



**X-Series**

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

# 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

# 挑战

## ▲ 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

### 缺乏合适工具之痛

- 大多数工具做的事和开发无关的管理工作
  - 编译，持续集成，源码管理，小作坊式的过程等等
- 管理工具越多，开发工作越艰难
  - 减慢开发节奏
- 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？
- 我们对此有没有清醒的认识？即使认识到了，我们可以做点什么？？
  - 面对残酷的人生，我们Shut up and enjoy?

# 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具体工作如何完成 [错误答案]

时序图的局限

- 仅能描述某一特定执行路径，对分支/循环无法直观表述

不幸的是这些图仍然需要进一步翻译

# 需求

## ▲ 开发并不仅仅意味着写代码

我们需要解决问题的正确途径

不幸的是我们解决所有问题都是用同一个原始的手段

## ▲ 我们需要人人都懂，无需翻译的媒介

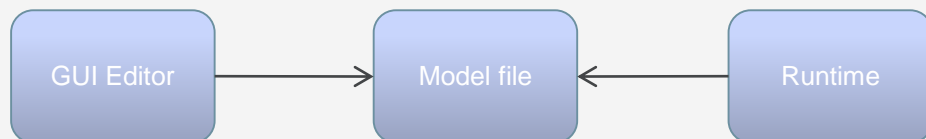
百闻不如一见 — 能可视化系统顶层模型

- 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

不是代码生成，而且在不同阶段模型都是同一个



# 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

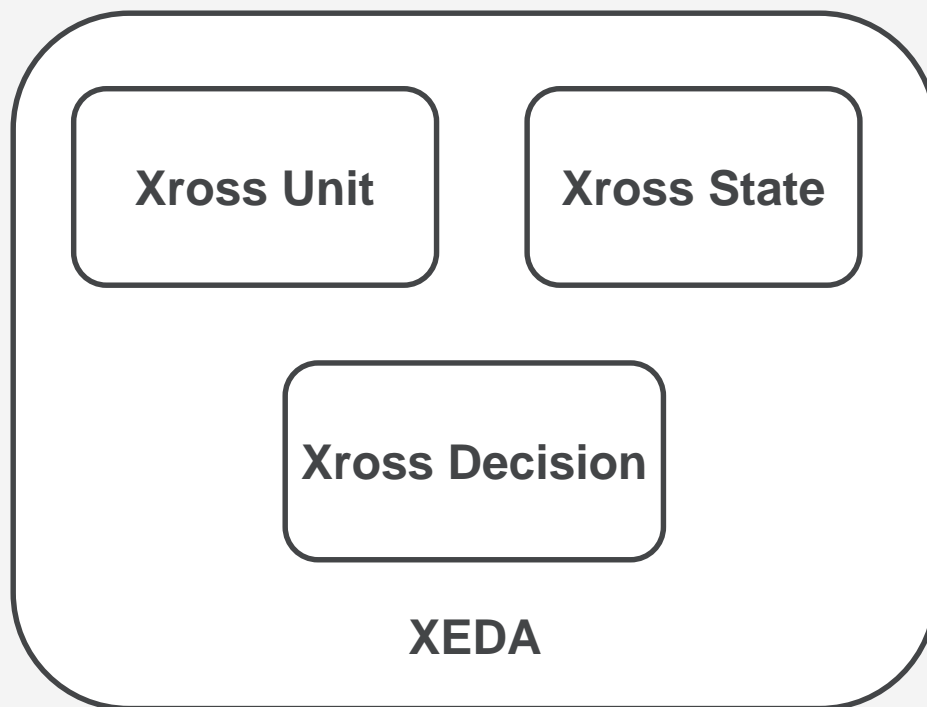
按照状态组织业务流程

领域级别

## ▲ Xeda

SEDA model

运行平台级别



# Xross Unit

## ▲ Xross unit 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

## ▲ 查看代码仅需双击进入!!!

参考缺省实现或者提供你自己定制的组件或结构



The screenshot shows the Eclipse IDE interface. The main editor window displays a flowchart diagram. The diagram starts with a component labeled 'a chain' which leads to a 'processor' box. This processor has three output paths: 'key1', 'not specified', and 'not specified'. The 'key1' path leads to a 'true' condition, which then branches into 'processor valid node' and 'processor invalid node'. The 'not specified' path leads to a 'processor update sum' box. The 'not specified' path also leads to a 'true' condition, which then branches into 'processor update sum' and 'processor update sum'. The 'processor update sum' boxes lead to a 'true' condition, which then leads to 'an adapter' box. The 'an adapter' box contains a 'processor unit 2' component. The bottom status bar shows the current file is 'Resource - com.xross...'.

