CARL VON OSSIETZKY universität OLDENBURG

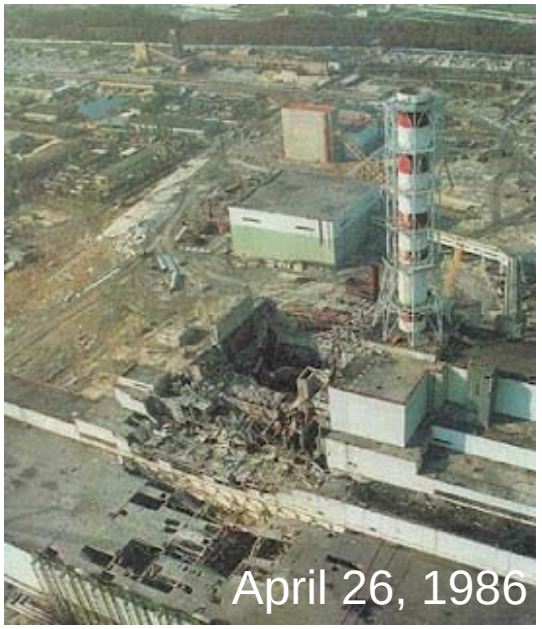
Model-driven Reengineering for a Blue Planet

- Refactoring for Energy Efficiency -

Andreas Winter

joint work with
Marion Gottschalk
Jan Jelschen
Mirco Josefiok

"Energy Turnaround"



not far away from here

"Energy Turnaround"

2020: have 35% green power
2022: switch off all nuclear power plants

online: March 1, 1986 – Sept. 9, 1988

March 11, 2011


online: Mar
Mar

How will

Software Engineering
Software Language Engineering
Model-driven Software Reengineering

help to manage the

energy turnover

Green Software Engineering

Current Situation

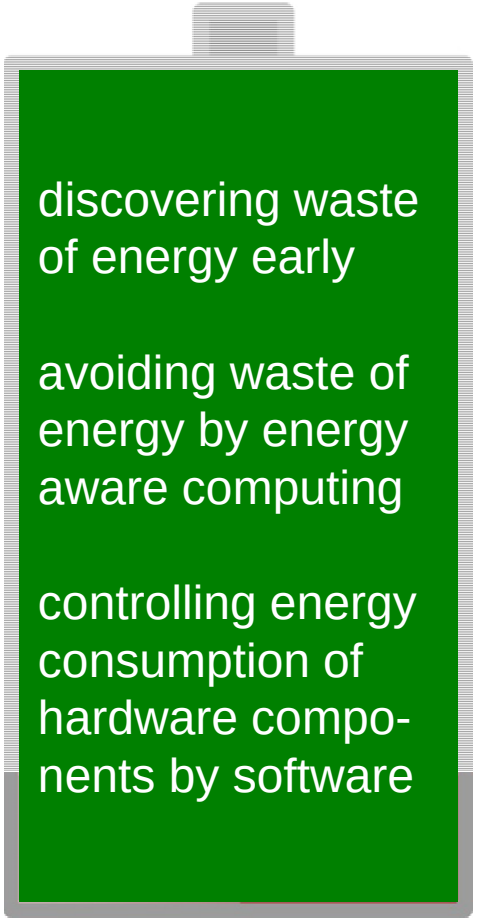
- ICT consumes more than 10% of Germanys energy budget (2007)
- ICT produces more CO_2 emissions than the entire German aviation sector

Research in Computer Science

- focusses on saving energy
  - in computing centers
  - on operating system level
  - on hardware level
- does not consider software applications

Applying Software-Reengineering

- analyzing code for wasting energy
- improving energy inefficient code
- providing information on user behavior to enable more energy efficient task scheduling by OS



