TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Lehrstuhl Softwaretechnologie

Komplexpraktikum:

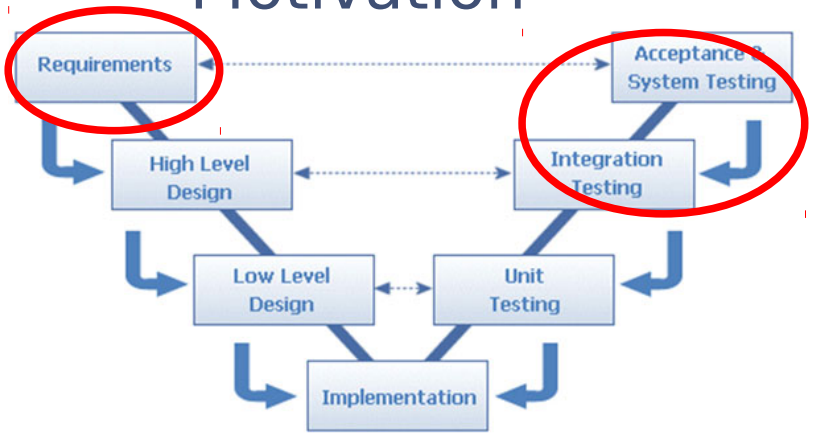
Quality Smells für Android-Applikationen

