

It's not you; it's the API!

Automatically avoiding API misuses

Sarah Nadi

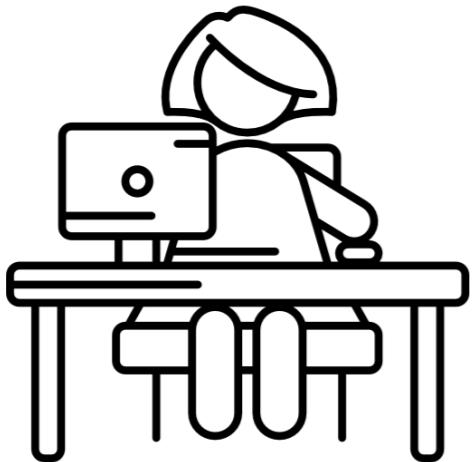


@sarahnadi



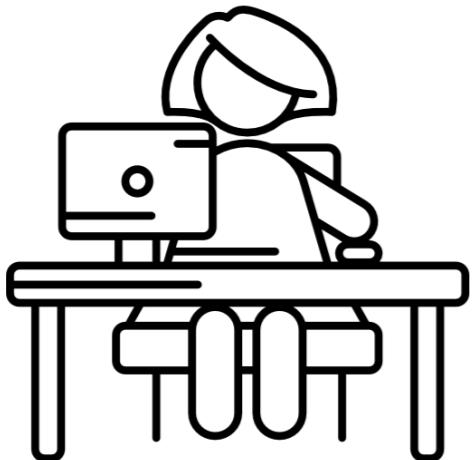
sarahnadi.org

It will Never Work in Theory — StrangeLoop 2022



Wants to define OpenAPI definitions

```
@OpenAPIDefinition(  
    info = @Info(  
        title = "Custom API title",  
        version = "3.14"  
    )  
)  
@Path("/hello")  
public class ExampleResource {  
  
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    public String hello() {  
        return "hello";  
    }  
}
```



Wants to define OpenAPI definitions

```
@OpenAPIDefinition(  
    info = @Info(  
        title = "Custom API title",  
        version = "3.14"  
    )  
)  
@Path("/hello")  
public class ExampleResource {  
  
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    public String hello() {  
        return "hello";  
    }  
}
```

No change reflected
in the open API
spec!



```
@OpenAPIDefinition(  
    info = @Info(  
        title = "Custom API title",  
        version = "3.14"  
    )  
)  
@Path("/hello")  
public class ExampleResource {  
  
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    public String hello() {  
        return "hello";  
    }  
}
```



```
@OpenAPIDefinition(  
    info = @Info(  
        title = "Custom API title",  
        version = "3.14"  
    )  
)  
@Path("/hello")  
public class ExampleResource  
    extends Application{  
  
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    public String hello() {  
        return "hello";  
    }  
}
```

"Try putting the annotation on the JAX-RS Application class"

Try putting the annotation on the JAX-RS Application class. I realize you don't need one of those in a Quarkus application, but I think it doesn't hurt either. For reference in the specification TCK:

