Using Aspect-Oriented Programming to Streamline Mobile Application Development

Allen Lau

CTO and Co-Founder
Tira Wireless
tirawireless.com

TS-5363

Goal of This Talk

Learn how to use Aspect-Oriented Programming (AOP) to simplify and speed up the development of mobile application.

Agenda

Development challenges

AOP primer

How AOP applies to mobile development

Demo

Agenda

Development challenges

AOP primer

How AOP applies to mobile development

Demo

Current Landscape

- Hundreds of mobile operators have deployed Java™ platform programs—Many have custom requirements

- Thousands of Java platform handset models exist new models being introduced every month

NOKIA
MOTOROLA
SIEMENS
SAMSUNG
SONY
SHARP
NEC
Panasonic
mitsubishi
FUJITSU
HITACHI
TOSHIBA

The Challenges

Why deploying on tiny devices can be so complex

- Devices are very different
  - Screen size, heap memory, key mapping, VM implementations difference
- Operators have different requirements
- Projects have shorter lifespan
- Lots of similar but different builds
- Lack of component model
  - OOP is not sufficient
  - Hard to reuse code from other device builds or other projects
  - Knowledge is in people's heads

Agenda

Development challenges

AOP primer

How AOP applies to mobile development

Demo

Aspect-Oriented Programming

What is AOP?

- AOP complements OO programming
- Dynamically modify static OO models
- Facilitates modularization of cross-cutting concerns
    - In simple terms, it means having a single module that can affect the behaviour of one or more classes
    - Centralized changes instead of scattering across existing model

Logging Example

```
void paint(Graphics g)
{
    // your paint code
}

void keyPressed(int keyCode)
{
    // your key processing code
}
```

Logging Example (Cont.)

Now adds ‘logging’ code in an old fashioned way

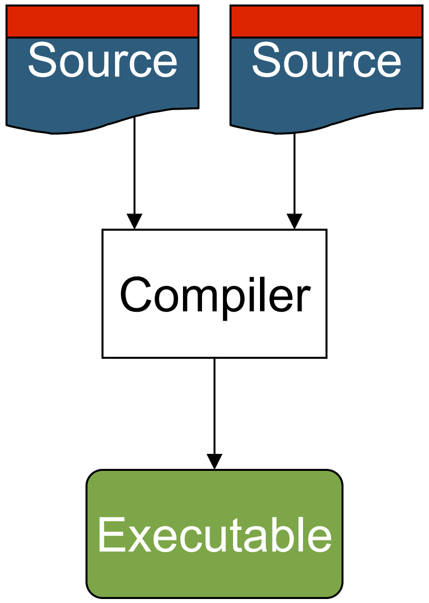
```
void paint(Graphics g)
{
   logging(“entering paint”);
   // your paint code
   logging(“leaving paint”);
}

void keyPressed(int keyCode)
{
   logging(“entering keyPressed”);
   // your key processing code
   logging(“entering keyPressed”);
}
```

Logging Example (Cont.)

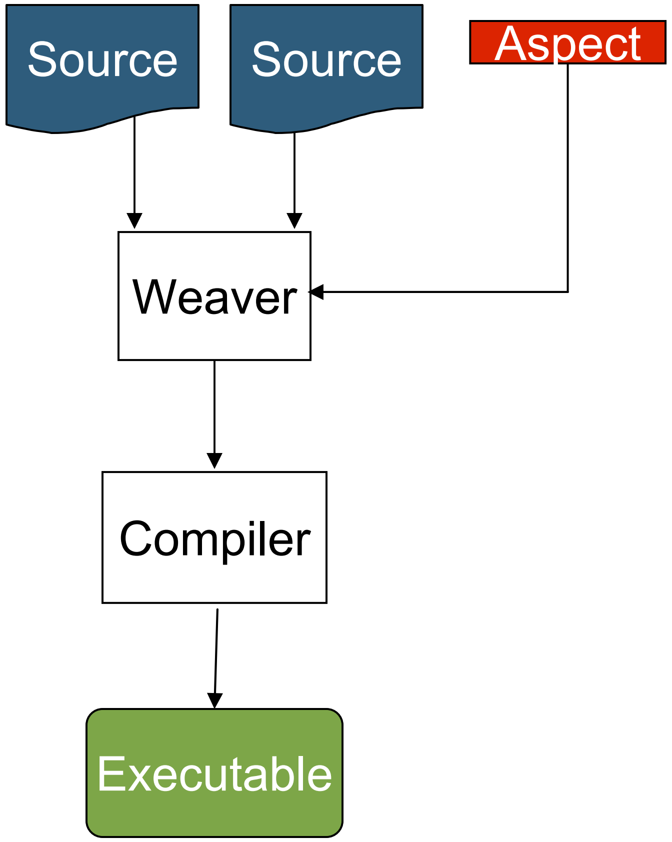
Pseudo code of an ‘aspect’

```
loggingAspect
{
   loggableCalls = paint, keyPressed;

   before: loggableCalls
   {
      logging(“entering “ + $methodName);
   }

   after: loggableCalls
   {
      logging(“leaving “ + $methodName);
   }
}
```