http://st.inf.tu-dresden.de/teaching/kp/

Dipl.-Inf. Jan Reimann

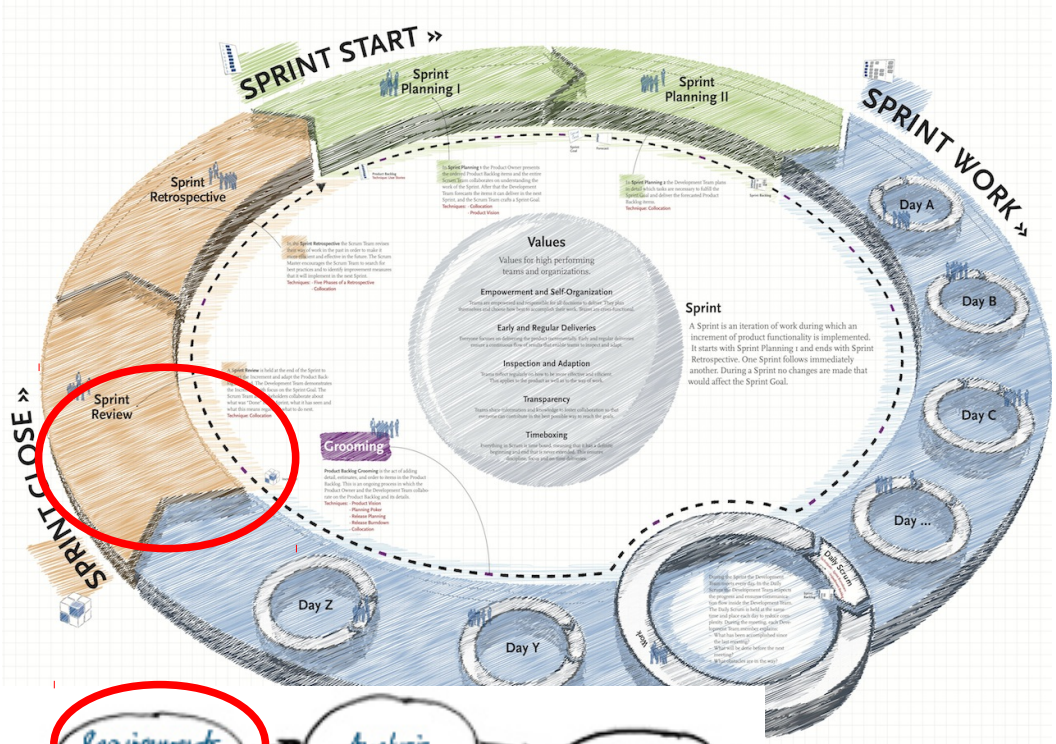
Software Technology Group

Motivation

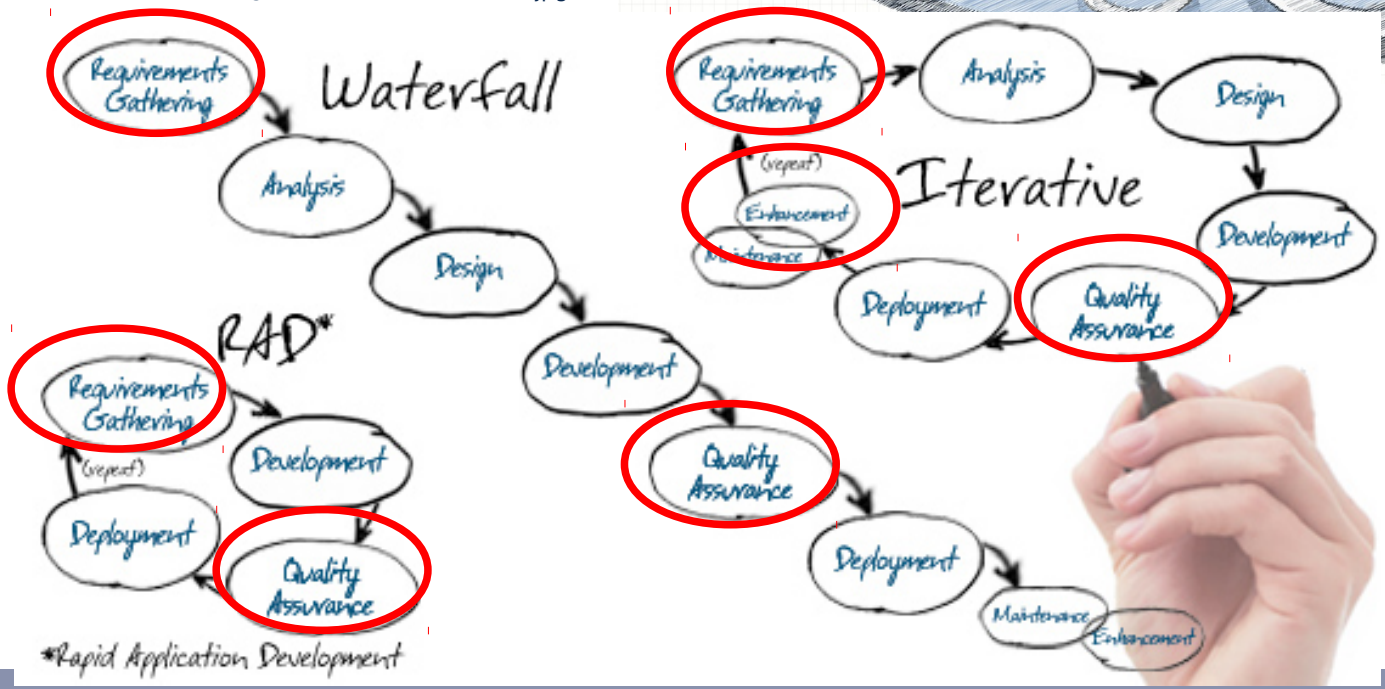


[http://rootsitservices.com/pictures/v-shaped.jpg]

[http://www.scrumbrowser.com/pkg/rc/SmallScrum/0/HEAD/file/f.img.scr.01.005.001/Scrum-Flow.jpg]



[http://cloverleafsolutions.com/img/methods/sdlc-variations.jpg]



Motivation

Developers must perform:

1) Select quality to focus

2) Identify bad implementations regarding selected quality
   → contain *bad smells*

   "...certain structures in the code that suggest (sometimes they scream for) [restructurings to improve design]" [Fowler, 1999]

3) Optimise bad implementations w.r.t. selected quality
   → apply *refactoring* [Fowler, 1999]

   "...is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure" [Fowler, 1999] [Opdyke+, 1990]

→ huge manual effort

Motivation

- ***Quality Smell*** [Reimann+, 2013a]:
  Is a bad smell with regard to a specific quality, expressing that this bad smell negatively influences the given quality of a model, and which can be resolved by specific refactorings. → enables focus on specific qualities in early phases
- **Problem:**
  - Developers are aware of quality smells only indirectly
  - Definitions are informal (best-practices, bug trackers, forum discussions)
  - Resources where to find them are distributed over the web

→ Catalogue containing precisely defined quality smells is needed

Quality Smell Schema

Concept	Description
Name	Unique and descriptive identifier
Context	Categoric relation (e.g. UI, sensors, etc.)
Affected Qualities	Qualities negatively influenced by this quality smell
Description	Detailed description of the specific problem including an example
Refactorings	Explains refactorings being able to resolve this quality smell
References	Further (web) resources regarding this quality smell
Related Quality Smells	Similar or related quality smells