# 关于 Xross Unit更多信息 1

## ▲ 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/相应的路径/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 2

## ▲ 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

因为他们全部缺乏他们最小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

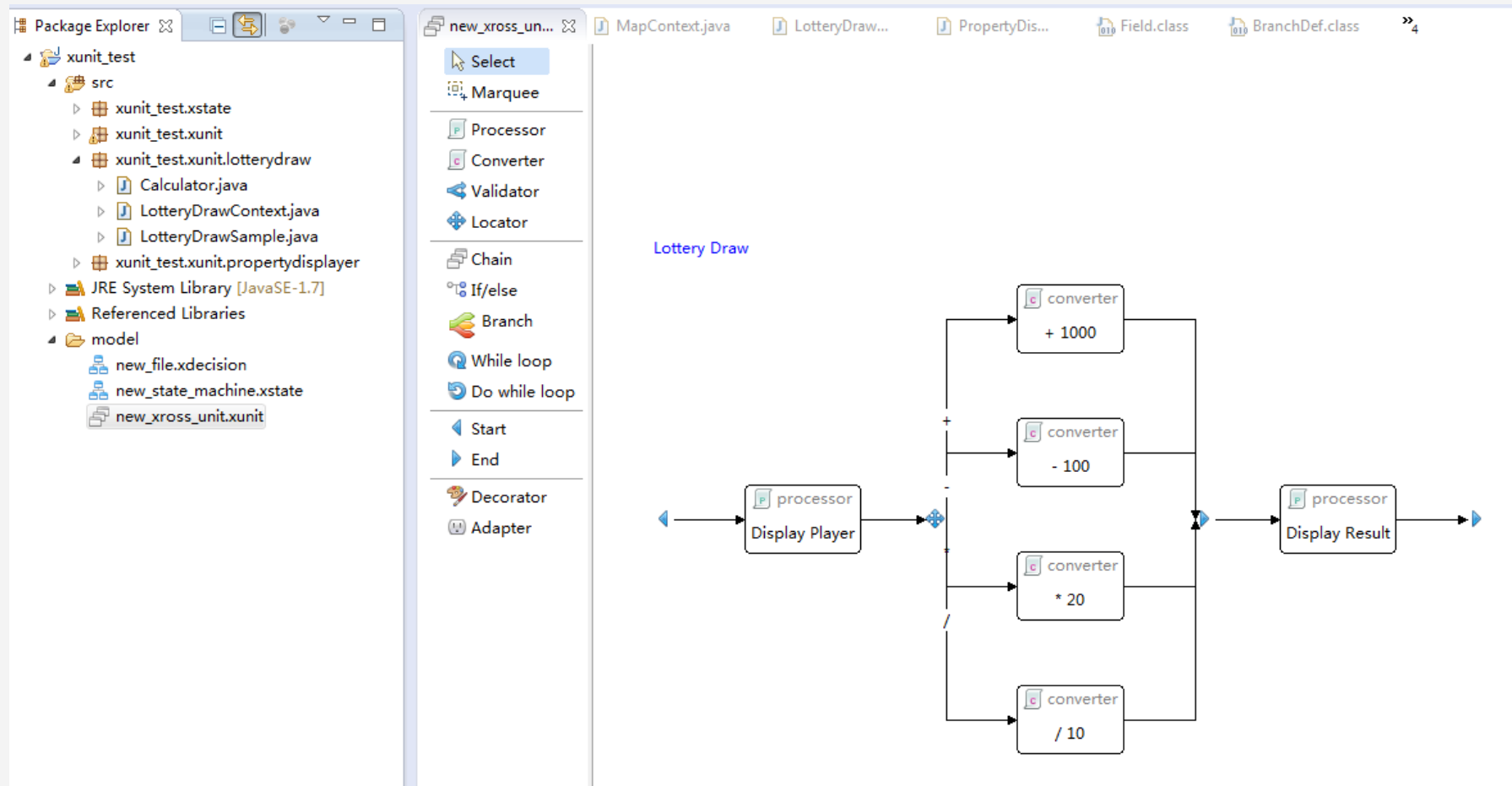
尽管有大量的小的仅仅只有一页代码的command

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]