Quality Goal
Energy Efficiency

Reengineering = Quality Improvement

Static Analysis

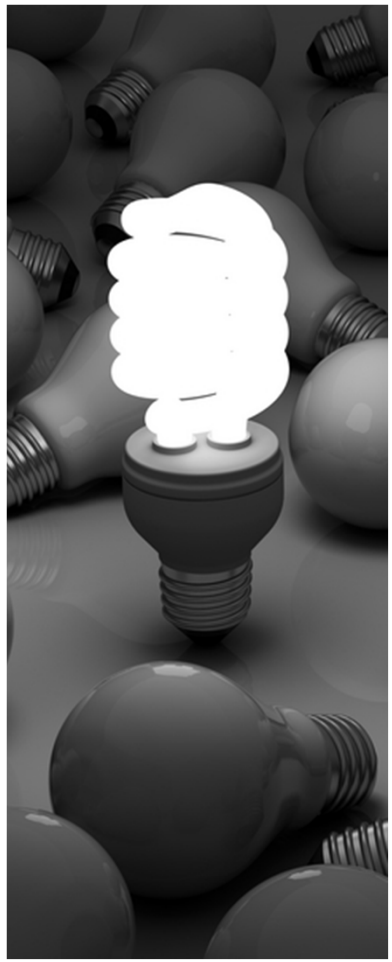
- define and detect “energy code smells”
- define metrics for energy-efficiency classification

Dynamic Analysis

- log application behavior and correlate with battery drain
- monitor user behavior and deduct usage patterns
- detect periodic wake-ups and synchronize

Refactoring

- replace energy-inefficient code
- introduce strategy patterns to enable apps to make energy-aware choices



Refactoring = Identification + Restructuring

Energy Code Smells

- depict energy-inefficient patterns in code

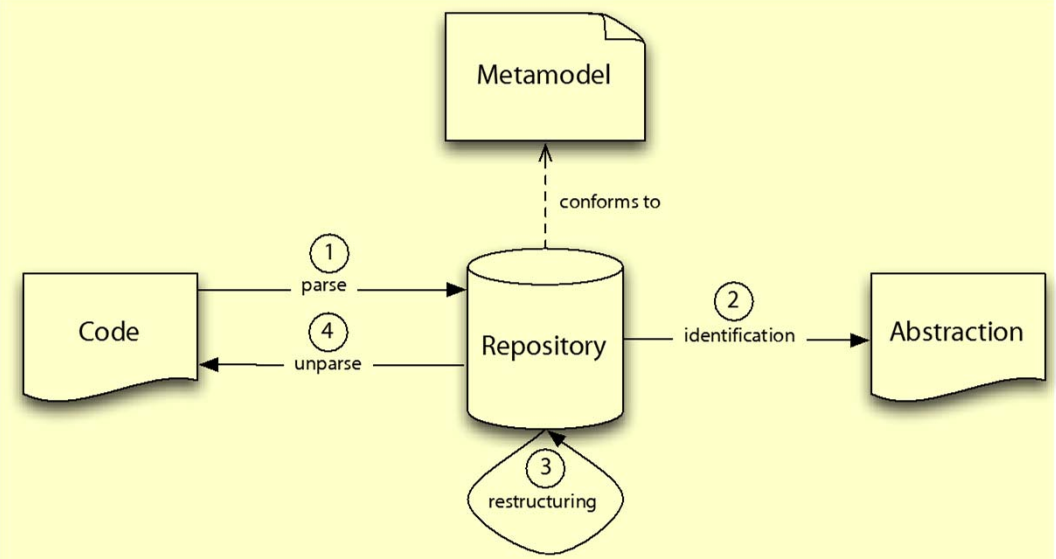
Process

① Preparation:
- parse source to TGraphs

② Identification:
- identify code patterns by graph queries (GReQL)