Quality Smell Catalogue For Android – An Excerpt

Quality	Quality Smell	Refactoring
Energy	**Durable WakeLock**	Aquire WakeLock with timeout
	Rigid AlarmManager	Use inexact AlarmManager, Use AlarmManager without wakeup
	Data Transmission Without Compression	Add Data Compression to Apache HTTP Client based file transmission
	Early Resource Binding	Move Resource Request to Visible State Method
Memory	**No Low Memory Resolver**	Override onLowMemory()
	Leaking Inner Class	Introduce Static Class, Introduce Weak Reference
User Experience	**Interruption From Background**	Introduce Notification
	Dropped Data	Save instance state
Security	**Tracking Hardware ID**	Use unique generated ID
	Public Data	Set Private Mode

http://modelrefactoring.org/smell_catalogue

Example: Interruption From Background

- **Context:** UI
- **Affected Qualities:** User Experience, User Conformance
- **Description:** For the user it's not considered to be expected behaviour if a background function (`BroadcastReciever` or `Service`) interrupts the current activity. Worst case might be if input gets lost.
- **Refactorings:** Introduce Notification

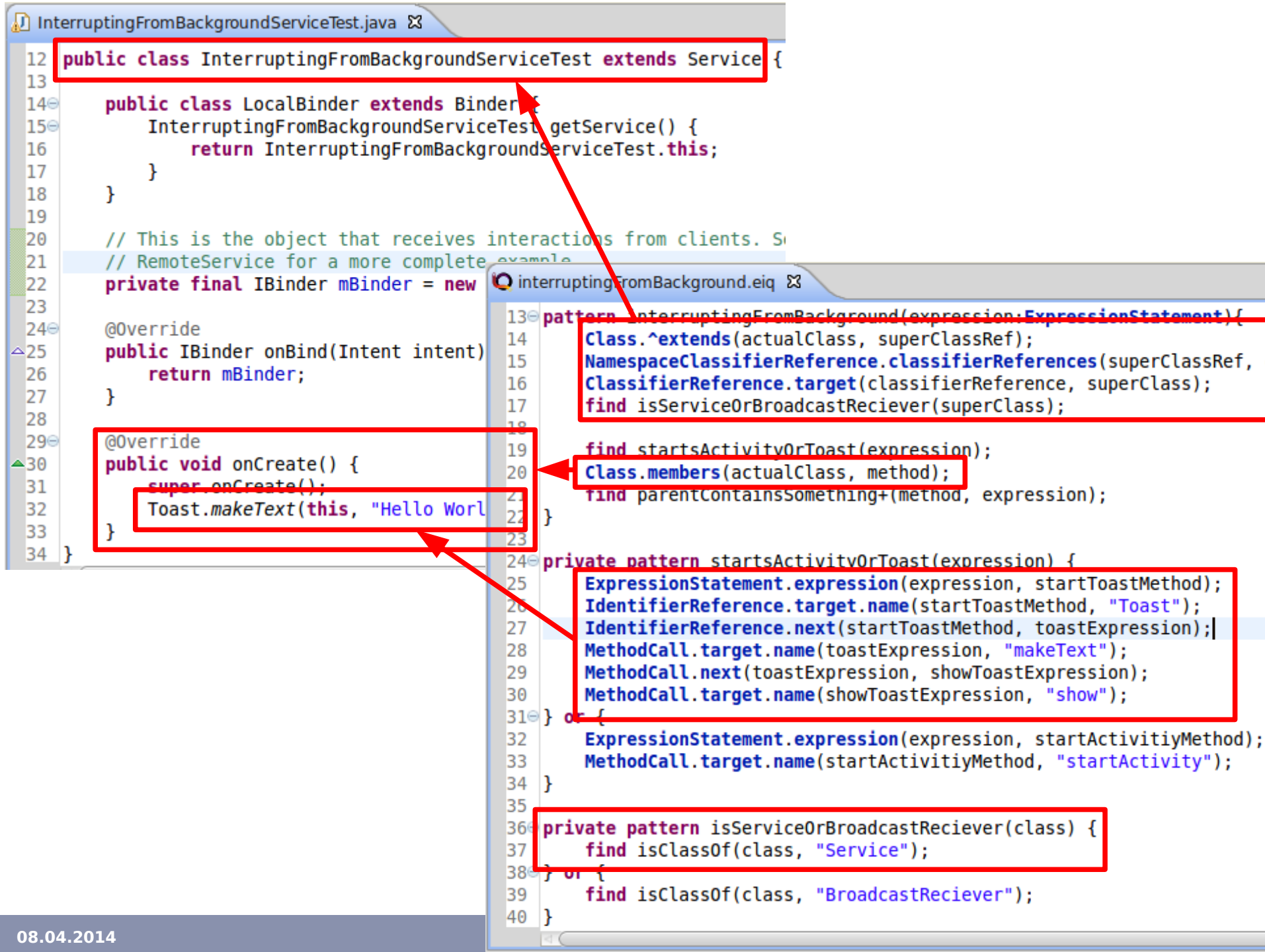
Demo

Structural Specification of Quality Smells

http://www.jamopp.org/

