TECHNISCHE UNIVERSITÄT MÜNCHEN

Extracting Domain Ontologies from Domain Specific APIs

Daniel Ratiu, Martin Feilkas, Jan Jürjens

http://www4.in.tum.de/~ratiu/knowledge_repository.html

CSMR
1-4 April 2008

Situation Today

- Software maintenance is a knowledge intensive activity
  - Maintainers make use of a lot of technologies knowledge
  - Concepts location is one of the central program comprehension activities

but ...

Today's automatic program analyses do not use this knowledge

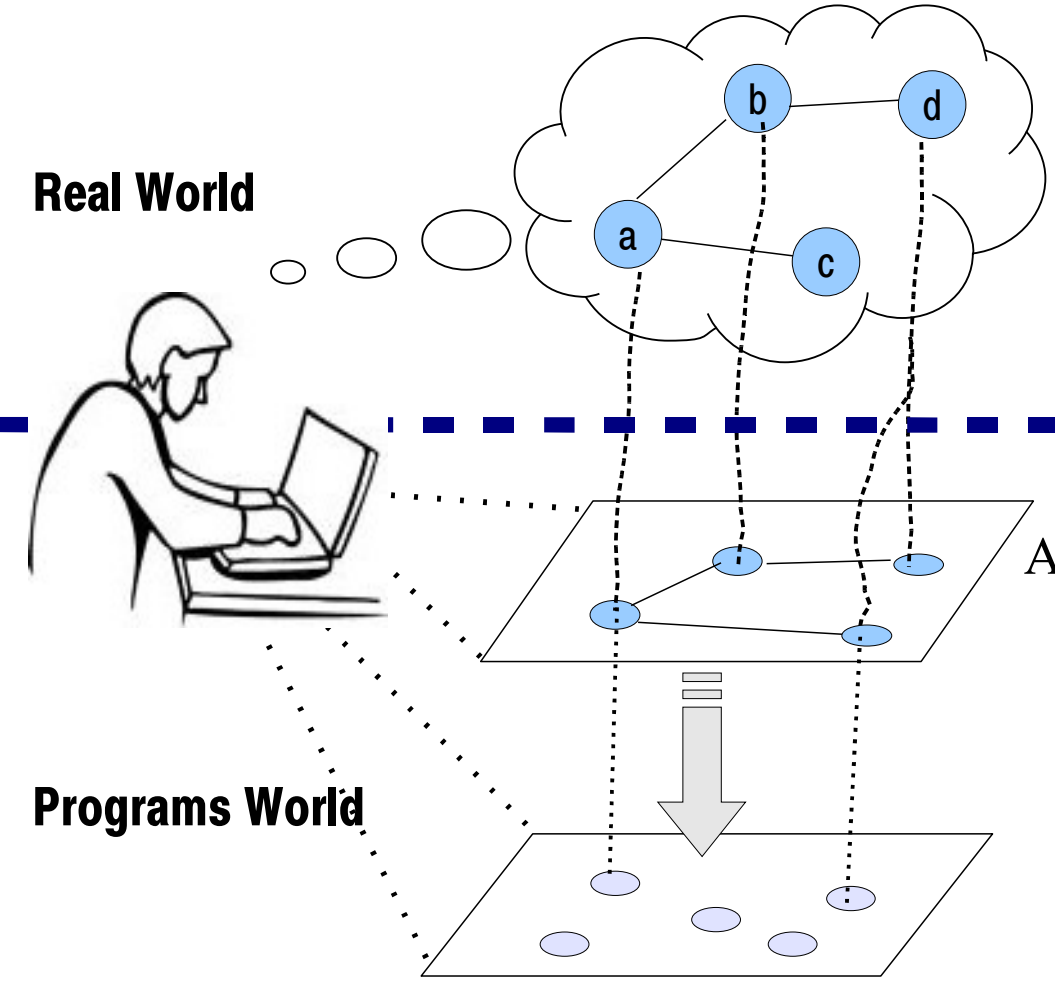
because ...

There is currently no knowledge base about the prog. technologies

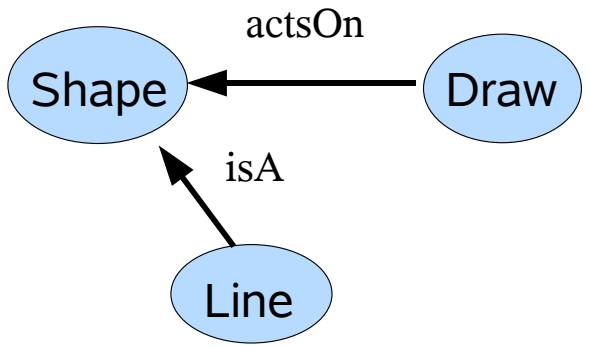
- Solution: a repository of ontologies about programming technologies knowledge