Without AOP



With AOP



Glossary

- Pointcut
- Advice
- Aspect

Pointcut

The point of execution in the application at which cross-cutting concern needs to be applied

```
loggingAspect
{
   loggableCalls = paint, keyPressed;

   before: loggableCalls
   {
      logging(“entering “ + $methodName);
   }

   after: loggableCalls
   {
      logging(“leaving “ + $methodName);
   }
}
```

Advice

The code that you want to apply to your existing model

```
loggingAspect
{
   loggableCalls = paint, keyPressed;

   before: loggableCalls
   {
      logging(“entering “ + $methodName);
   }

   after: loggableCalls
   {
      logging(“leaving “ + $methodName);
   }
}
```

Aspect

The combination of pointcut(s) and the advice(s)

```
loggingAspect
{
   loggableCalls = paint, keyPressed;

   before: loggableCalls
   {
      logging("entering " + $methodName);
   }

   after: loggableCalls
   {
      logging("leaving " + $methodName);
   }
}
```

Agenda

Development challenges

AOP primer

How AOP applies to mobile development

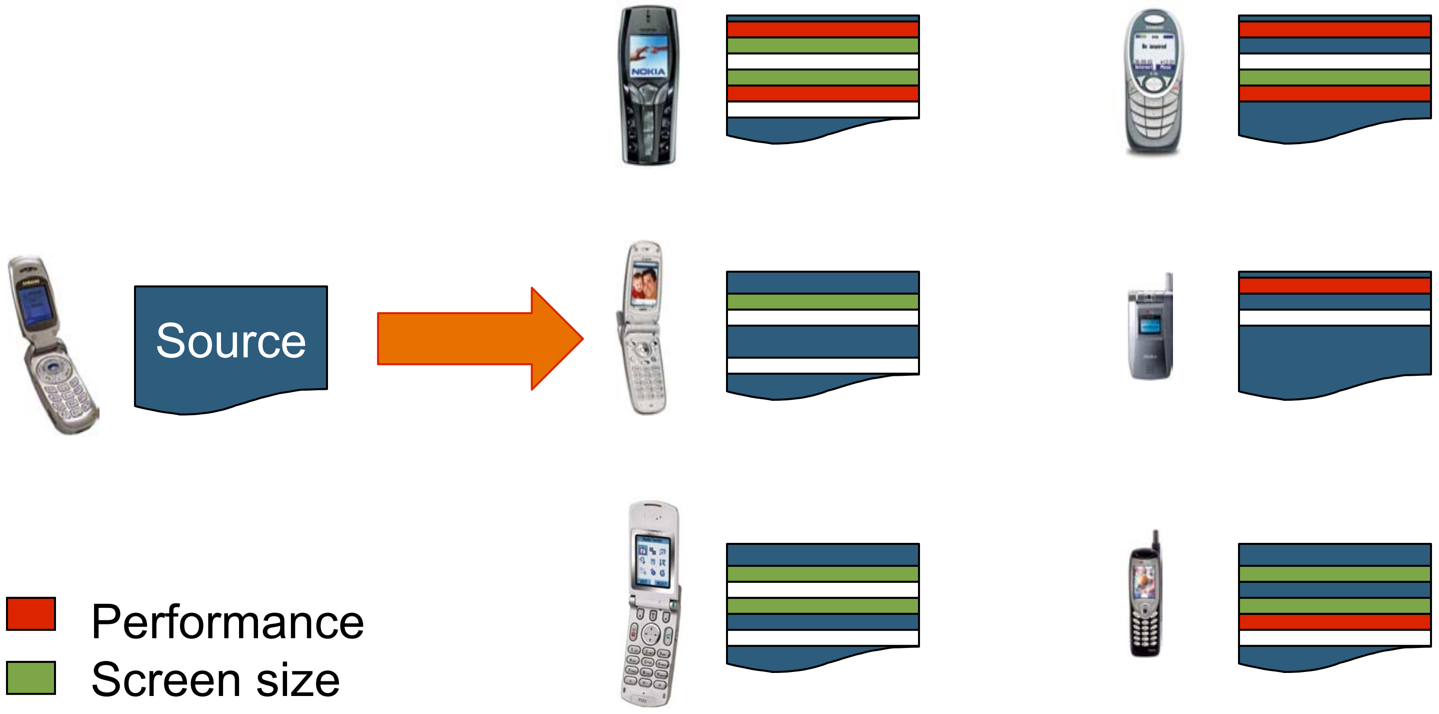
Demo

How AOP Applies to Mobile

- Modularizing cross-cutting concerns
- Encourage code reuse
- Enable knowledge discovery
- Survey indicates developers are wasting 25%–50% of their valuable time due to:
    - Inability to reuse code previous developed
    - Inability to realize the existence of reusable code