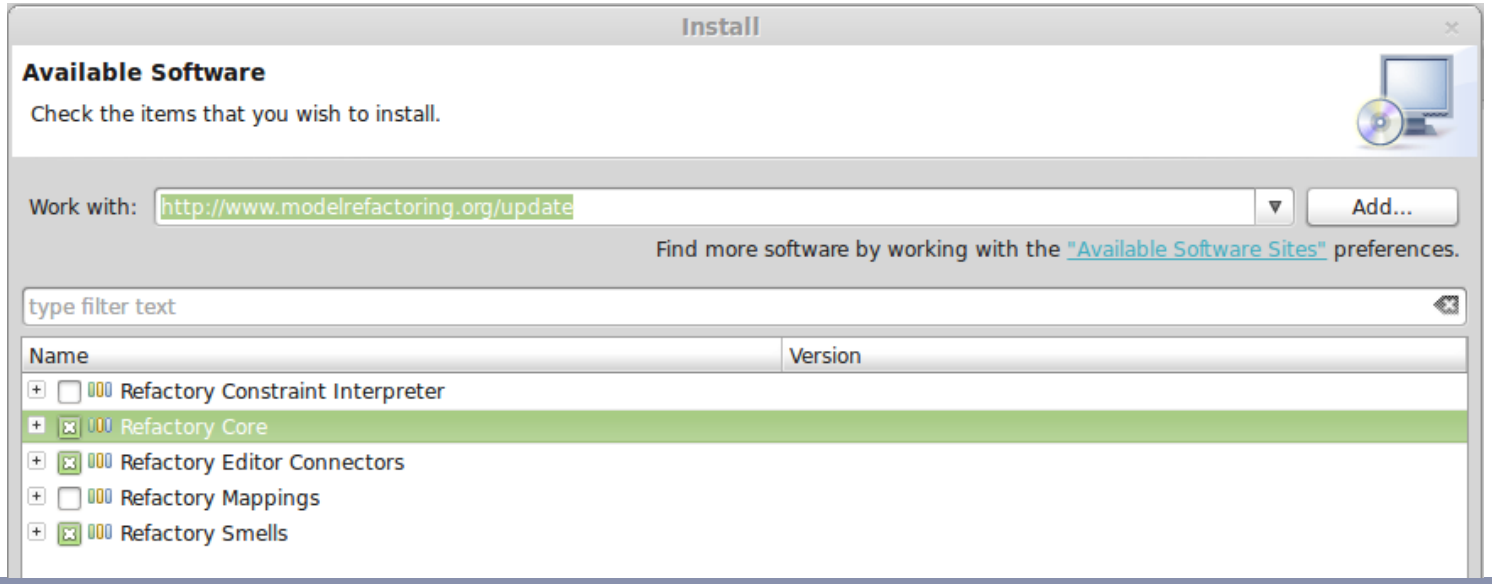
+

http://eclipse.org/incquery/


InterruptingFromBackgroundServiceTest.java
public class InterruptingFromBackgroundServiceTest extends Service {
public class LocalBinder extends Binder {
InterruptingFromBackgroundServiceTest getService() {
return InterruptingFromBackgroundServiceTest.this;
}
}
// This is the object that receives interactions from clients. S
// RemoteService for a more complete example
private final IBinder mBinder = new
@Override
public IBinder onBind(Intent intent)
return mBinder;
}
@Override
public void onCreate() {
super.onCreate();
Toast.makeText(this, "Hello Worl
}
}
interruptingFromBackground.eiq
pattern interruptingFromBackground(expression:ExpressionStatement){
Class.^extends(actualClass, superClassRef);
NamespaceClassifierReference.classifierReferences(superClassRef,
ClassifierReference.target(classifierReference, superClass);
find isServiceOrBroadcastReciever(superClass);
find startsActivityOrToast(expression);
Class.members(actualClass, method);
find parentContainsSomething+(method, expression);
}
private pattern startsActivityOrToast(expression) {
ExpressionStatement.expression(expression, startToastMethod);
IdentifierReference.target.name(startToastMethod, "Toast");
