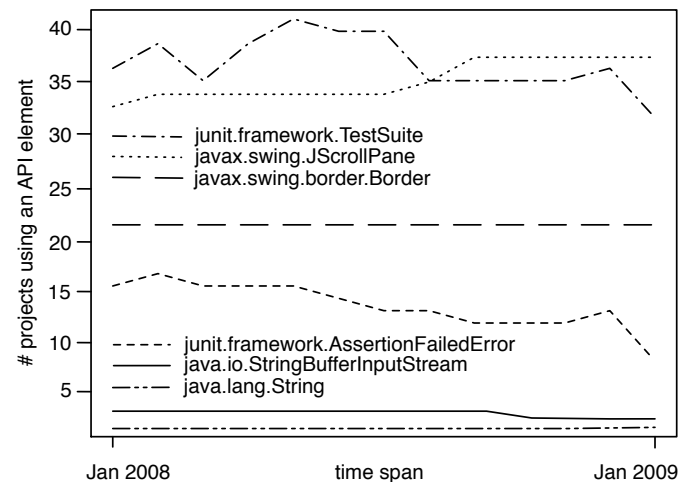


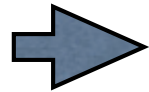
Mining API Popularity



Yana Mileva, Valentin Dallmeier, Andres Zeller
Saarland University, *Germany*

Mining **API** Popularity

API



java.io.StringBufferInputStream

java.io.StringReader

java.io.PrintStream

...

Mining API **Popularity**



popular/used



unpopular/unused

Mining API Popularity



```
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}
```

Mining API Popularity

- API *quality*
- API *usability*
- API *compatibility*
- ...

How was it **till now**?

bug-tracking systems

- API **quality** documentation
- API **usability**
- API **compatibility**

forums

- ...

emails and groups

Mining API Popularity



200 Projects

```
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}
```



Jan'08 Jan'09

Mining API Popularity

```
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

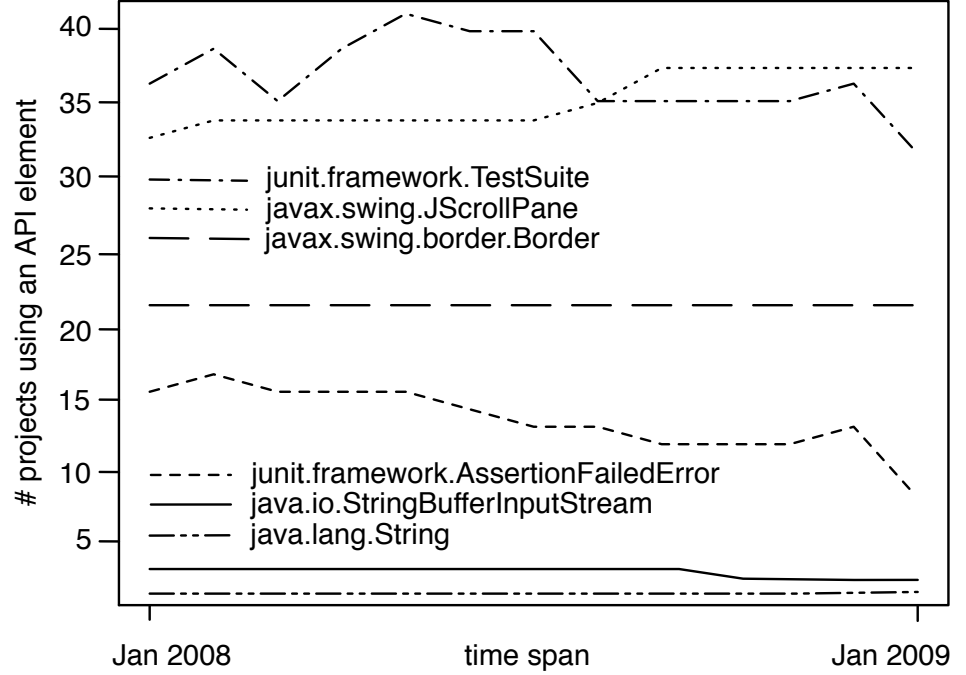
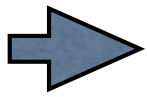
        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}

import java.util.Iterator;
import java.util.List;
import java.util.Set;

public class BcelTypeMunger extends ConcreteTypeMunger {
    private boolean mungeNewParent(BcelClassWeaver weaver)
    {
        LazyClassGen target = weaver.getLazyClassGen();
        boolean cont = true;
        cont = enforce_abstractMethods(weaver);

        List methods = newParent.getWithoutIter(true);
        for (int i = 0; i < method.size(); i++) {
            ResolvedMember superMethod = iter.next();
        }
    }
}
```