③ Restructuring:
- improve code through graph transformations

④ Post processing:
- unparse TGraphs to source

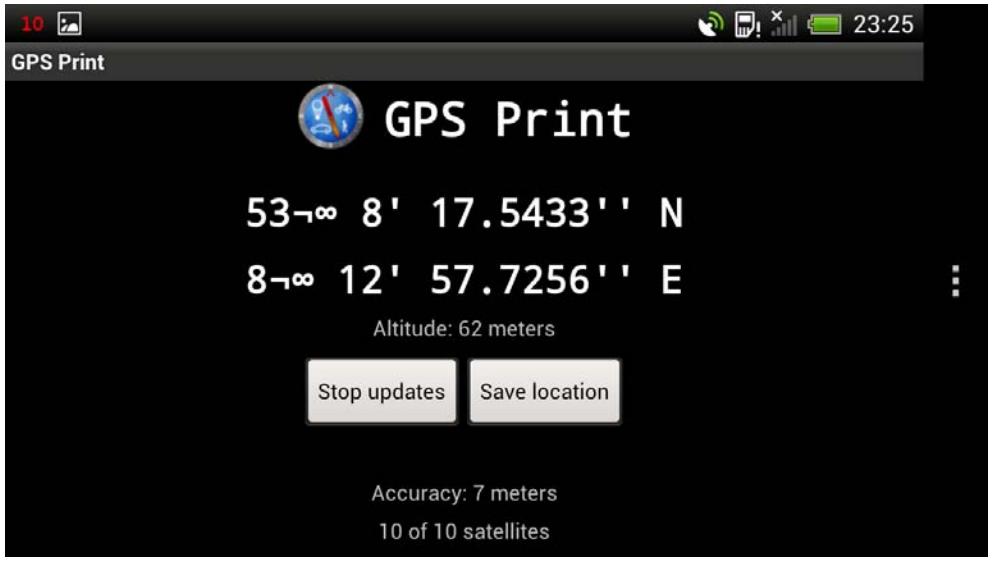


Reengineering Architecture

Example: GPSPrint

Energy Code Smell

binding resources too early



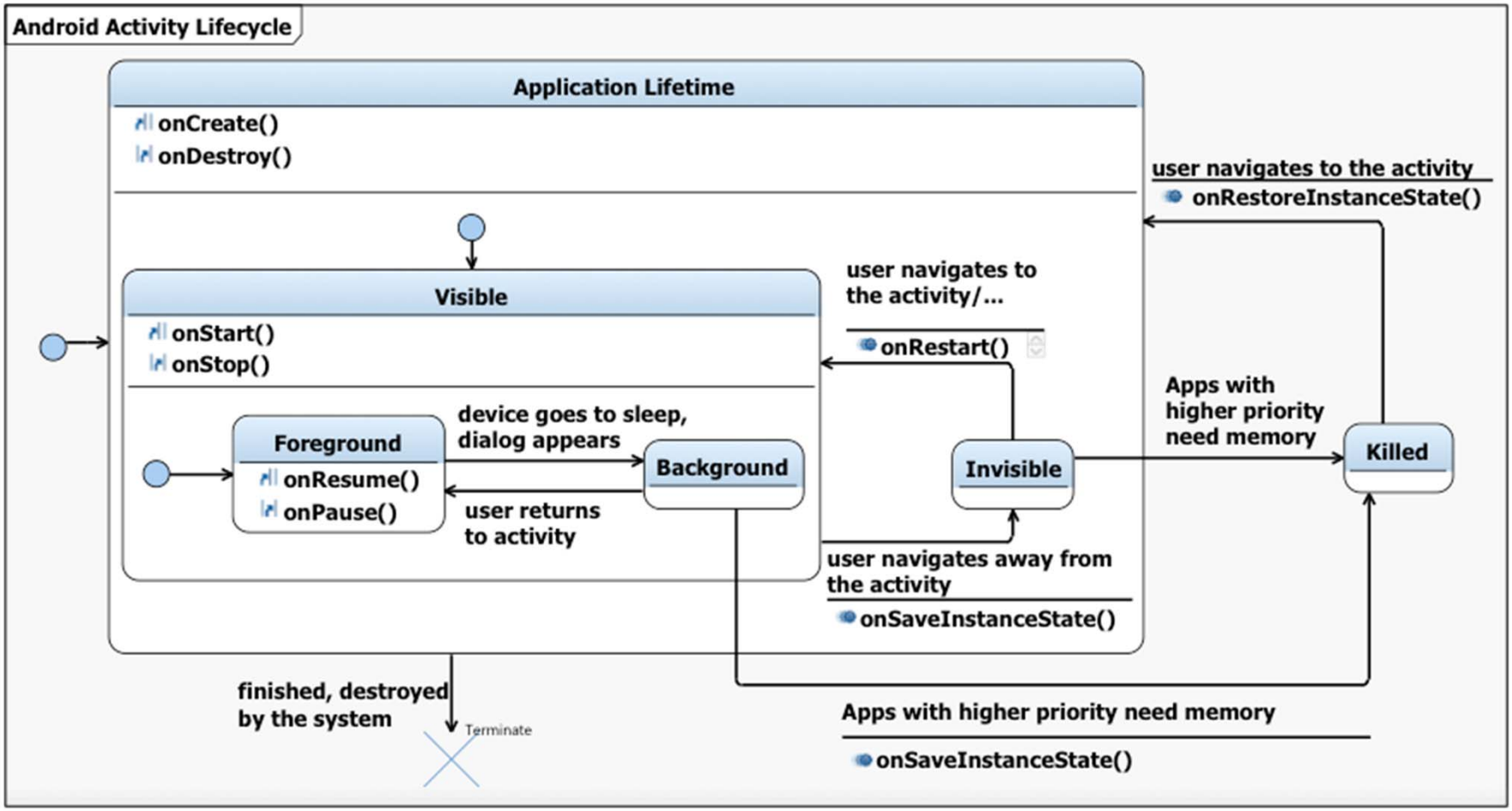

```
public class GpsPrint extends Activity
    implements OnClickListener,Listener,
    LocationListener {
[...]
  public void onCreate(Bundle
      savedInstanceState) {
[...]
    LocationManager lm=(LocationManager)
        this.getSystemService(Context.
        LOCATION_SERVICE);
    if(lm.getAllProviders().contains(
        LocationManager.GPS_PROVIDER)) {
      if(lm.isProviderEnabled(
          LocationManager.GPS_PROVIDER)){
        lm.addGpsStatusListener(this);
        lm.requestLocationUpdates(LocationManager.
            GPS_PROVIDER, 1000, 0, this);
        status_view.setText(
            "GPS service started");
        } else {
        status_view.setText(
            "Please enable GPS");
        save_location_button.setEnabled(
            false); }
[...] }
```