IdentifierReference.next(startToastMethod, toastExpression);
MethodCall.target.name(toastExpression, "makeText");
MethodCall.next(toastExpression, showToastExpression);
MethodCall.target.name(showToastExpression, "show");
} or {
ExpressionStatement.expression(expression, startActivitiyMethod);
MethodCall.target.name(startActivitiyMethod, "startActivity");
}
private pattern isServiceOrBroadcastReciever(class) {
find isClassOf(class, "Service");
} or {
find isClassOf(class, "BroadcastReciever");
}

How to Install?

- Download Eclipse Modeling Tools:
  http://eclipse.org/downloads/packages/eclipse-modeling-tools/keplersr2
- Install Refactory from update site:
  http://www.modelrefactoring.org/update
  http://www.modelrefactoring.org/update_trunk
- Download source project
  http://www.modelrefactoring.org/smell_catalog/org.emftext.language.java.refactoring.smell.android.zip

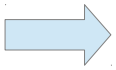


How to Use Existing Quality Smells?

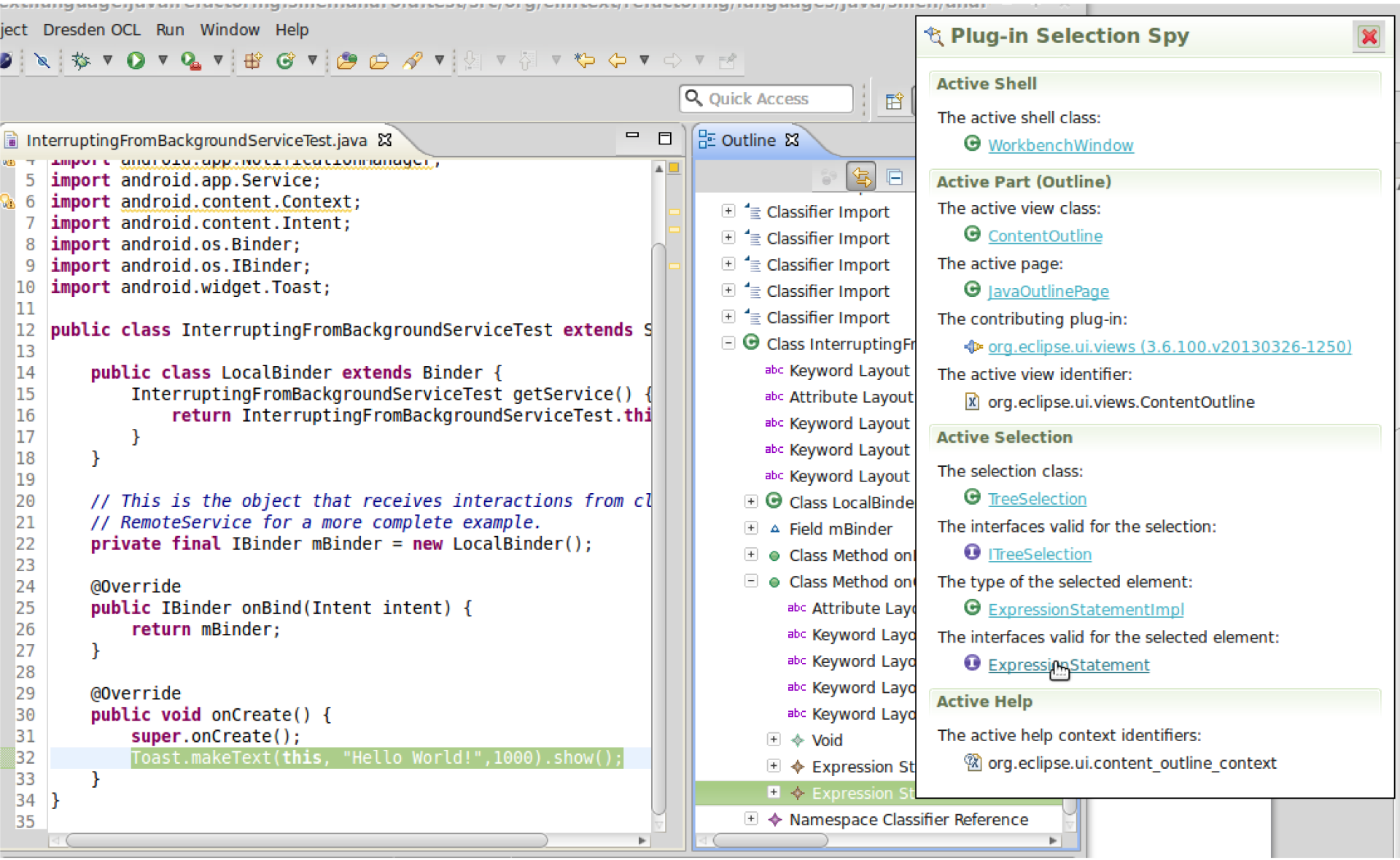
- Start runtime Eclipse
- Import test plugin http://www.modelrefactoring.org/smell_catalog/org.emftext.language.java.refactoring.smell.android.test.zip
- Open Query Explorer
- Open Java file in EMFText java Editor and press ▶
- Open query (*.eiq) file and press ▶