http://www4.in.tum.de/~ratiu/knowledge_repository.html

Domain specific APIs reflect the domain knowledge



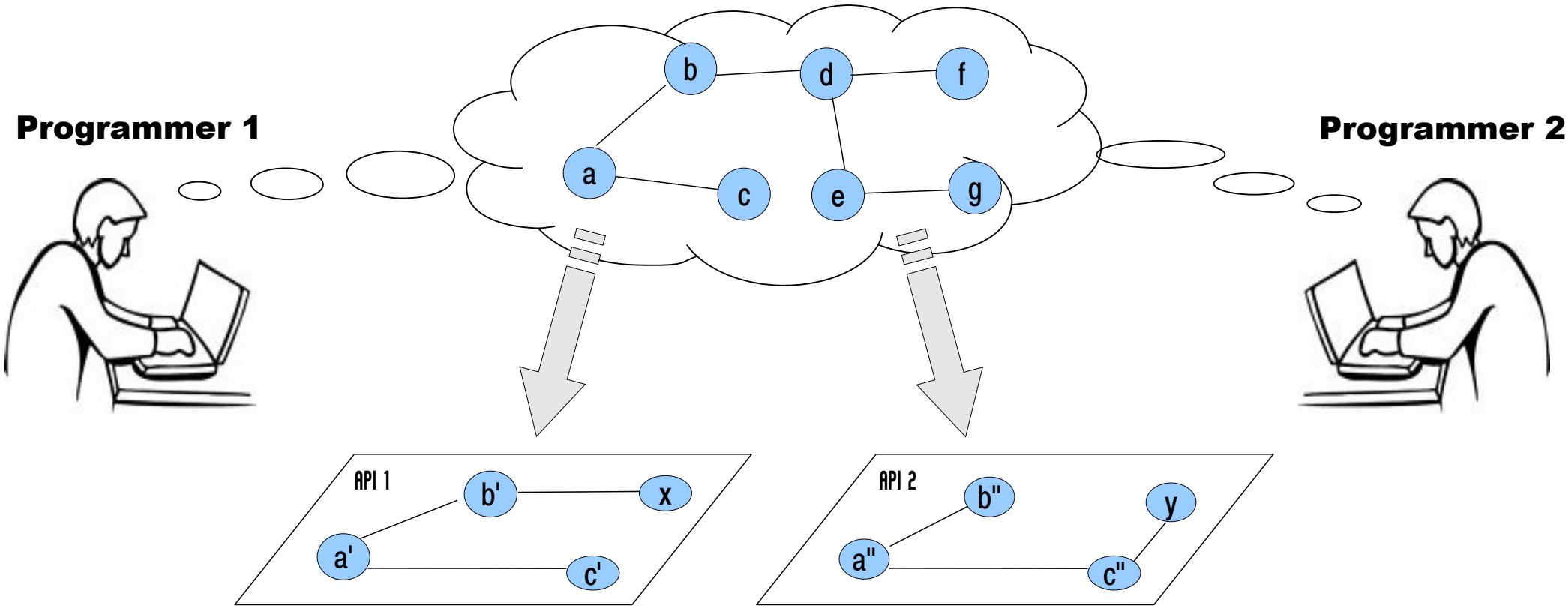
Example:




```
class Shape { ... }
class Line2D extends Shape { ... }
class Graphics2D {
      void draw(Shape aShape) { ... }
}
```

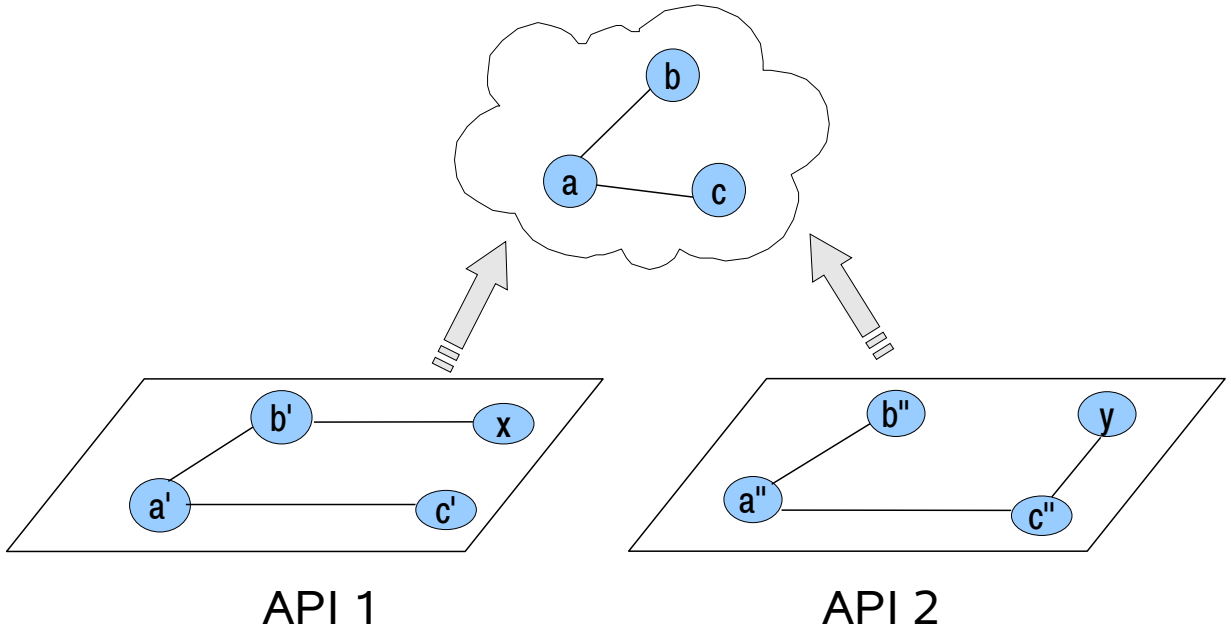
```
void clientProgram(Shape aShape) {
     aGraphics.draw(aShape);
}
```