# Xunit 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



# Xunit 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");
    }
}
```

# Xunit 示例

## 结合代码和系统蓝图，配置参数

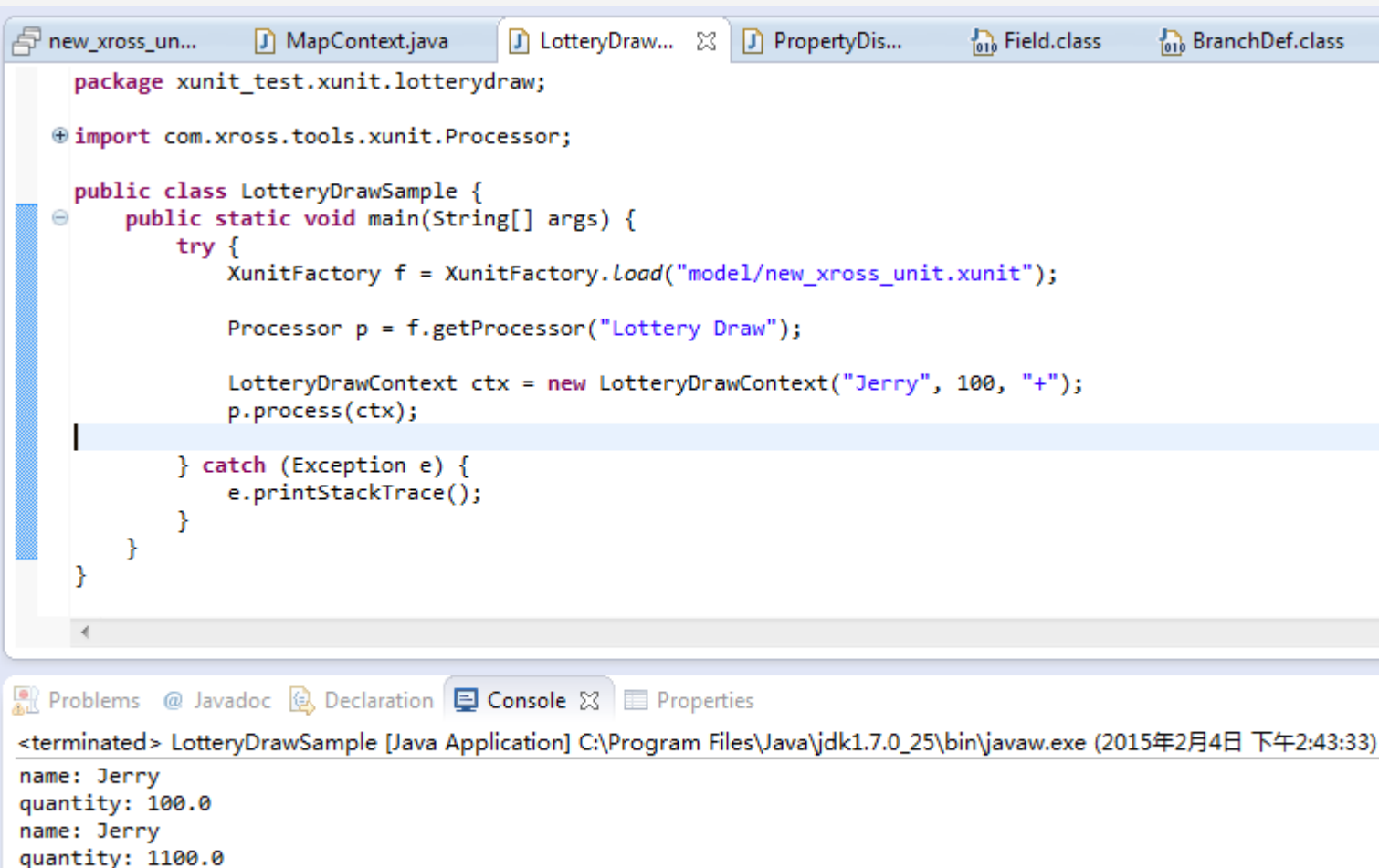
The screenshot displays the Xunit IDE interface. On the left, a sidebar contains a list of components: Chain, If/else, Branch, While loop, Do while loop, Start, End, Decorator, and Adapter. The main workspace, titled "Lottery Draw", shows a system blueprint with two "converter" blocks. The top block is labeled "+ 1000" and the bottom block is labeled "- 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 button labeled "OK".

Property Value

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

# Xunit 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);

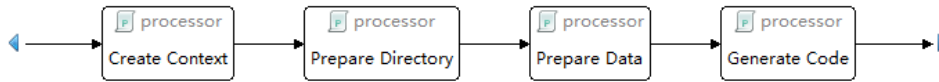
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Below the code editor, the `Console` tab is selected, displaying the output of the application:

```
<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
```

# Xunit 已经经过实际检验

## C# Code Generator



## Prepare C# Data Steps



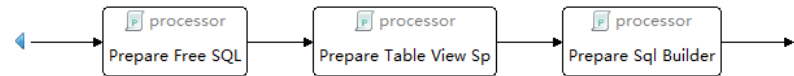
## Generate C# Code Steps



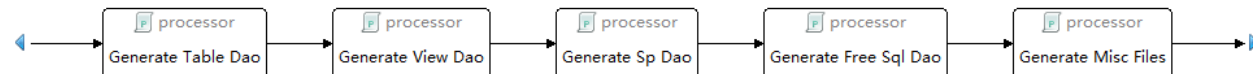
## Java Code Generator



## Prepare Java Data Steps



## Generate Java Code Steps





# 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

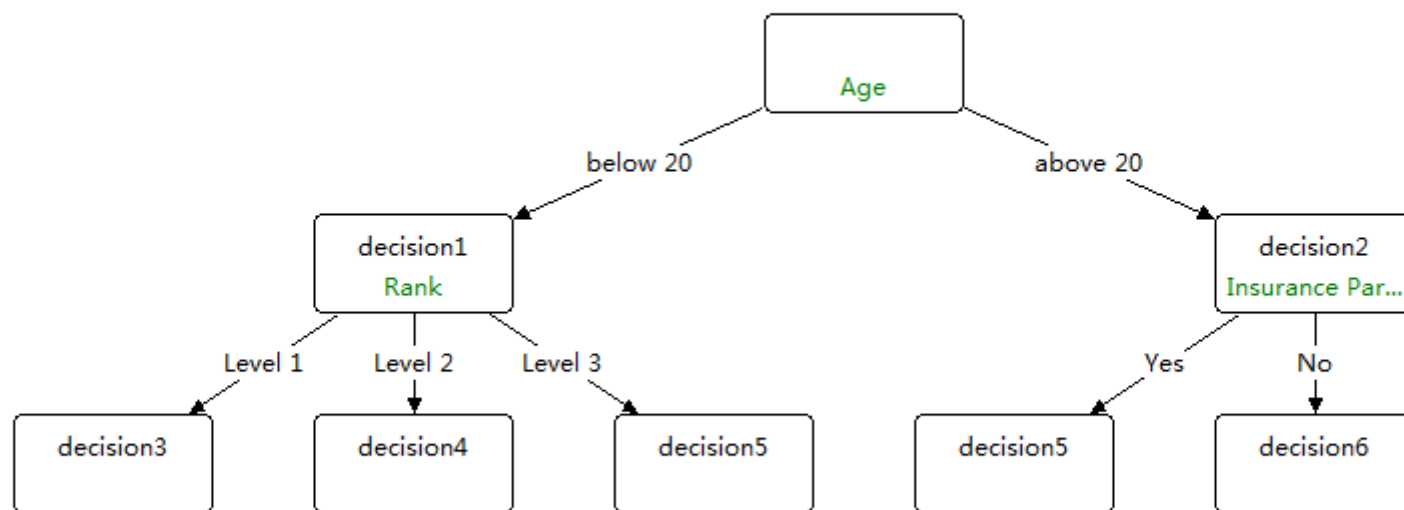
纯模型，无代码

方便重用

替代if/else，极大的简化代码

# Decision Tree 示例

## ▲ 极易学习



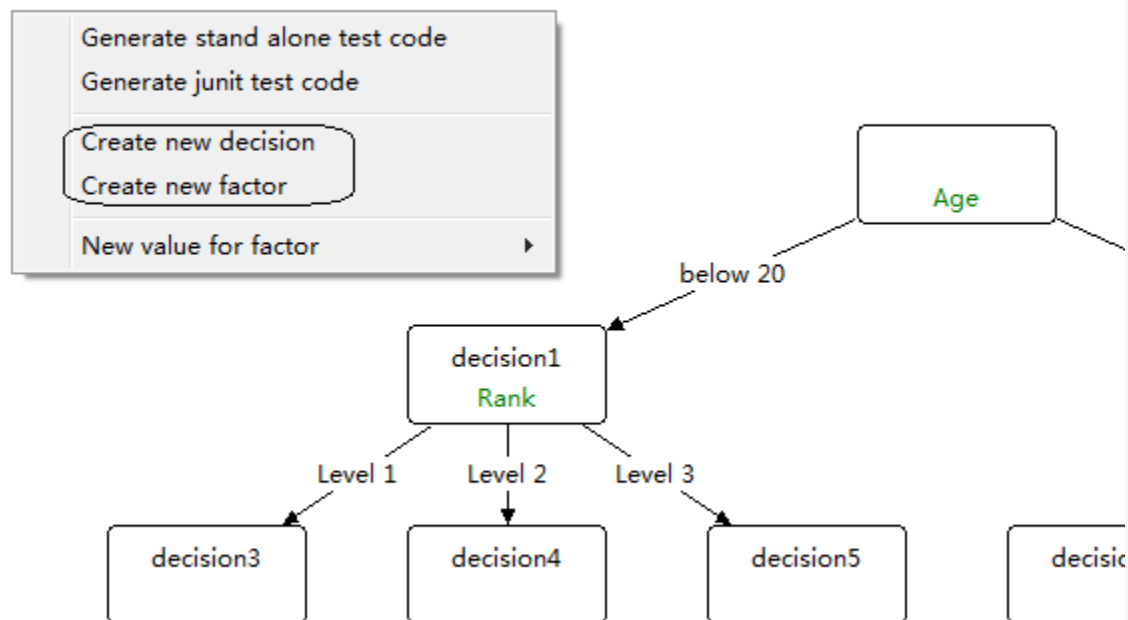
# Decision Tree 示例

## ▲ 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

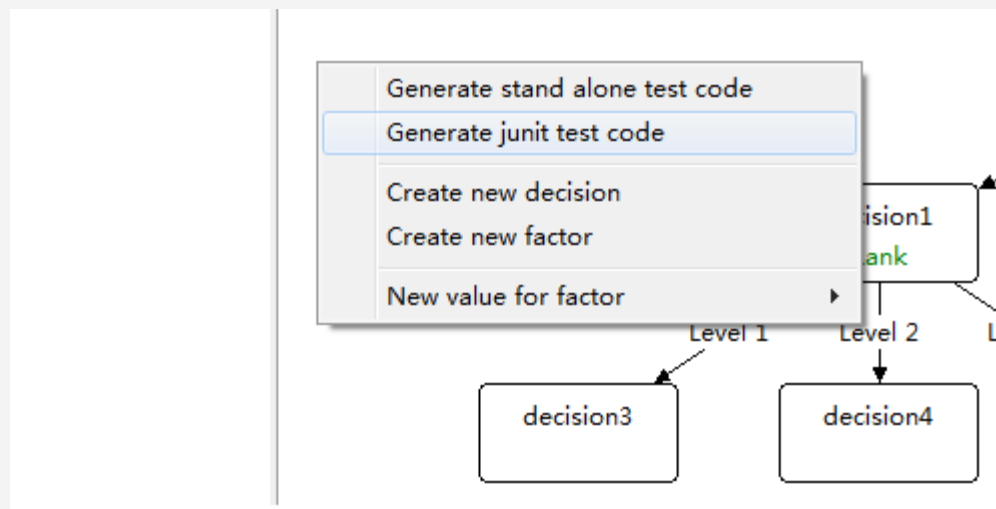


# Decision Tree 示例

## ▲ 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 decision tree editor. A context menu is open over a decision node labeled 'decision1' which has a green 'rank' factor. The menu options are: 'Generate stand alone test code', 'Generate junit test code' (highlighted), 'Create new decision', 'Create new factor', and 'New value for factor'. The decision tree diagram shows 'decision1' at the top, with arrows pointing to 'Level 1' and 'Level 2'. 'Level 1' points to 'decision3' and 'Level 2' points to 'decision4'.