Potential waste of energy

- energy inefficient use of GPS sensor

[http://play.google.com/store/apps]

Resource Utilization in Android



[cmp. Android Developers: “Activities”, 2012. http://developer.android.com/guide/topics/fundamentals/activities.html]

Example: Binding resources too early

Identification

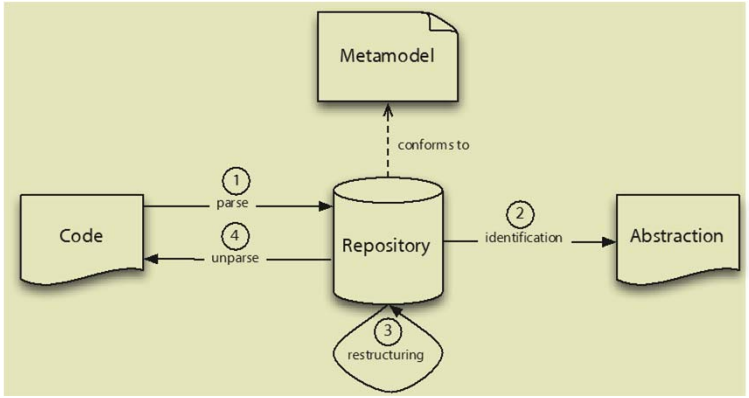
- Where is the GPS sensor turned on?
- Where does the `locationManager` calls `requestLocationUpdates()` for the GPS?

Bad energy smell

- call of `requestLocationUpdates()` in `onCreate()`

```
1  public class GpsPrint extends Activity
2      implements OnClickListener,Listener,
3      LocationListener {
4  [...]
5    public void onCreate(Bundle
6        savedInstanceState) {
7  [...]
8      LocationManager lm=(LocationManager)
9          this.getSystemService(Context.
10         LOCATION_SERVICE);
11     if(lm.getAllProviders().contains(
12         LocationManager.GPS_PROVIDER)) {
13       if(lm.isProviderEnabled(
14           LocationManager.GPS_PROVIDER)){
15         lm.addGpsStatusListener(this);
16         lm.requestLocationUpdates(LocationManager.
17             GPS_PROVIDER, 1000, 0, this);
18         status_view.setText(
19             "GPS service started");
20         } else {
21         status_view.setText(
22             "Please enable GPS");
23         save_location_button.setEnabled(
24             false); }
25 [...] }
26 [...]
27   public void onPause() {
28 [...]
29     lm.removeUpdates(this);
30 [...]
31   public void onResume() {
32 [...]
33     lm.requestLocationUpdates(
34         LocationManager.GPS_PROVIDER,
35         1000, 0, this);
36 [...] }
37 }
```