Different APIs offer different Views over the Domain



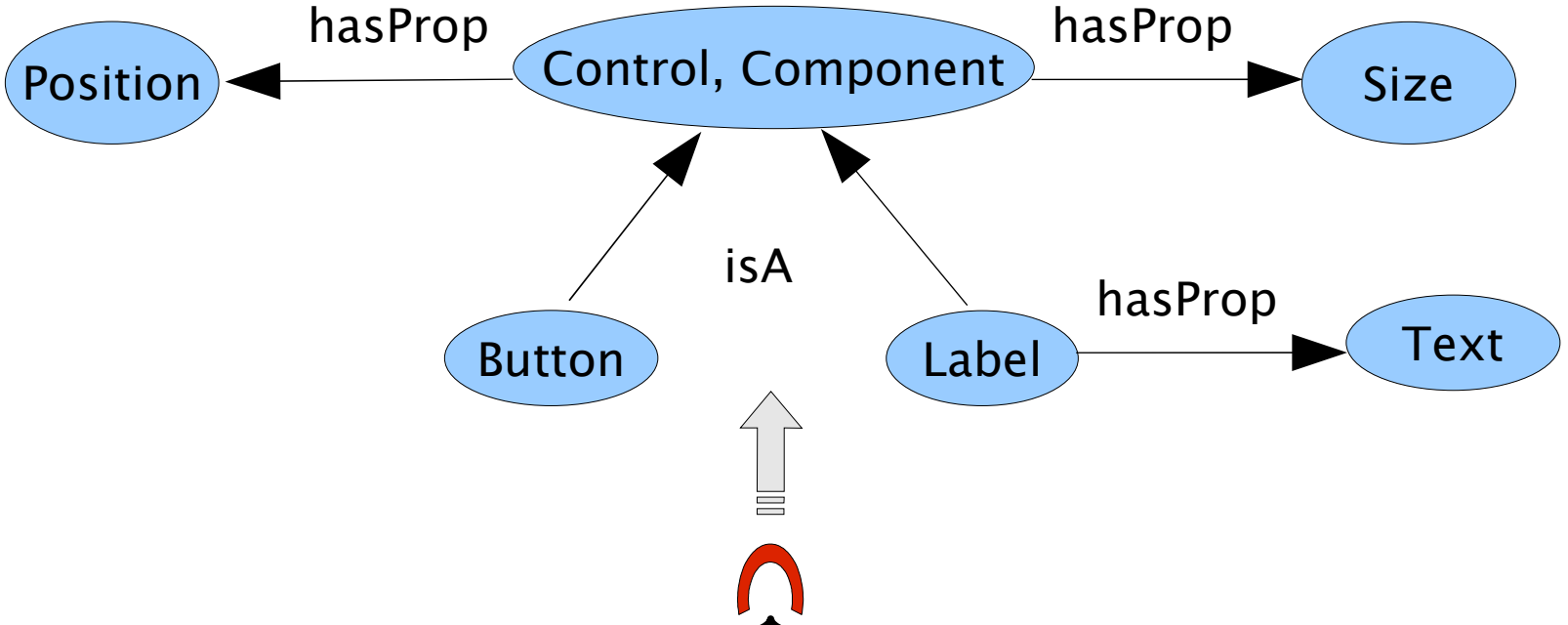
... but the **domain** is the same

Extracting Domain Knowledge from APIs



... by exploiting the similarities between the APIs

Example



Java AWT

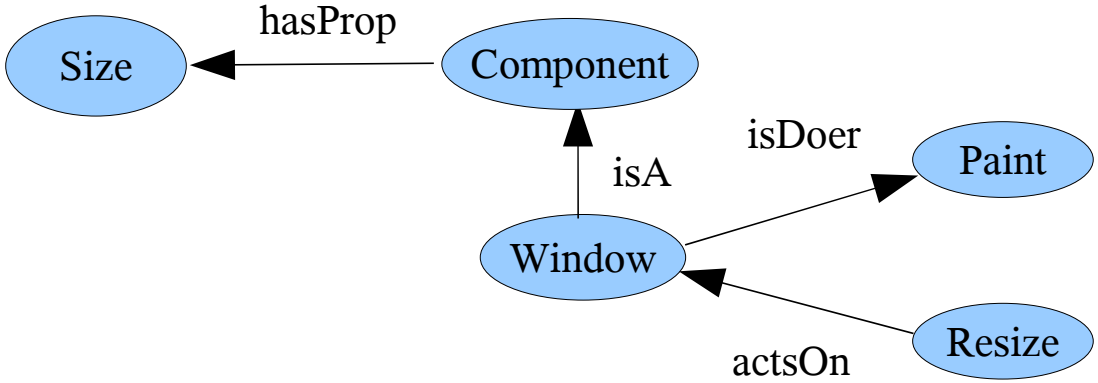
```
package java.awt;
class Component extends Object {
  int getSize() { ... }
  int getLocation() { ... }
}
class Button extends Component { ... }
class Label extends Component {
  String getText() { ... }
}
```