```
Decision tree editor[Xross tools]
package com.ebay.xdomain.xcomponent;

import org.junit.Test;

import static org.junit.Assert.*;

import com.xross.tools.xdecision.MapFactors;
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("SimpleDeci
        } catch (Exception e) {
            e.printStackTrace();
            fail();
        }
    }
}
```

# Decision Tree 示例

Finished after 0.061 seconds

Runs: 1/1   Errors: 0   Failures: 0

▶ xunit\_test.xdecision.SimpleDecisonTest [Run]

Failure Trace

```
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");
```

# Xross State

- ▲ 一个允许开发人员创建状态机的编辑器
  - 用处广泛
  - 没有通用的解决方案
- ▲ 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

# Xross State基本元素

Select

Marquee

State Machine

State Node

Start Node

End Node

Transition

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

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

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

Maven project sample

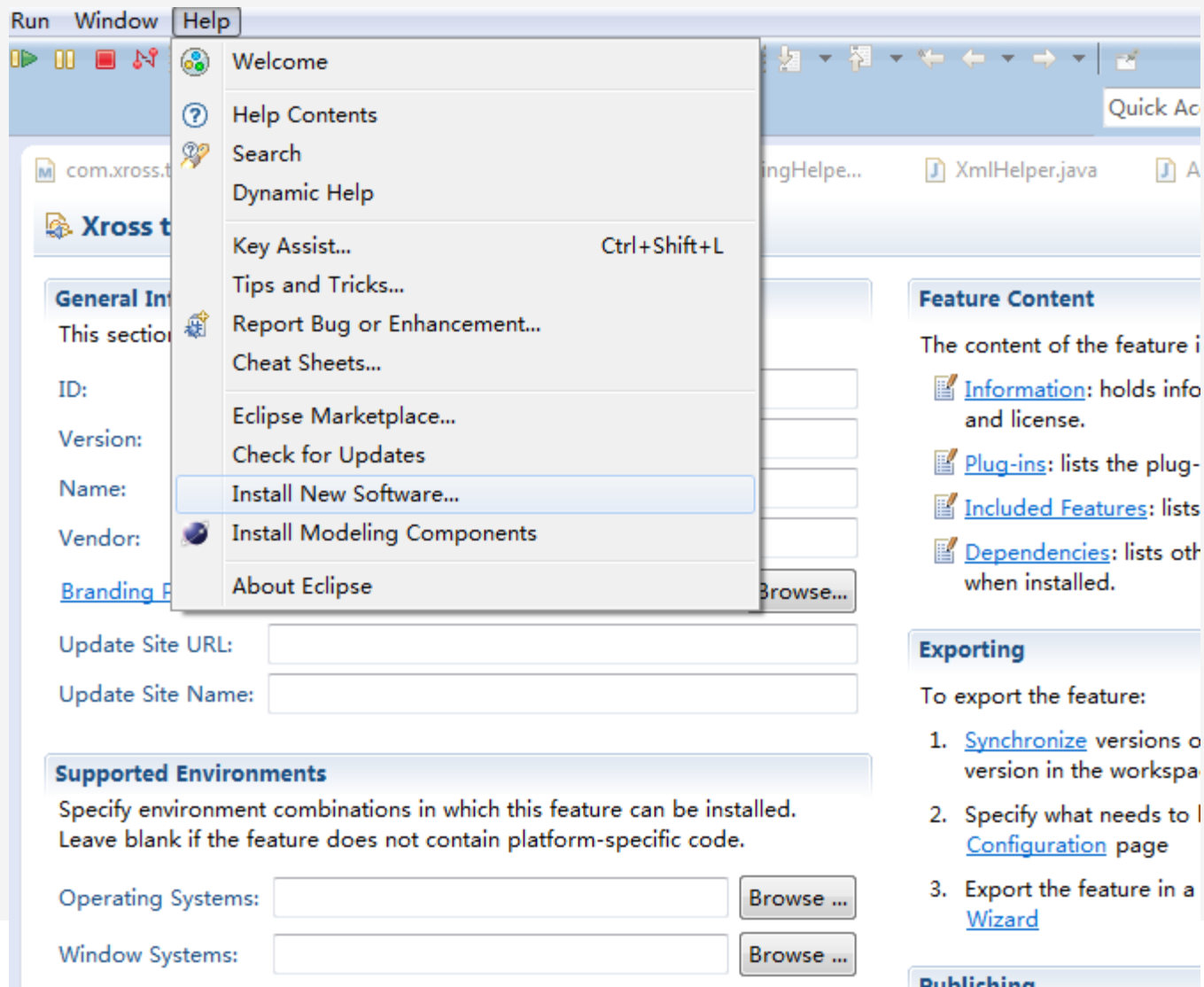
<https://github.com/hejiehui/xross-tools-installer/blob/master/com.xross.tools.xunit.feature/installer/x-series-sample.zip>