200 Projects



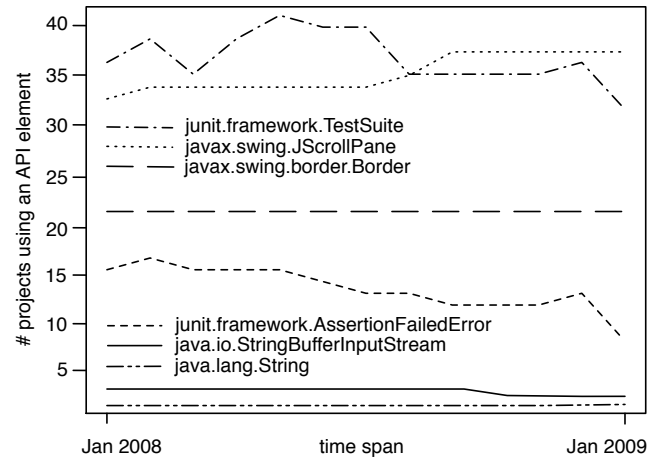
Jan'08 ————— Jan'09

usage trends

Who would benefit?

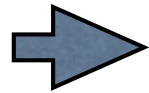


API users



API producers

API users

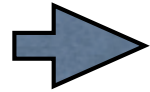


always want to use **the best product**

want **free of defects** software

don't want to **waste time**

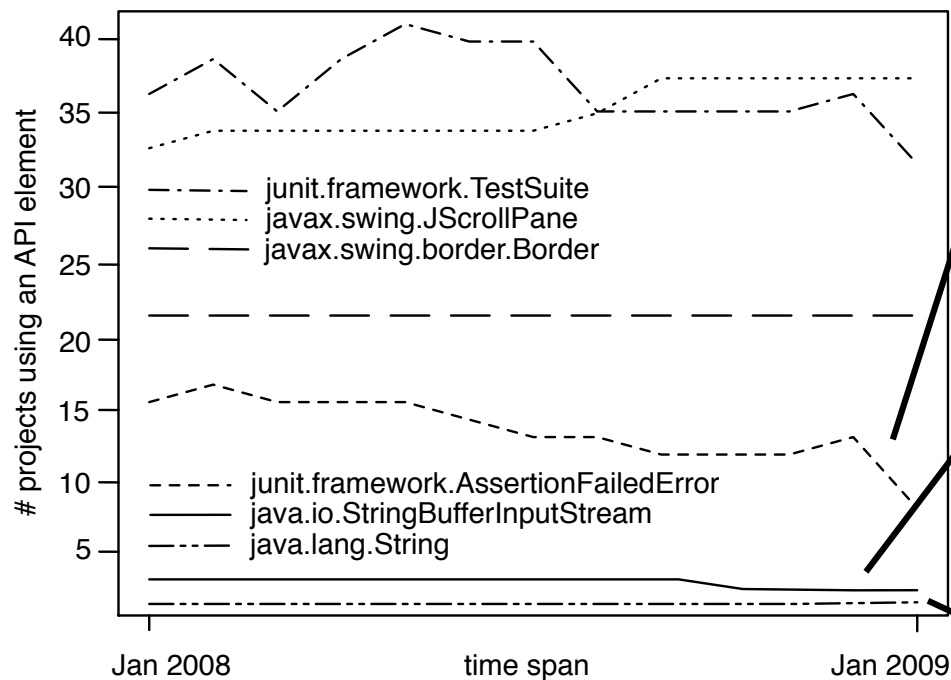
API producers



want to produce **good product**

want **feedback** from the users

Examples



junit.framework.AssertionFailedError
compatibility problems

java.io.StringBufferInputStream
code defect

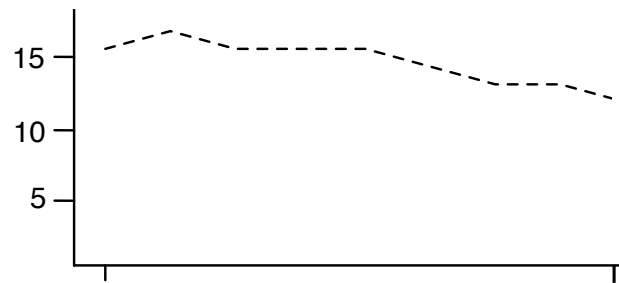
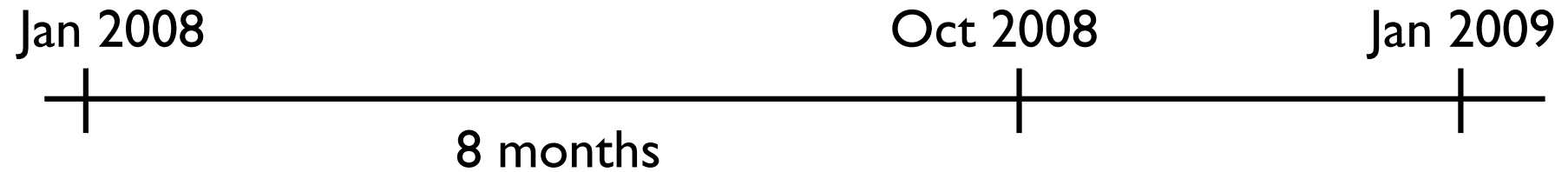
java.lang.String - code smell

Hypothesis

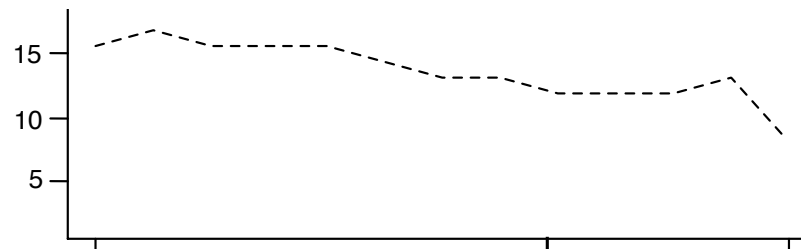
The past usage trend of an API element is predictive of the future usage trend of the same element.

Evaluation Scenario