.Net Windows Forms

```
namespace  Windows.Forms;
class Control  :  ... {
  public Point Location { get; set; }
  public Size Size { get; set; }
  public string Text { get; set; }
}
class Label  :  Control  { ... }
class ButtonBase  :  Control  { ... }
```

Describing Domain Knowledge with Ontologies

- Describe the domain as light-weighted ontologies represented as graphs (set of triples: subject – verb - object):
  - Nodes – domain concepts
  - Edges – relations between the concepts
    - **isA -** between subordinate and superordinate
      - e.g. Window *isA* Component
    - **hasProperty** – between a concept and its properties
      - e.g. Component *hasProperty* Size
    - **actsOn** – between an action and the entities on which it is performed
      - e.g. Resize *actsOn* Window
    - **isDoer** – between an object and the
      - e.g. Window *isDoer* Paint



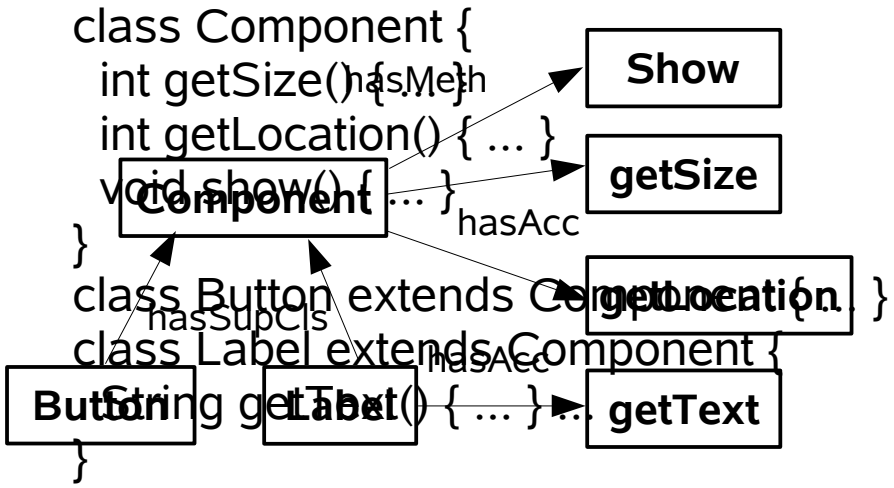
Extraction Steps

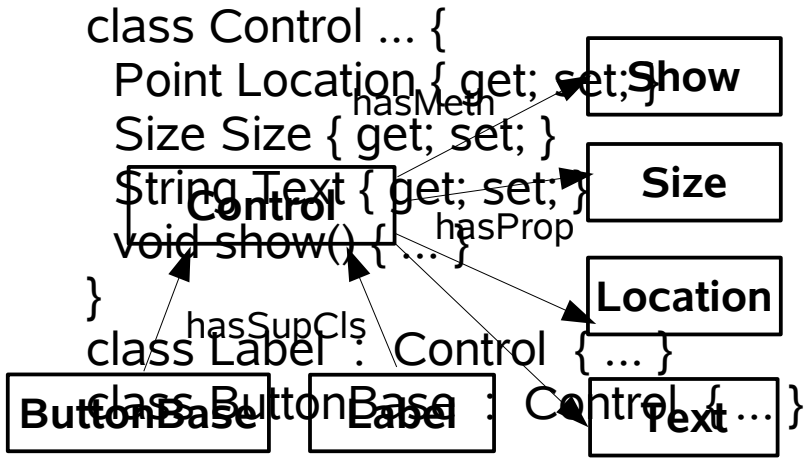
- **Step 1:** abstract the APIs in order to facilitate their comparison
- **Step 2:** inspect how are the abstract relations reflected in APIs
- **Step 3:** apply the ontology extraction algorithm
- **Step 4:** eliminate the noise

Challenges of the automatic extraction of domain knowledge:
1) uniform different implementations of the same real-world situation
2) eliminate the noise

Step 1: Abstracting APIs

- We abstract APIs as graphs
  - Nodes = (lexically normalized) names of public program elements
  - Edges = program relations between the program elements
    - hasSupCls, hasAcc (Java), hasAtt, hasProp (C#), hasMeth, hasPar, hasType, hasConstr

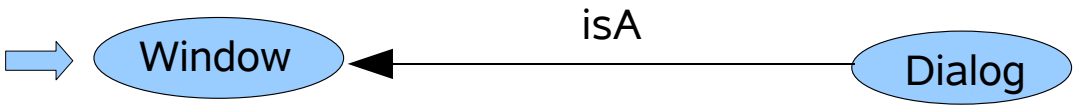




Step 2: Reflecting abstract relations in APIs

- The taxonomic relation (isA) is typically implemented as:
  - sub-classing (hasSupCls)
  - sequences of sub-classes (<hasSupCls, hasSupCls>)
  - the type of a variable (hasType)
- **Example:**