## ▲ All-in-one Installer

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

# 安装 X series

## ▲ 下载安装



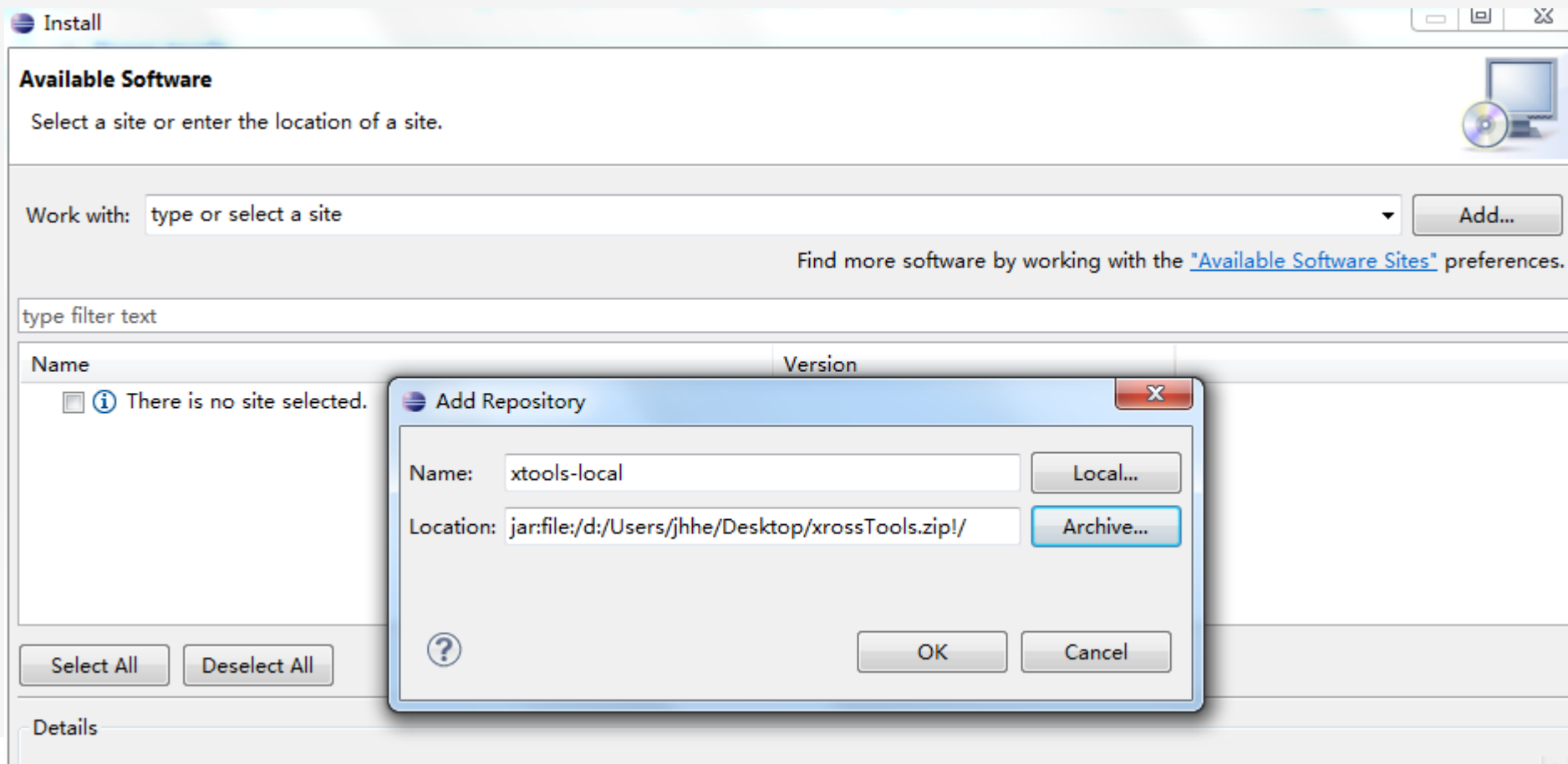
# 安装 X series

## ▲ 下载



# 安装 X series

## ▲ 指定安装路径



# Install 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

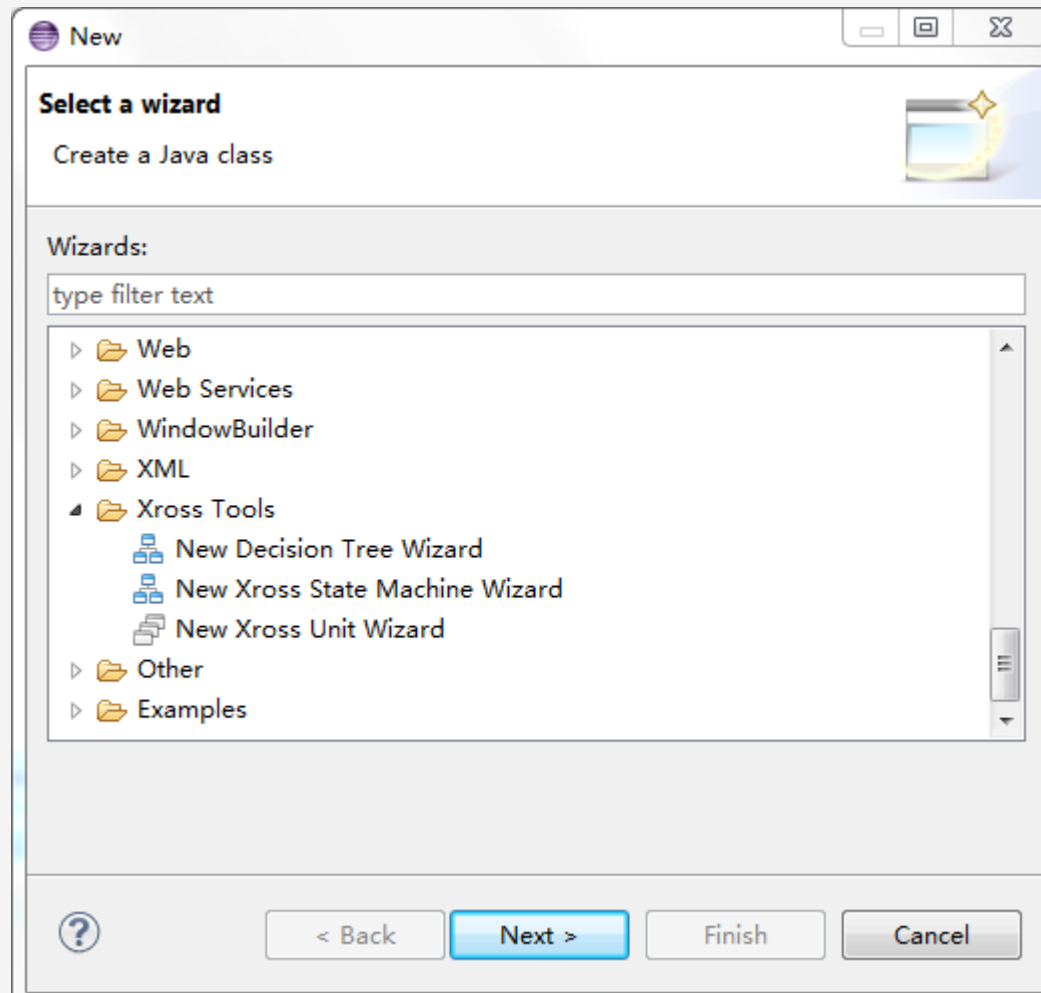
☒ Hide items that are already installed

What is [already installed?](#)

? < Back Next >

# Install X Series

## ▲ New...Xross Tools



# The Next

## ▲ XEDA

SEDA/Microservice implementation

Service transition

Managing/Monitoring

You must be  
this tall to use  
microservices

---





问题比答案更重要