Metamodel



Mission

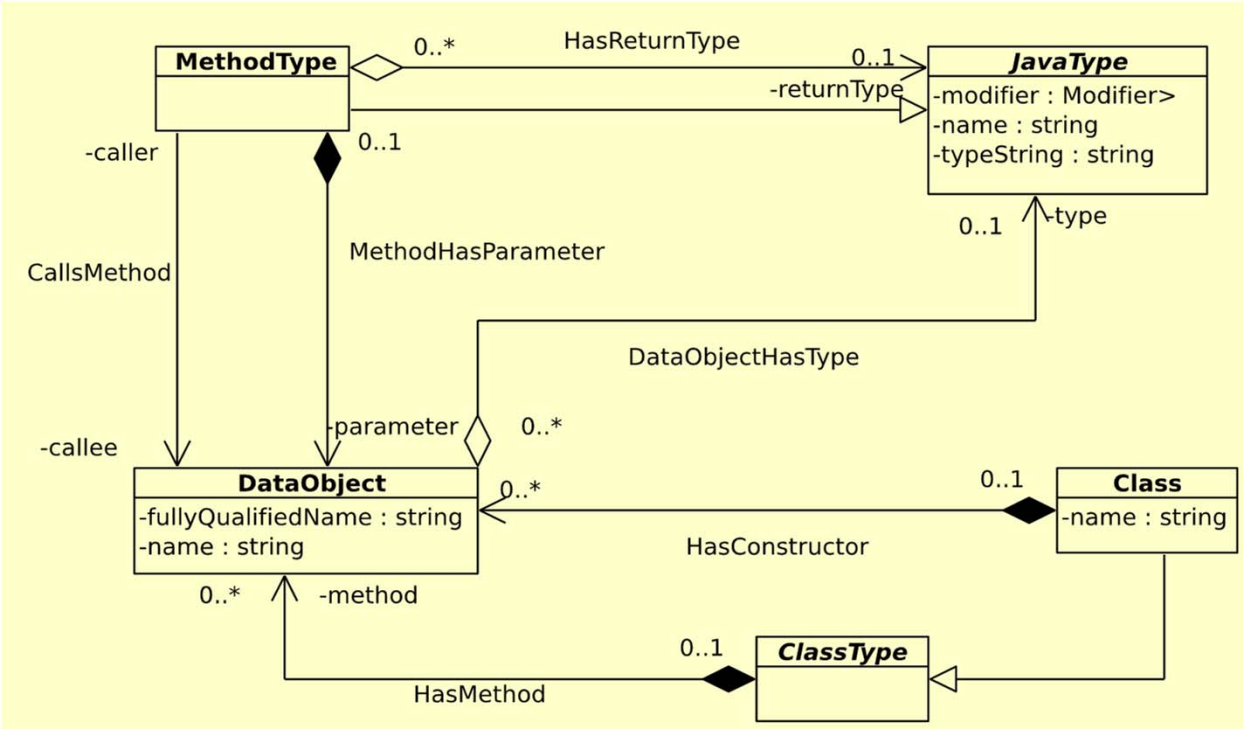
- representing Java code for efficient querying

soamig Metamodel

- fine grained Java representation
- contains
  - 86 node types
  - 67 edge types