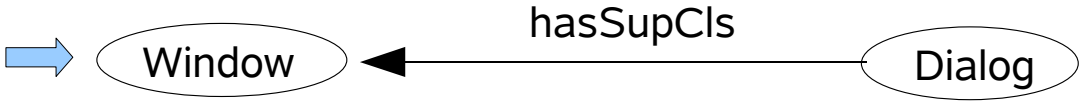
Dialog is a Window



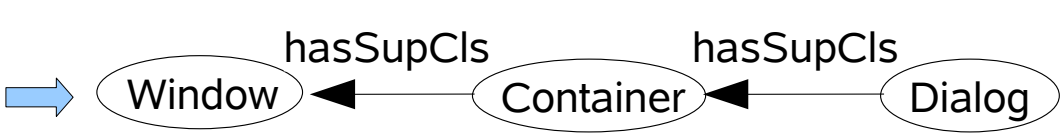
Ontology

APIs

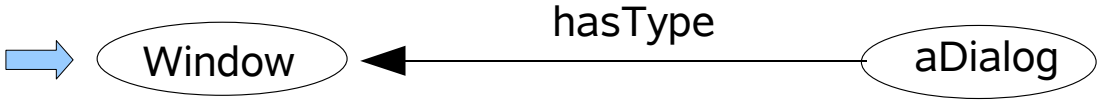
```
public class Window { ... }
public class Dialog extends Window {...}
```




```
public class Window { ... }
public class Container extends Window {...}
public class Dialog extends Container {...}
```




```
public class SomeClass {
   public Window aDialog; ...
}
```



Step 2: Reflecting abstract relations in APIs (2)

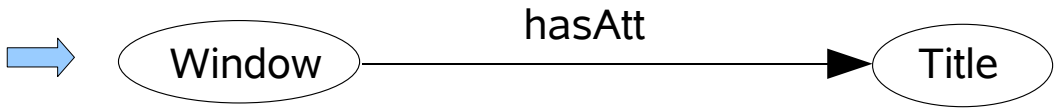
- The relation **"hasProperty"** between a concept and its properties is typically implemented as:
  - the attribute of a class (hasAtt)
  - the accessor of a class (hasAcc)
  - the parameter of a constructor (<hasConstr, hasPar>)
- **Example:**

Window has Title

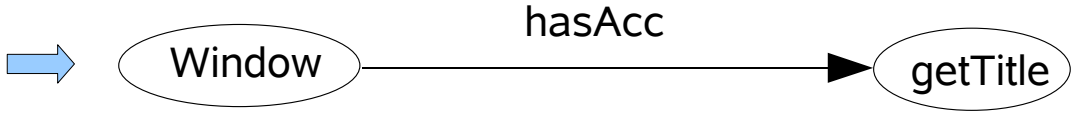
Ontology

APIs

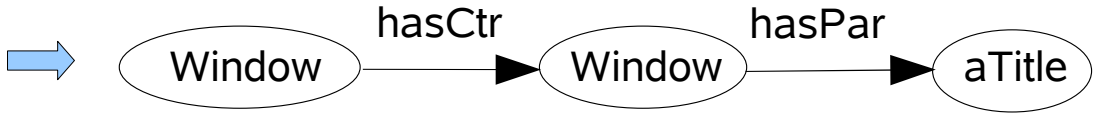
```
public class Window {
  public String title; ...
}
```




```
public class Window {
  public String getTitle() { ... }
}
```




```
public class Window {
  public Window(String aTitle) { ... }
}
```

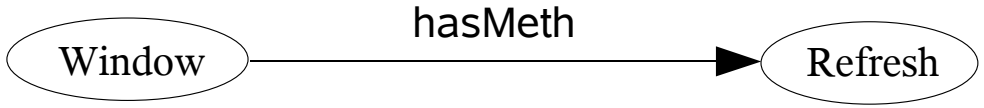


Step 2: Reflecting abstract relations in APIs (3)

- The relation **"isDoer"** between an agent and its actions is typically implem. as
  - the method of a class (hasMethod)
- **Example:**

Window performs a Refresh


```
public class Window {
   public void refresh() { ... }
}
```



- The relation **"actsOn"** between an action and its patients is typically implem. as
  - the parameter of a method
- **Example:**

Draw actsOn Shape


```
public class Graphics {
   public void draw(Shape s) { ... }
}
```

Structural mismatch

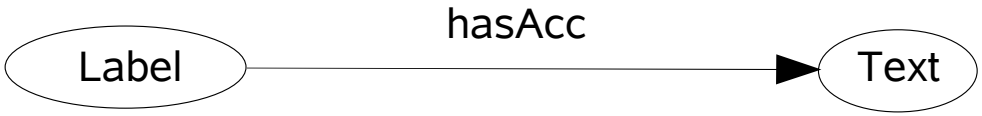
- In the real libraries, due to design decisions many relations are “diffused”:
  - properties of a concept are implemented in the super-class of the concept
  - the taxonomic relation can be implemented with gaps
  - ...