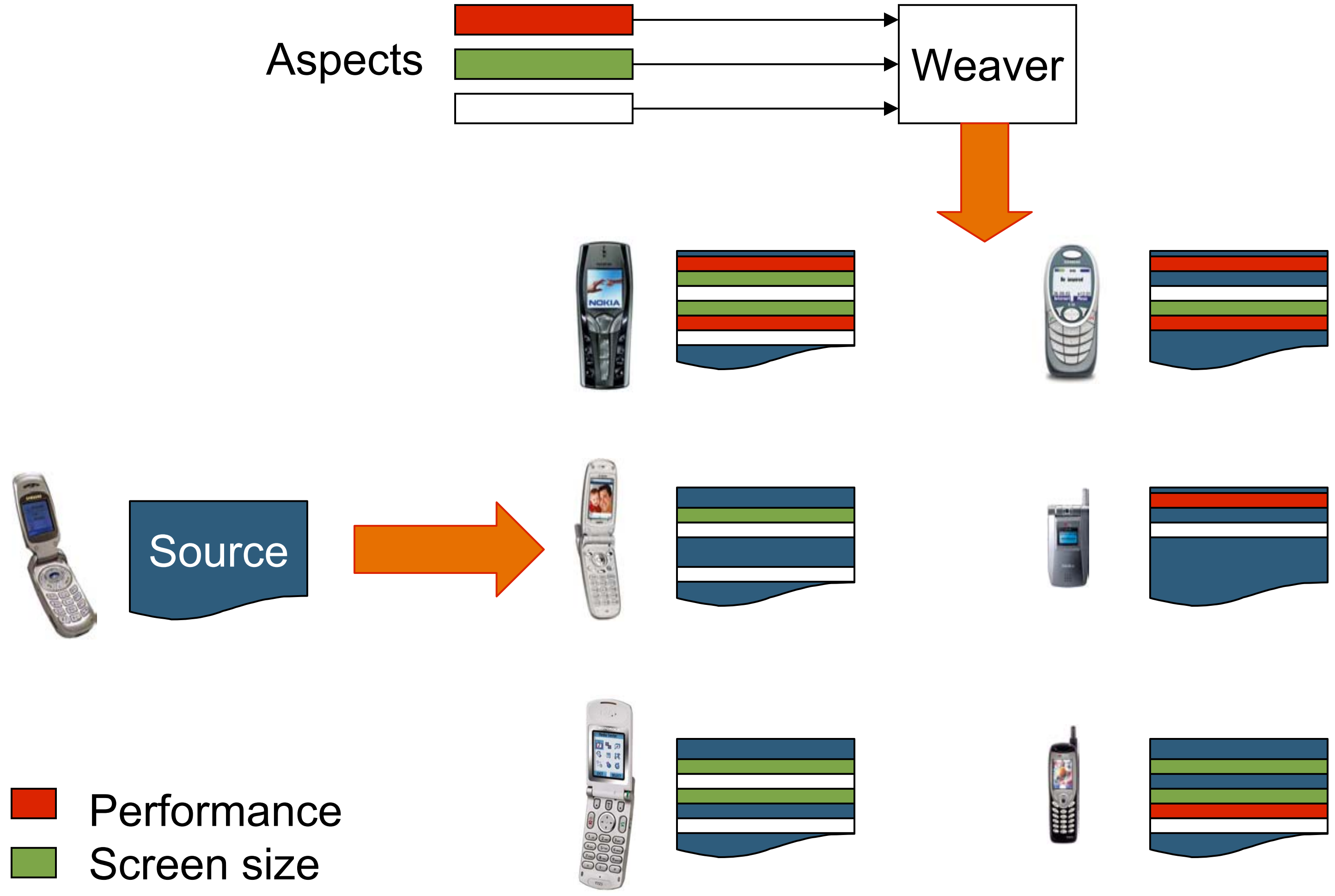
Source: Tira Wireless

Without Modularising Concerns Into Aspects



🟥 Performance
🟩 Screen size
□ Memory

Modularising and Reusing Concerns Into Aspects



🟥 Performance
🟩 Screen size
⬜ Memory

Knowledge Discovery

Reusable code is useless if nobody knows its existence

- Overtime a lot of aspects are developed
- How can developers find the right aspects to reuse?
- How can developers leverage the broader community?
- Aspects are “just code”, more metadata is needed

Jumplet

Aspects with metadata

- A collection of aspects that addresses a particular issue
  - An issue example: Sprint builds require GameLobby
  - A Sprint GameLobby Jumplet can contain the following aspects:
    - Bootstrap
    - High score screen
- Metadata
  - Tags
  - Device properties
    - e.g., List of Sprint devices
  - Usage count

