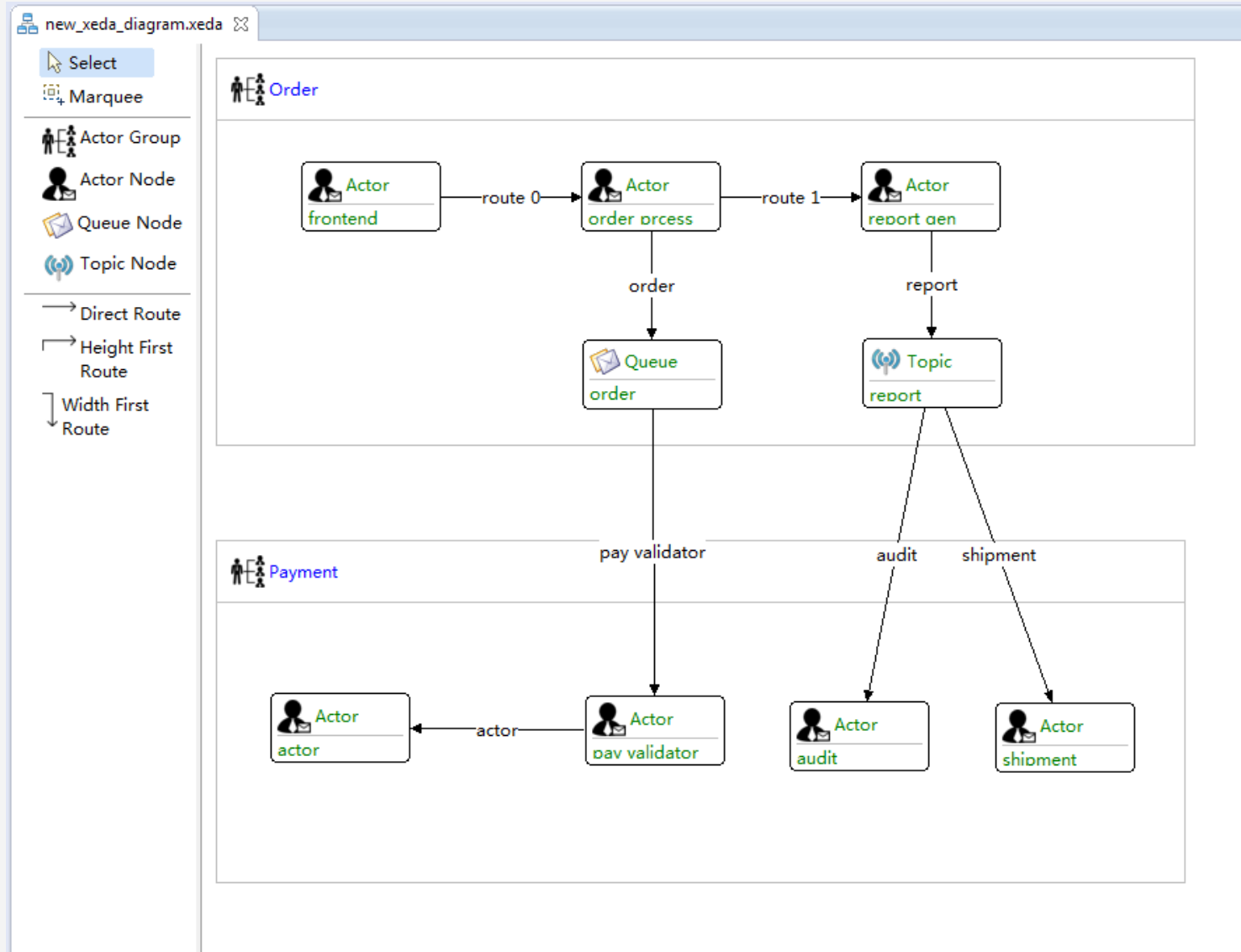
Service transition

Managing/Monitoring

Distributed OS



XEDA Preview



X series 资源

X series 资源

▲ Codebase

<https://github.com/hejiehui/xUnit>

<https://github.com/hejiehui/xDecision>

<https://github.com/hejiehui/xState>

▲ Sample

Normal project sample

https://github.com/hejiehui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/xunit_test.zip