- **Example:**

Label hasProperty Text

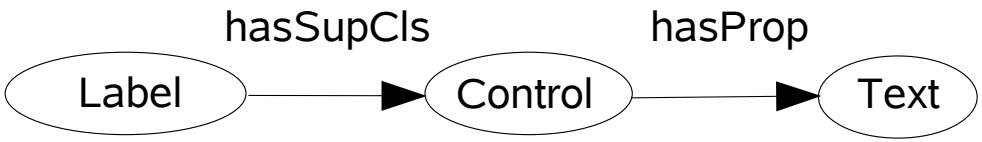
a) Direct implementation

```
package java.awt;
class Label  {
  public getText() { ... }
}
```



b) Generalization

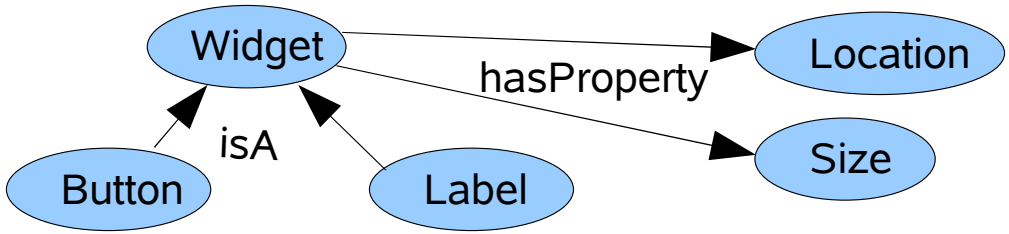
```
namespace  Windows.Forms;
class Control  :  ... {
  public String Text { get; set; }
}
class Label  :  Control  { ... }
```



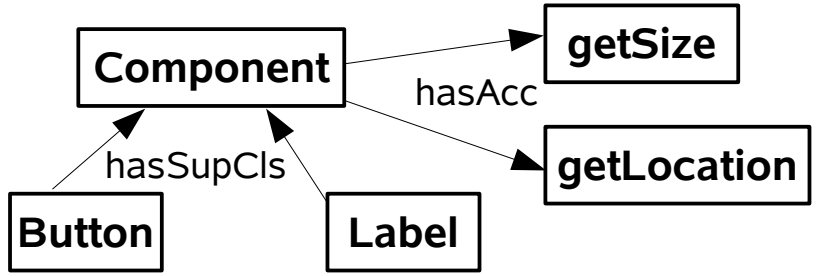
Vocabulary mismatch

- The same domain concept is referenced under different names (synonymy)
  - The structural information can help in identifying synonyms
- **Example:**

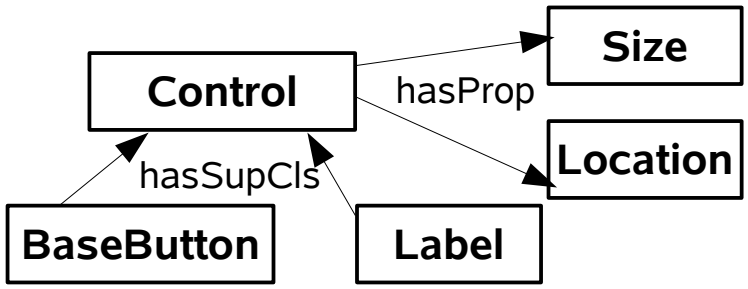
Widget	hasProperty	Location
Widget	hasProperty	Size
Label	isA	Widget
Button	isA	Widget




```
class Component {
  int getSize() { ... }
  int getLocation() { ... }
}
class Label extends Component { ... }
class Button extends Component { ... }
```



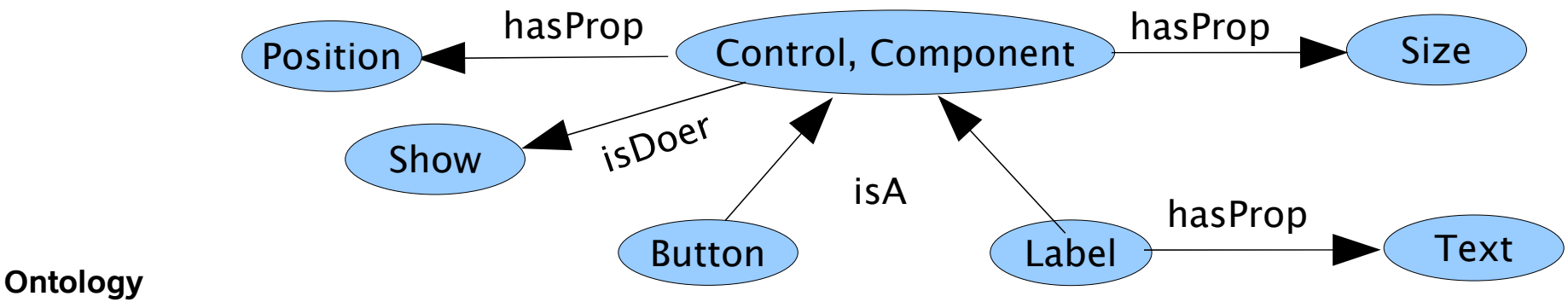

```
class Control ... {
  Point Location { get; set; }
  Size Size { get; set; }
}
class Label  :  Control  { ... }
class BaseButton : Control { ... }
```

