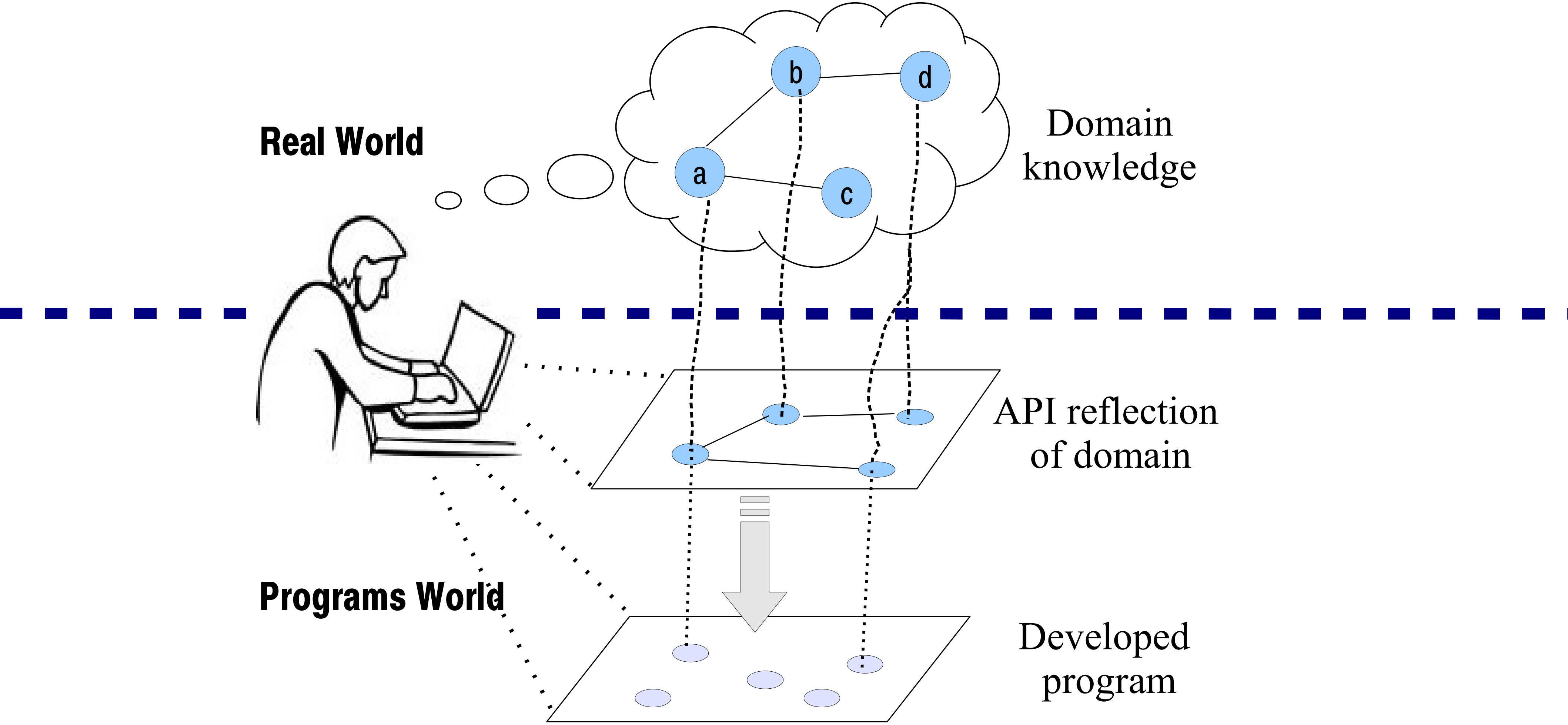
Evaluating the Reference and Representation of Domain Concepts in APIs

Daniel Ratiu, Jan Jürjens

ICPC
12 June 2008

Domain specific APIs reflect the domain knowledge



APIs' Quality through the Practitioners Eyes

- APIs should have (Mathieu Jacques [www.codeproject.com])
  - Good visibility
  - Good conceptual model
  - Good mapping (natural analogies for representing concepts)

- APIs can be seen as machines (Bill Veners [www.artima.com])
  - APIs “shape” = names, types methods
  - APIs “semantics” = domain concepts that the API implements

- Good vs. Bad APIs (Michi Henning)
  - Good APIs a joy to use
    - “Work without friction and disappear from sight”
      - Right call is available at the right time
  - Bad APIs are hard to use

Distilling Practitioners Knowledge

- What is the **explicitness** in the implementation of concepts in the API?
  - What is the **conceptual complexity** of the API?

- How uniform can the API users **combine the concepts** at the API level?
  - How **difficult is to make errors** by realizing combinations that make no sense?