Maven project sample

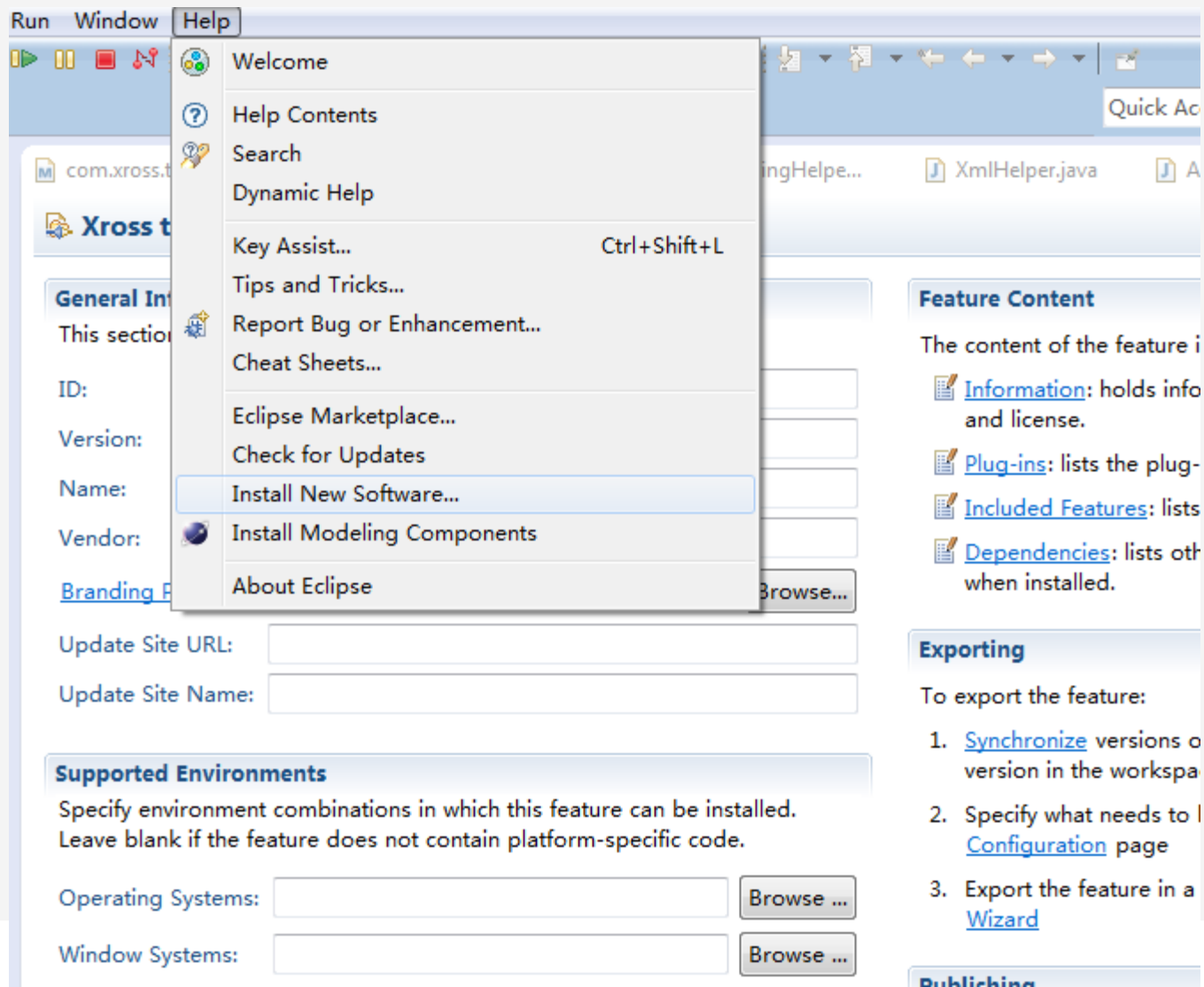
<https://github.com/hejiehui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/x-series-sample.zip>