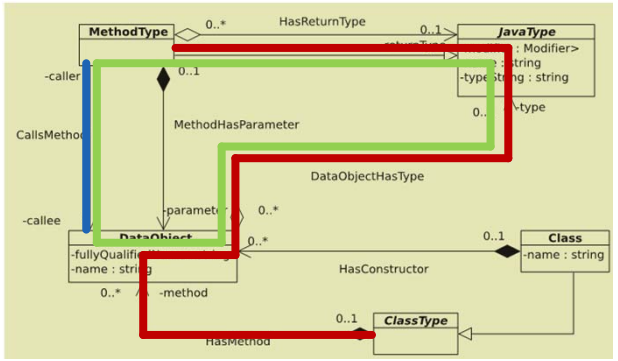
GPSPrint Graph

- contains
  - 9034 nodes
  - 14880 edges



Identification

finding the smelling call of `requestLocationUpdates()`



JGraLabUI – GPSPrint.tg

File Edit Query

```
from
        onCreate, caller : V{frontend.java.MethodType},
        actClass : V{frontend.java.Class},
        superClass, callee : V{frontend.java.DataObject}
with
        onCreate.name = "onCreate" and
        superClass.fullyQualifiedName = "android.app.Activity" and
        callee.name = "requestLocationUpdates" and
        callee <--{frontend.java.ext.CallsMethod} caller
        (<--{frontend.java.DataObjectHasType} <--{frontend.java.ext.CallsMethod})*
        onCreate <--{frontend.java.DataObjectHasType}
        <--{frontend.java.HasMethod} actClass
        -->{frontend.java.HasSuperClass} superClass
report
        actClass.name, caller.name
end
```

((GpsPrint, onCreate))

direct call
indirect call
link to enclosing class (directly)

AST JAVA Graph opened.