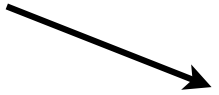


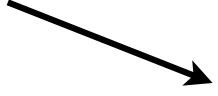
junit.framework.AssertionFailureError



13 months



Evaluation Results

Trends			
	67%	33%	0%
	1%	98%	1%
	0%	18%	82%

Restrictions

- The approach might cancel itself? **No.**
- Unused *imports* remain in code? **No.**
- Works only for Java? **No.**

Future Work

- Find reasons behind a decline in usage.
- Target the recommendations better.
- Analyze more data.
- Make user surveys.

How was it **till now?**

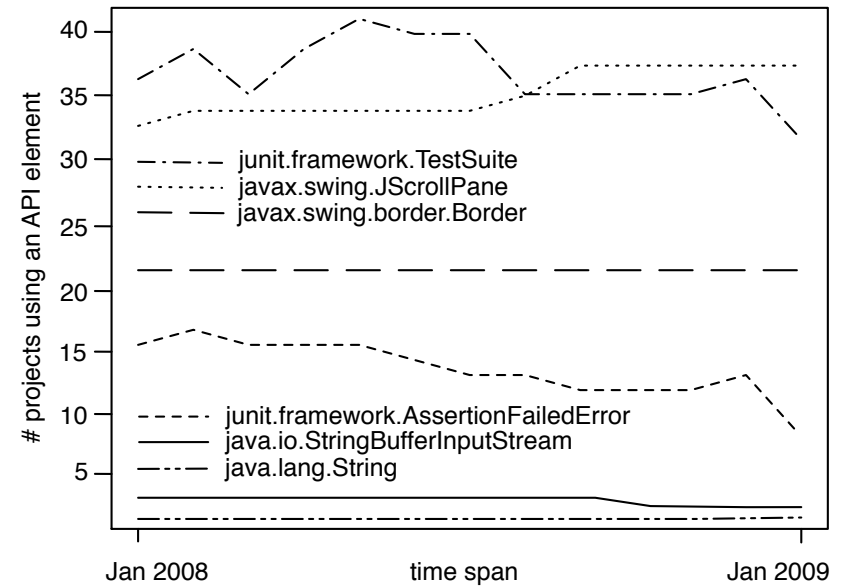
bug-tracking systems

- API **quality** documentation
- API **usability**
- API **compatibility**

forums

- ...

emails and groups



Evaluation **Results**

Trends	↗	→	↘
↗	67%	33%	0%
→	1%	98%	1%
↘	0%	18%	82%

Future Work

- **Find reasons** behind a decline in usage.
- **Target** the recommendations better.
- Analyze **more data**.
- Make **user surveys**.