▲ All-in-one Installer

<https://github.com/hejiehui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/xrossTools.zip>

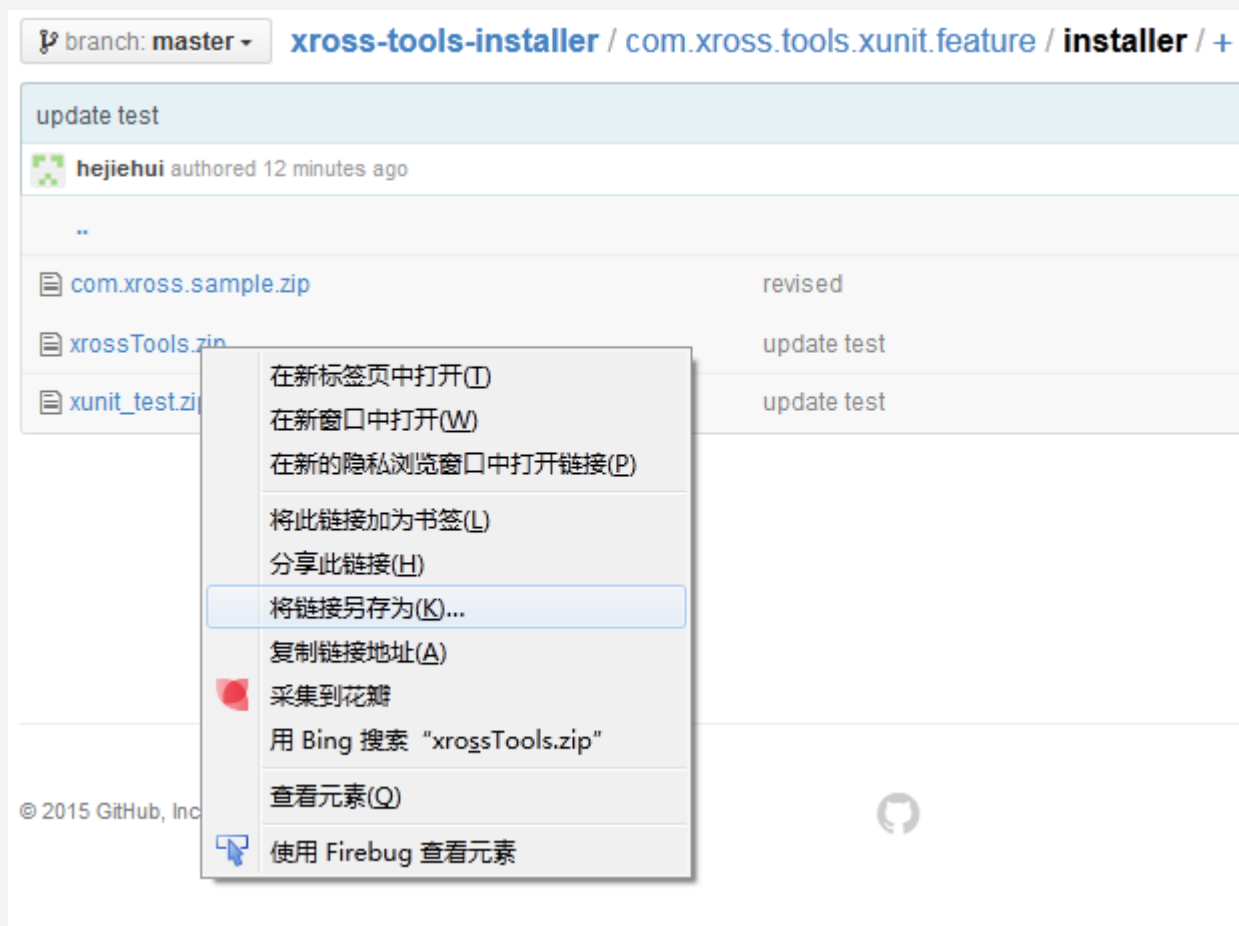
安装 X series

▲ 下载安装



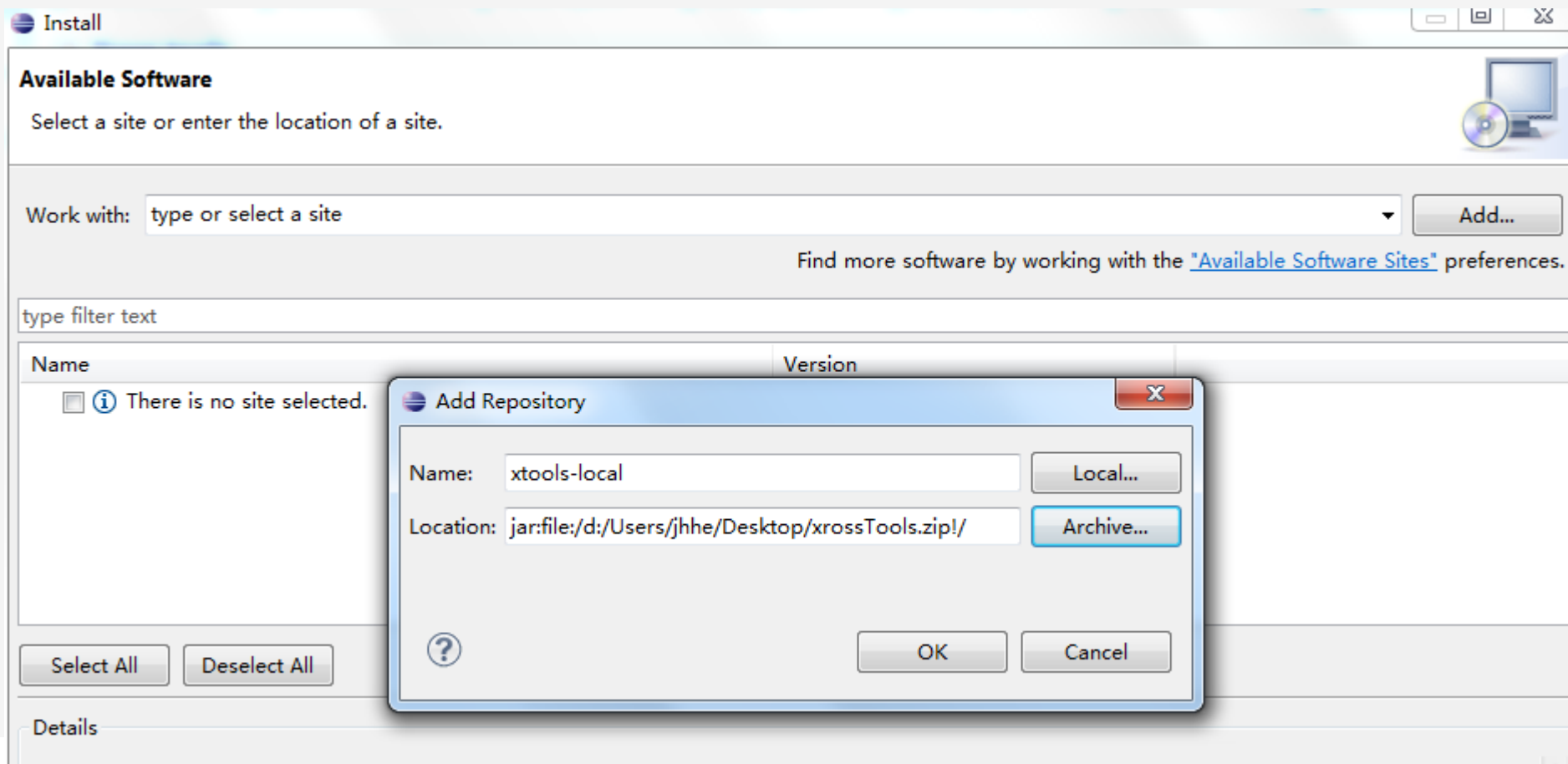
安装 X series

▲ 下载



安装 X series

▲ 指定安装路径



安装 X series

- ▲ In case you have installed GEF, you can uncheck the following

Details