How to Determine Java Metaclasses?

- Create example class containing the desired structure
- Right-click → Open With → EMFText java Editor
- Navigate through code and observe properties view and selection in outline
- Alt + Shift + F1 on selection in outline → Plug-in Selection Spy opens



Screenshot next slide


ject Dresden OCL Run Window Help
Quick Access
InterruptingFromBackgroundServiceTest.java
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.widget.Toast;
public class InterruptingFromBackgroundServiceTest extends S
public class LocalBinder extends Binder {
InterruptingFromBackgroundServiceTest getService() {
return InterruptingFromBackgroundServiceTest.thi
}
}
// This is the object that receives interactions from cl
// RemoteService for a more complete example.
private final IBinder mBinder = new LocalBinder();
@Override
public IBinder onBind(Intent intent) {
return mBinder;
}
@Override
public void onCreate() {
super.onCreate();
Toast.makeText(this, "Hello World!",1000).show();
}
}
Outline
Classifier Import
Classifier Import
Classifier Import
Classifier Import
Classifier Import
Class InterruptingFr
Keyword Layout
Attribute Layout
Keyword Layout
Keyword Layout
Keyword Layout
Class LocalBinde
Field mBinder
Class Method onI
Class Method onC
Attribute Layo
Keyword Layo
Keyword Layo
Keyword Layo
Keyword Layo
Void
Expression St
Expression St
Namespace Classifier Reference
Plug-in Selection Spy
Active Shell
The active shell class:
WorkbenchWindow
Active Part (Outline)
The active view class:
ContentOutline
The active page:
JavaOutlinePage
The contributing plug-in:
org.eclipse.ui.views (3.6.100.v20130326-1250)
The active view identifier:
org.eclipse.ui.views.ContentOutline
Active Selection
The selection class:
TreeSelection
The interfaces valid for the selection:
ITreeSelection
The type of the selected element:
ExpressionStatementImpl
The interfaces valid for the selected element:
ExpressionStatement
Active Help
The active help context identifiers:
org.eclipse.ui.content_outline_context

Organisation

- 3 groups:
  - Ye Song, Xin Yang
  - Willi Zobel, Nico Braunisch
  - Peter Höhne, Ronny Marx
- Next meeting 23.04.14
  - Time: 10:00? 13:00? 14:00? 15:00? 16:00? other?
- 2 phases:
  - Specification of IncQuery patterns
  - Specification of Refactorings
- Meeting after each phase
- Communication via mailinglist: kp-qs@mail-st.inf.tu-dresden.de

Bibliography

[Fowler, 1999] Fowler: *Improving the Design of Existing Code*, Addison-Wesley, 1999

[Opdyke+, 1990] Opdyke, Johnson: *Refactoring: An aid in designing application frameworks and evolving object-oriented systems*, SOOPPA Symposium 1990

[Reimann+, 2013a] Reimann, Aßmann: Quality-Aware Refactoring For Early Detection And Resolution Of Energy Deficiencies, 4th International Workshop on Green and Cloud Computing Management 2013