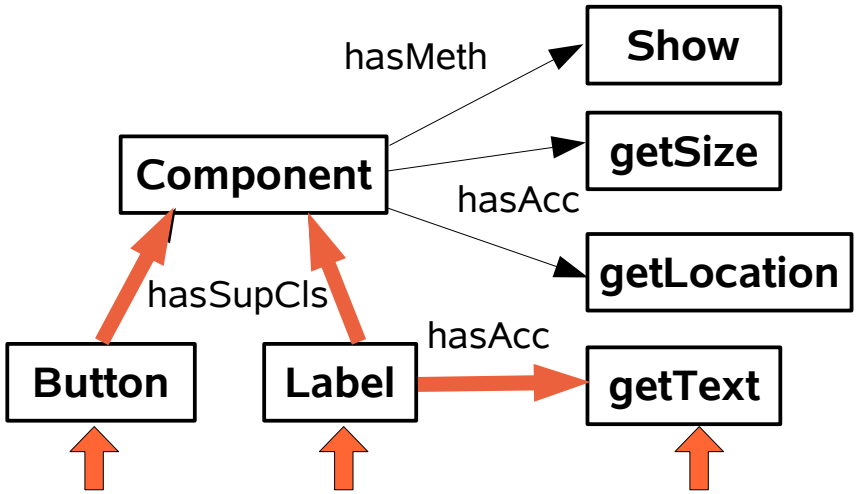


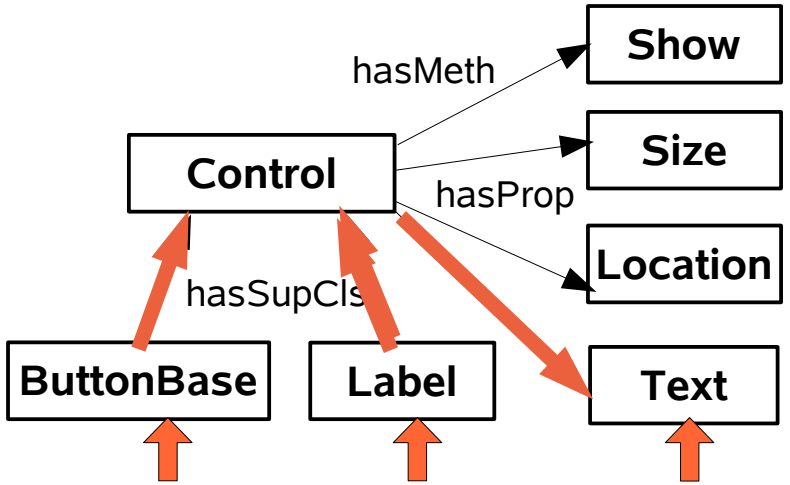
Step 3: Ontology extraction algorithm

- Extract the ontology as the intersection of the API graphs:
  - Matching nodes – based on the similarity of the names
  - Matching edges – according to the defined mapping strategies

At every step we discover a triple from our ontology







Step 4: Noise elimination

- Noise are triples that do not make sense from the point of view of domain know.
  - e.g. similar implementation details, design decisions, “random matches”, etc
    - add -- actsOn -- obj
    - buffer -- isDoer -- clone

- **Noise elimination:** count how many times were the concepts matched and leave out the triples that are found too less times

Case Studies – Research Questions

- **Q1)** Is the overlapping between different APIs big enough for extracting domain ontologies?
- **Q2)** What is the amount of noise in the extracted ontology? What is the effort for eliminating the noise?
- **Q3)** How appropriate is the extracted ontology for locating concepts in code?

Graphical Widget APIs

- **Analyzed APIs:** AWT, Swing, SWT (Java), Windows Forms (.Net)
- **Experimental results:**
  - Vocabulary overlapping

	awt	swing	swt	.NET
awt	-	0.32	0.19	0.31
swing	-	-	0.20	0.38
swt	-	-	-	0.35

  - Structural overlapping

	#Concepts	#Relations	#isA	#hasProperty	#isDoer	#actsOn
GUI	926	2918	203	1625	796	294

Concepts Examples

- Accelerator
- Accessible
- Accessible Context
- Accessible Description
- Action
- Action Command
- Action Listener
- Activate
- Active
- Align
- Alignment
- Alignment X
- Alignment Y
- Button Border
- Alpha
- Angle
- Append
- Applet
- Arc
- Arc Angle
- Arc Height
- Arc Width
- Area
- Ascent
- Attribute
- Back
- Back Color
- Background
- Background Image
- Bar
- Bar Menu
- Base
- Baseline
- Bgcolor
- Block
- Blue
- Border
- Border Style
- Bottom
- Bound
- Box
- Browser
- Button