☐ Show only the latest versions of available software

☐ Group items by category

☐ Show only software applicable to target environment

☐ Contact all update sites during install to find required software

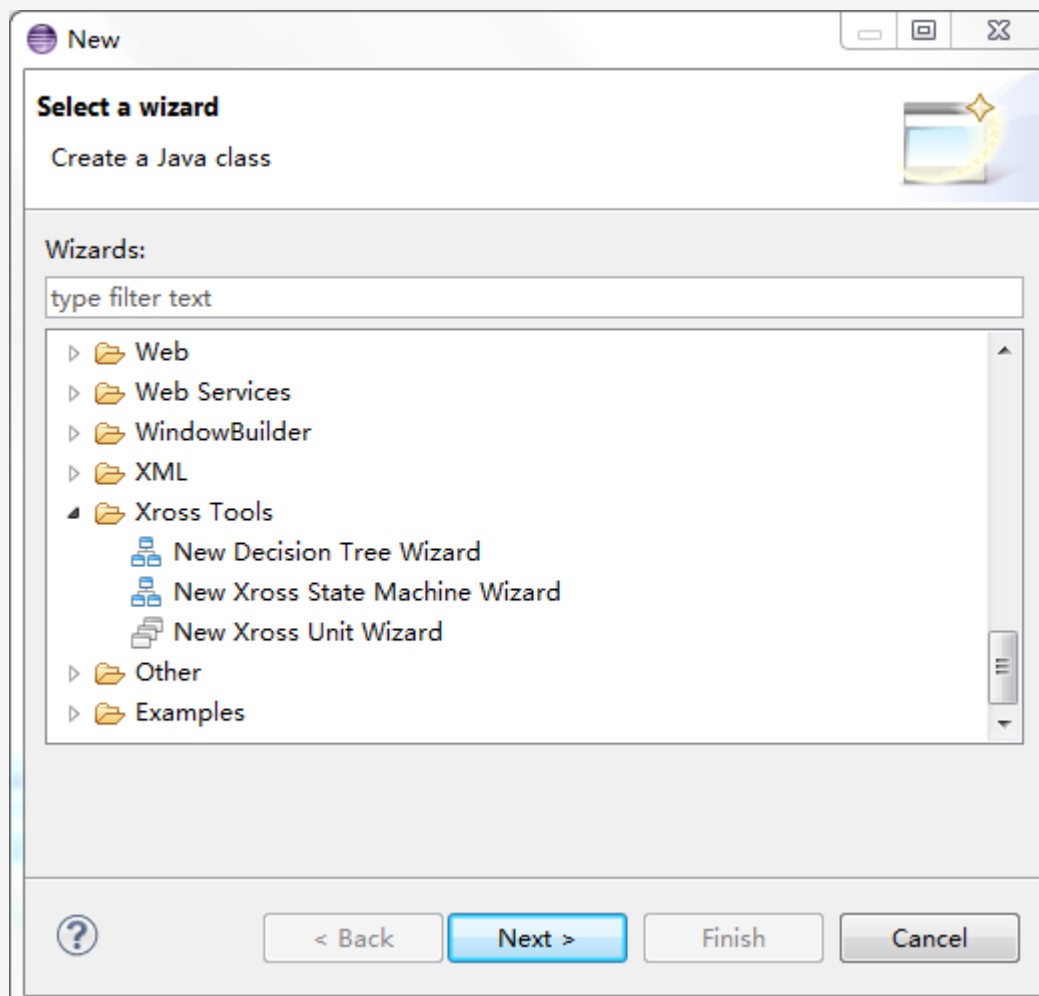
☒ Hide items that are already installed

What is [already installed?](#)

? < Back Next >

安装 X series

▲ New...Xross Tools



支持C#

- ▲ Xross unit C# runtime

https://github.com/hejiehui/xUnit/blob/master/doc/xunit_c%23.zip

- ▲ Xross State C# runtime

https://github.com/hejiehui/xState/blob/master/doc/xstate_c%23.zip

- ▲ Xross Decision C# runtime

https://github.com/hejiehui/xDecision/blob/master/doc/xdecision_c%23.zip

Before The End

- ▲ 在语言层面打转是没有出路的
- ▲ 大规模开发必须要求合适的工具
- ▲ 方向和眼光永远比速度重要

问题比答案更重要