To answer these questions we need *explicit* relations between

- **domain concepts**
- **API program elements**

Reference of Concepts

- Reference of concepts = program elements that refer to the concept

$$\overrightarrow{Ref}: C \rightarrow \wp(P) \qquad\qquad \overleftarrow{Ref}: P \rightarrow \wp(C)$$

  - Define the “Shape” of the API
  - Influence how concepts can be found and addressed at the API level

- Example:

```
class Circle extends Figure {
  public Point _pos;
  public void setPosition(Point2D position) { ... }
  public void setRadiusAndColor(int radius, int color) { ... }
}
```

$$\overrightarrow{Ref}(\text{radius}) = \{\mathit{setRadiusAndColor}, \mathit{radius}\}$$

$$\overrightarrow{Ref}(\text{color}) = \{\mathit{setRadiusAndColor}, \text{color}\}$$

$$\overrightarrow{Ref}(\text{position}) = \{\mathit{position}, \text{\_pos}\}$$

$$\overleftarrow{Ref}(\mathit{setRadiusAndColor}) = \{\text{radius}, \text{color}\}$$

$$\overleftarrow{Ref}(\text{color}) = \{\text{color}\}$$

$$\overleftarrow{Ref}(\text{radius}) = \{\text{radius}\}$$

$$\overleftarrow{Ref}(\mathit{position}) = \{\text{position}\}$$

$$\overleftarrow{Ref}(\text{\_pos}) = \{\text{position}\}$$

Representation of Concepts

- Representation of concepts = types of variables that refer to a concept

$$\overrightarrow{Rep} : C \rightarrow \wp\left(P_{type}\right) \qquad \overleftarrow{Rep} : P_{type} \rightarrow \wp\left(C\right)$$

  - Define the “Semantics” of the API
  - Influence how concepts can be combined at the API level

- Example:

```
class Circle extends Figure {
  public Point _pos;
  public void setPosition(Point2D position) { ... }
  public void setRadiusAndColor(int radius, int color) { ... }
}
```

$$\overrightarrow{Rep}(\text{radius}) = \{\text{int}\}$$
$$\overrightarrow{Rep}(\text{color}) = \{\text{int}\}$$
$$\overrightarrow{Rep}(\text{position}) = \{Point\ , Point2D\}$$

$$\overleftarrow{Rep}(\text{int}) = \{\text{radius}\ , \text{color}\}$$
$$\overleftarrow{Rep}(Point) = \{\text{position}\}$$
$$\overleftarrow{Rep}(Point2D) = \{position\}$$

Characterizing Reference

- Explicit reference = the program elements that refer only to a concept
- Reference explicitness ratio (ER)
  - The ratio of program elements that refer to a concept
  - Example:

$$ER = 10 \; / \; 11$$

```
class Circle extends Figure {
  public Point _pos;
  public void setPosition(Point2D position) { ... }
  public void setRadiusAndColor(int radius, int color) { ... }
}
```

- Conceptual complexity (CC)
  - The average number of concepts referred by program elements
  - Example:

$$CC = 12 \; / \; 11$$

Representation defects

- Representation overloading
  - A type is used to represent different concepts
  - Example:

$$\overleftarrow{Rep}(\text{int})=\{\text{radius}, \text{color}\}$$

→

```
int c = getColor();
int r = getRadius();
aCircle.setRadiusAndColor(c, r);
```

- Overloading ratio = the average number of concepts represented by the types

- Representation ambiguity
  - A concept is represented through several types
  - Example:

$$\overrightarrow{Rep}(\text{position})=\{\mathit{Point}, \mathit{Point2D}\}$$

→

```
Point p1 = aCircle._pos;
Point2D p2 = new Point2D(p1.getX(), p1.getY());
anotherCircle.setPosition(p2);
```

- Ambiguity ratio = the average number of types that represent a concept

Examples from Java

```
package java.rmi.server;
public class RMIClassLoader { ...
   public static Class loadClass(
                      String codebase,
                      String name,
                      ClassLoader defaultLoader) {...}
 ... }
```

```
package javax.rmi.CORBA;
public class Util { ...
  public static Class loadClass(
                      String className,
                      String remoteCodebase,
                      ClassLoader loader) {...}
 ... }
```

Type	OD
int	788
boolean	56
float	42
Object	39
String	39
double	32
MediaType	30

AWT

Type	OD
int	319
String	245
boolean	88
Object	62
Attribute	59
Region	51
Tag	40

SWING

Concept	AD
OldI	17
Src	16
Dst	12
Offset	6
Right	4
Left	4
Width	4

AWT

Concept	AD
Listener	11
Type	7
Height	6
Width	6
New	6
Name	5
Item	5

SWING

Thank you!