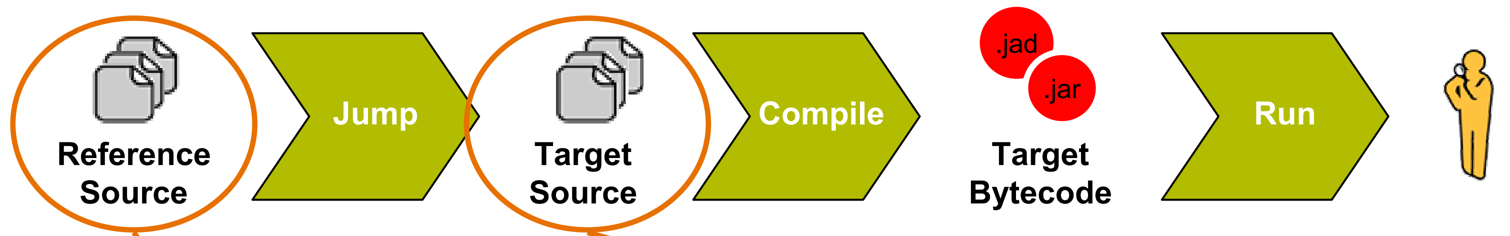
Agenda

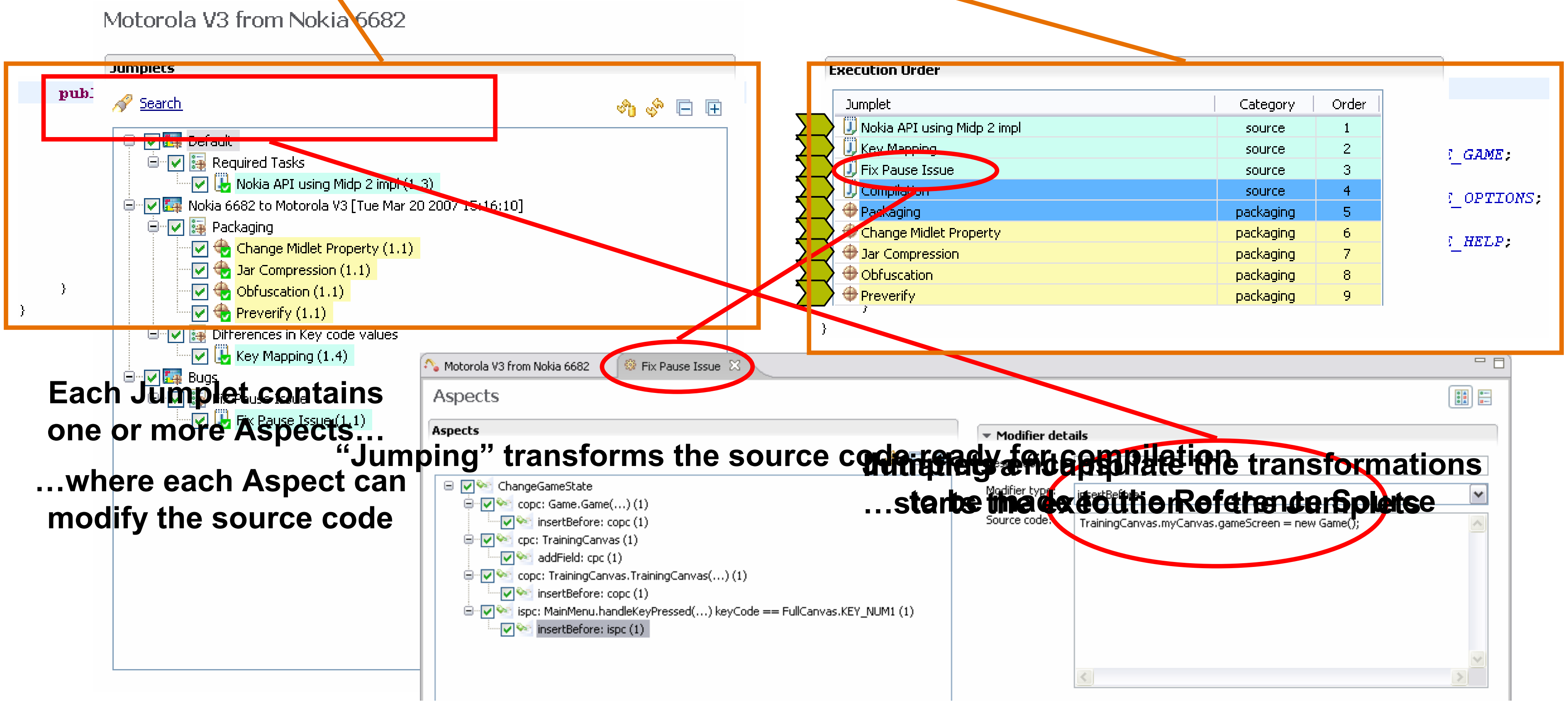
Development challenges

AOP primer

How AOP applies to mobile development

Demo




Motorola V3 from Nokia 6682

Jumplets

Search

- Default
  - Required Tasks
    - Nokia API using Midp 2 impl (1.3)
- Nokia 6682 to Motorola V3 [Tue Mar 20 2007 15:16:10]
  - Packaging
    - Change Midlet Property (1.1)
    - Jar Compression (1.1)
    - Obfuscation (1.1)
    - Preverify (1.1)
  - Differences in Key code values
    - Key Mapping (1.4)
- Bugs
  - Fix Pause Issue
    - Fix Pause Issue (1.1)

Execution Order

Jumplet	Category	Order
Nokia API using Midp 2 impl	source	1
Key Mapping	source	2
Fix Pause Issue	source	3
Compilation	source	4
Packaging	packaging	5
Change Midlet Property	packaging	6
Jar Compression	packaging	7
Obfuscation	packaging	8
Preverify	packaging	9

Motorola V3 from Nokia 6682 | Fix Pause Issue

Aspects

Aspects

- ChangeGameState
  - copc: Game.Game(...) (1)
    - insertBefore: copc (1)
  - cpc: TrainingCanvas (1)
    - addField: cpc (1)
  - copc: TrainingCanvas.TrainingCanvas(...) (1)
    - insertBefore: copc (1)
  - ispc: MainMenu.handleKeyPressed(...) keyCode == FullCanvas.KEY_NUM1 (1)
    - insertBefore: ispc (1)

Modifier details

Modifier type: insertBefore

Source code:

```
TrainingCanvas.myCanvas.gameScreen = new Game();
```

Each Jumplet contains one or more Aspects…

…where each Aspect can modify the source code

“Jumping” transforms the source code ready for compilation

Jumplets encapsulate the transformations

…to be made to the Reference Source

Initiating a Jump…

…starts the execution of the Jumplets

Sprint Game Lobby

- Sprint’s gaming community
  - View Leaderboard
  - Rate the Game
  - Recommend the Game
  - My Stats

Sprint Game Lobby Implementation

- Bootstrap code
  - Insert Game Lobby class library
  - Subclass from GCMIDlet instead of MIDlet
  - Implement abstract methods (e.g., rxData)
- User interface
  - Menu
  - Score posting, rating, and recommendations (http calls)
  - Leaderboard, My Stats UI

DEMO

Implementing Sprint Game Lobby Using AOP

Summary

- OOP is not sufficient as the only component model for mobile development
- AOP provides the “missing link”
- Mobile Development 2.0—Leverage the community

For More Information

- www.tirawireless.com
- www.eclipse.org/aspectj
- wikipedia.org/wiki/aspect-oriented_programming

Q&A

Allen Lau

CTO and Co-Founder, Tira Wireless

Using Aspect-Oriented Programming to Streamline Mobile Application Development

Allen Lau

CTO and Co-Founder
Tira Wireless
tirawireless.com

TS-5363