Triples Examples

- Button | hasProperty | Active
- Button | hasProperty | Alignment
- Button | hasProperty | Background
- Button | hasProperty | Container
- Button | hasProperty | Content
- Button | hasProperty | Enable
- Button | hasProperty | Focus
- Button | hasProperty | Image
- Button | hasProperty | Label
- Button | hasProperty | Margin
- Button | hasProperty | Minimum Size
- Button | hasProperty | Mnemonic
- Button | hasProperty | Name
- Button | hasProperty | Parent
- Button | hasProperty | Style
- Button | hasProperty | Text
- Button | isA | Component
- Button | isA | Container
- Button | isA | Control
- Button | isA | Item
- Button | isA | Widget
- Button | isDoer | Check
- Button | isDoer | Click
- Button | isDoer | Focus
- Button | isDoer | Lost Focus
- Button | isDoer | Mouse Drag
- Button | isDoer | Mouse Move
- Button | isDoer | Notify

Triples Examples (2)

- Font | hasProperty | Bound
- Font | hasProperty | Family
- Font | hasProperty | Handle
- Font | hasProperty | Name
- Font | hasProperty | Size
- Font | hasProperty | Style
- Font | isA | Attribute
- Font | isA | Resource
- Font | isA | Text

Effort estimation

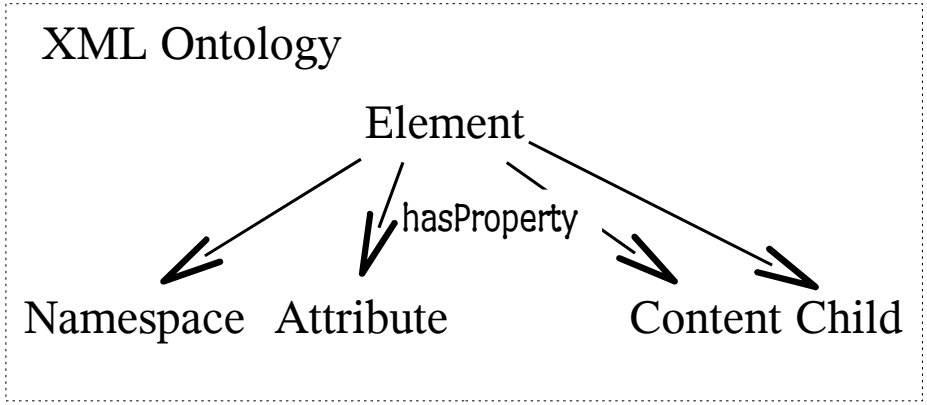
- **Noise:** about 50% of the triples are valid
  - Taking into consideration the frequency of triples increase the precision but decrease the recall

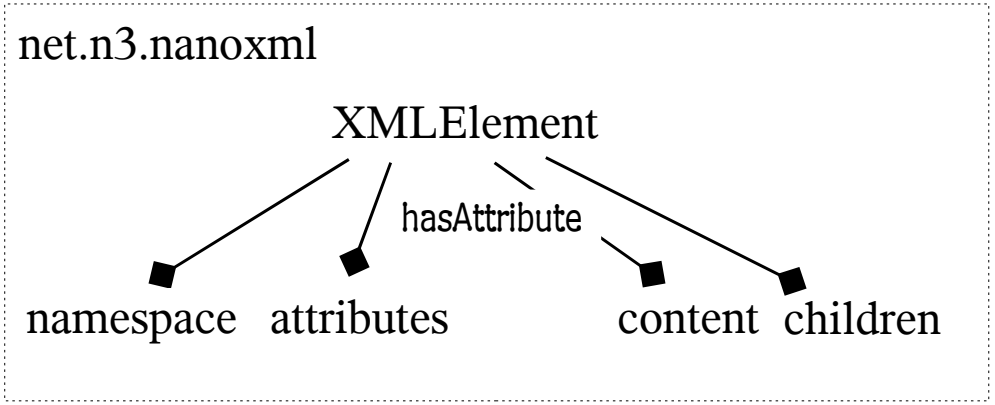
	Hours	Auto
Selection of APIs	2	No
Preparation of APIs	3	No
Running the algorithm	3	Yes
Heuris. Noise Elim.	0.5	Yes
Manual Noise Elim.	1	No

Concepts location

- We mapped the extracted ontologies on JHotDraw
  - We identified 179 concepts that were assigned to 1370 program elements

- **Example:**





Open Problems

- A program relation can be interpreted through more real-world relations
  - An attribute can denote either a property or a part
- Increase the weight of the ontology
  - More relations, constraints
- Evaluation of the quality of the extracted ontology
  - Normalization, completeness
- Evolution of the extracted ontology
  - What happens when we analyze yet another API?
- Building a repository of ontologies
  - The GUI and XML are only the start

Thank You!

- Button | hasProperty | Active
- Button | hasProperty | Alignment
- Button | hasProperty | Background
- Button | hasProperty | Minimum Size
- Button | hasProperty | Mnemonic
- Button | hasProperty | Name
- Button | hasProperty | Parent
- Button | hasProperty | Prefer Size
- Button | hasProperty | Selection
- Button | hasProperty | Style
- Button | hasProperty | Text
- Button | isA | Component
- Button | isDoer | Lost Focus
- Button | isDoer | Mouse Drag
- Button | isDoer | Mouse Move
- Button | isDoer | Notify
- Button | isDoer | Press
- Button | isDoer | Remove

If you are interested, you can get involved as contributor or as user

http://www4.in.tum.de/~ratiu/knowledge_repository.html