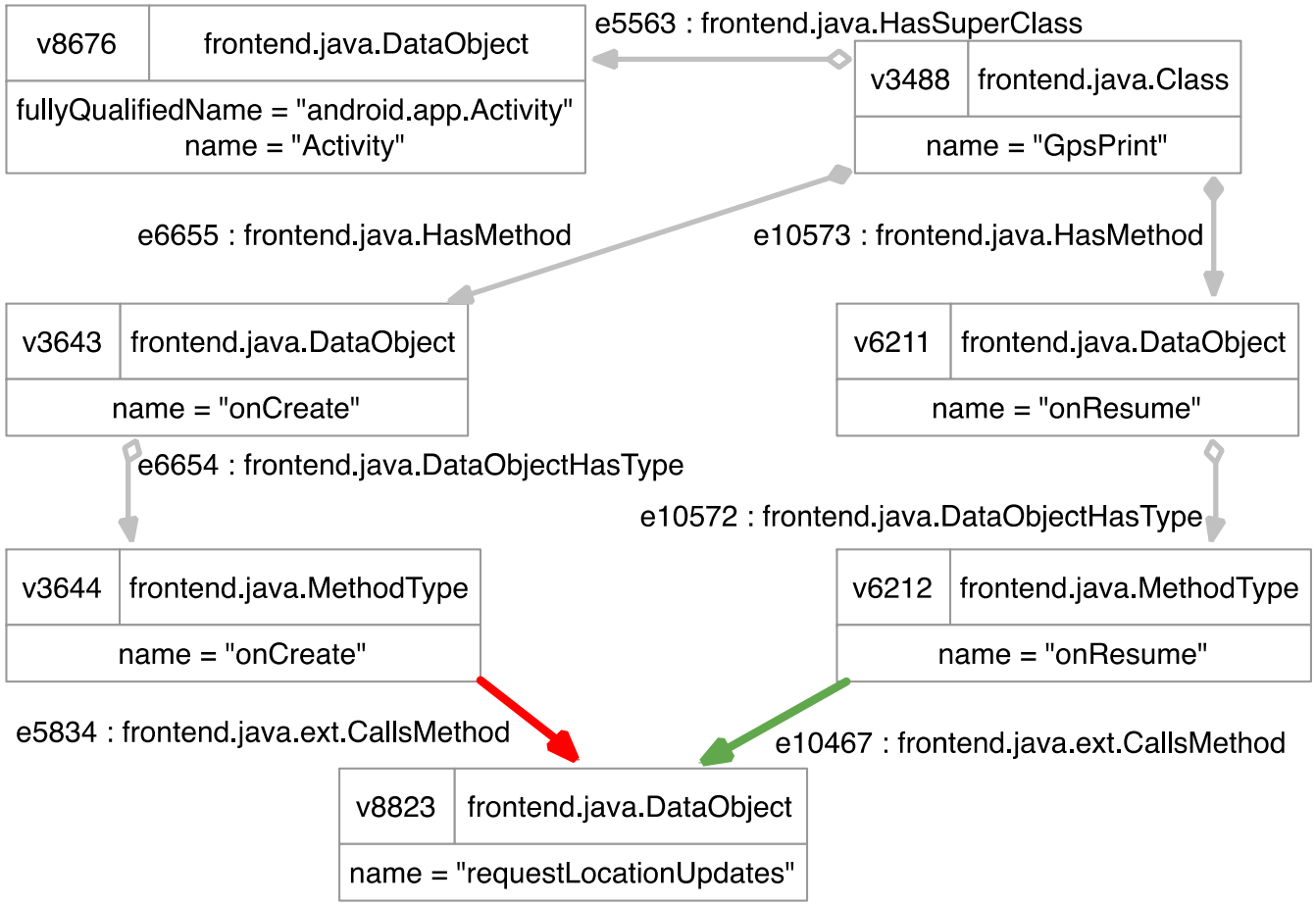
Restructuring

- identify energy code smell
- identify code to be refactored
- do the refactoring

```
1  public class GpsPrint extends Activity
2      implements OnClickListener,Listener,
3      LocationListener {
4  [...]
5    public void onCreate(Bundle
6        savedInstanceState) {
7  [...]
8      LocationManager lm=(LocationManager)
9          this.getSystemService(Context.
10         LOCATION_SERVICE);
11     if(lm.getAllProviders().contains(
12         LocationManager.GPS_PROVIDER)) {
13       if(lm.isProviderEnabled(
14          LocationManager.GPS_PROVIDER)){
15         lm.addGpsStatusListener(this);
16         lm.requestLocationUpdates(LocationManager.
17             GPS_PROVIDER, 1000, 0, this);
18         status_view.setText(
19             "GPS service started");
20         } else {
21         status_view.setText(
22             "Please enable GPS");
23         save_location_button.setEnabled(
24             false); }
25 [...] }
26 [...]
27   public void onPause() {
28 [...]
29     lm.removeUpdates(this);
30 [...] }
31   public void onResume() {
32 [...]
33     lm.requestLocationUpdates(
34         LocationManager.GPS_PROVIDER,
35         1000, 0, this);
36 [...] }
37 }
```

Before Refactoring

Restructuring – Graph Transformation



Restructuring

Graph Transformation

- using JGraLab

```
1  public class DeleteEdgeGPSPrint extends
       Transformation<Object> {
2   [...]
3   @Override
4   protected Object transform() {
5    OpenSaveTG tg = new OpenSaveTG();
6    // loads TGraph (see figure 3)
7    tGraph = tg.openTG("GPSPrint-pruned07.tg");
8    try {
9     // searches for super class of 'GpsPrint'-class
10    [...]
11    // searches for MethodType 'onCreate' which calls
          the dataObject 'requestLocationUpdates'
12    for (Vertex v : tGraph.vertices()) {
13     if (v.getAttribute("name").equals("onCreate") &&
           (v.isBefore(superClass) || v.isAfter(
          superClass))) {
14      for (Edge e : v.incidences()) {
15      // considers only out-going edges
16       if (e.getId() > 0) {
17        if (e.getThat().getAttribute("name").equals("
              requestLocationUpdates")) {
18         // deletes unnecessary edge
19         e.delete();
20         tg.saveTG("GPSPrintModified");
21         return "";
22        }
23       }
24  [...]
25 }
```

Further Classes of Energy Code Smells

Loop Bug
- application is repeating the same activity

Dead Code
- source code which is never used, but loaded to memory

In-line method
- replacing method calls by the body of the called method

Moving too much data
- unnecessary communication between processor and memory

Immortality Bug
- applications respawning after explicitly being killed by the user

Redundant storage of data
- different methods of store the same data in memory

Using expensive resources
- energy-expensive resources are exchange to “cheaper” alternatives (algorithms and hardware components)

Types of Energy Refactorings
- classical refactoring (Fowler)
- general energy refactorings
- platform specific

Types of Energy Analysis
- static analysis
- dynamic analysis

Remark
- energy refactorings change behavior of apps

I like

SoTeSoLa

Model driven Reengineering is applicable to improve energy-behavior of software

- using TGraphs for representation
- applying GReQL for querying
- utilizing JGraLab for code restructurings

Further investigation on energy code smells

- finding and defining more energy code smell classes and their associated refactorings
- providing an platform independent means for energy quantification
  - to prove benefits of energy code smell refactorings