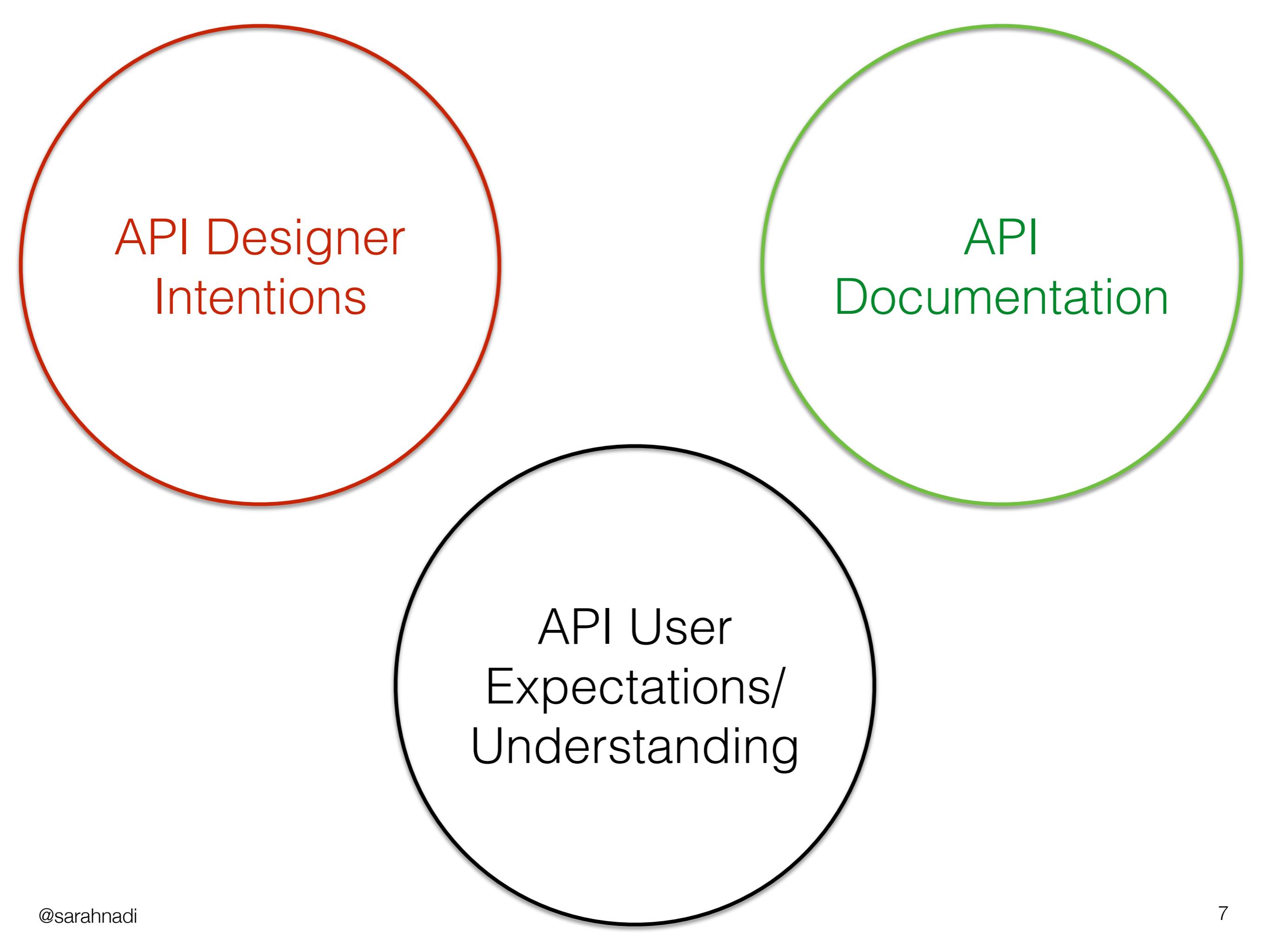
2

<https://github.com/eclipse/microprofile-open-api/blob/master/tck/src/main/java/org/eclipse/microprofile/openapi/apps/airlines/JAXRSApp.java>

Share Improve this answer Follow

answered Dec 4, 2019 at 13:22

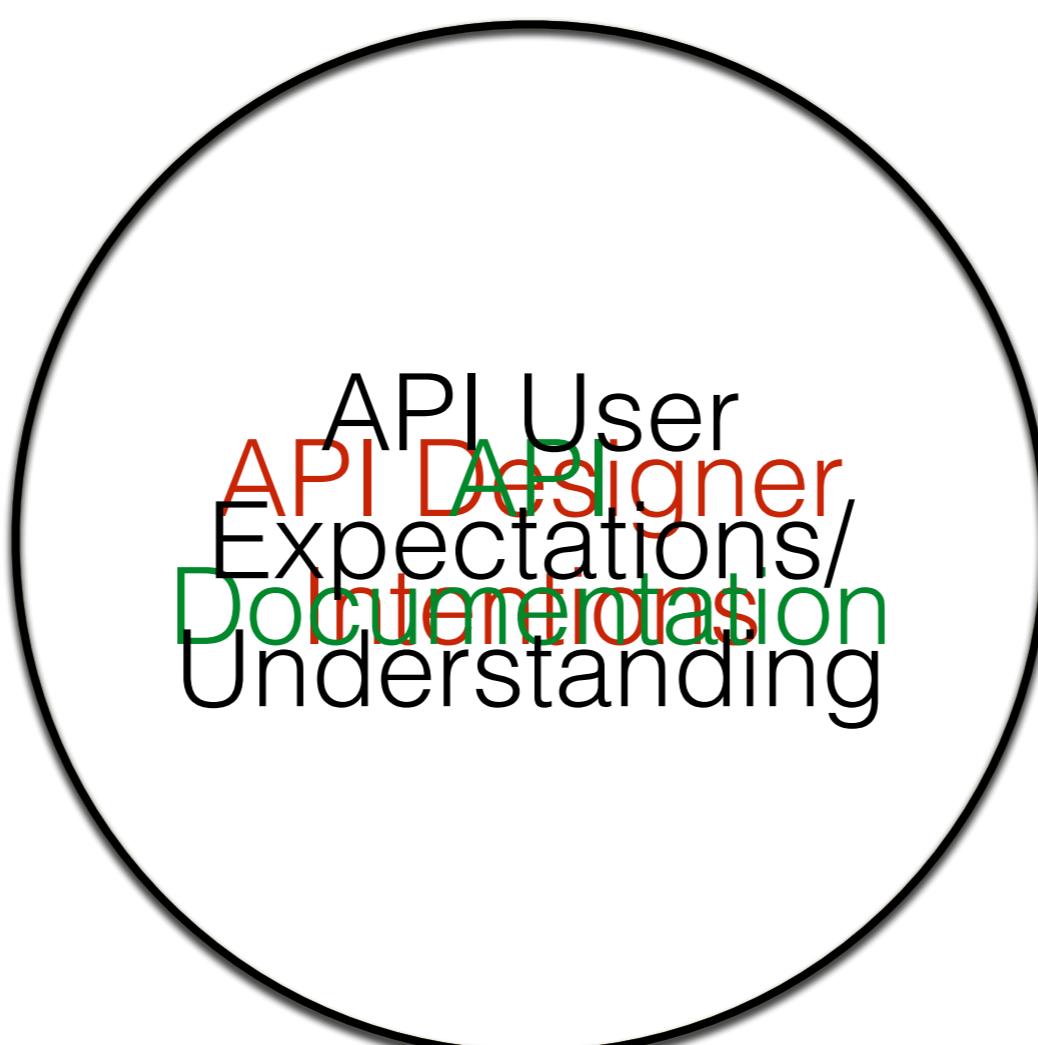
-
- 1 Worked Unfortunately just not well documented it seems. Should also be noted that you can put the same information in `src/main/resources/META-INF/openapi.yml` and the information there will be merged with the information gleaned from annotations. (meaning if you just put the `info` part in the yml, the paths will be generated for you and integrated together) – Dec 4, 2019 at 22:15



API Designer
Intentions

API
Documentation

API User
Expectations/
Understanding

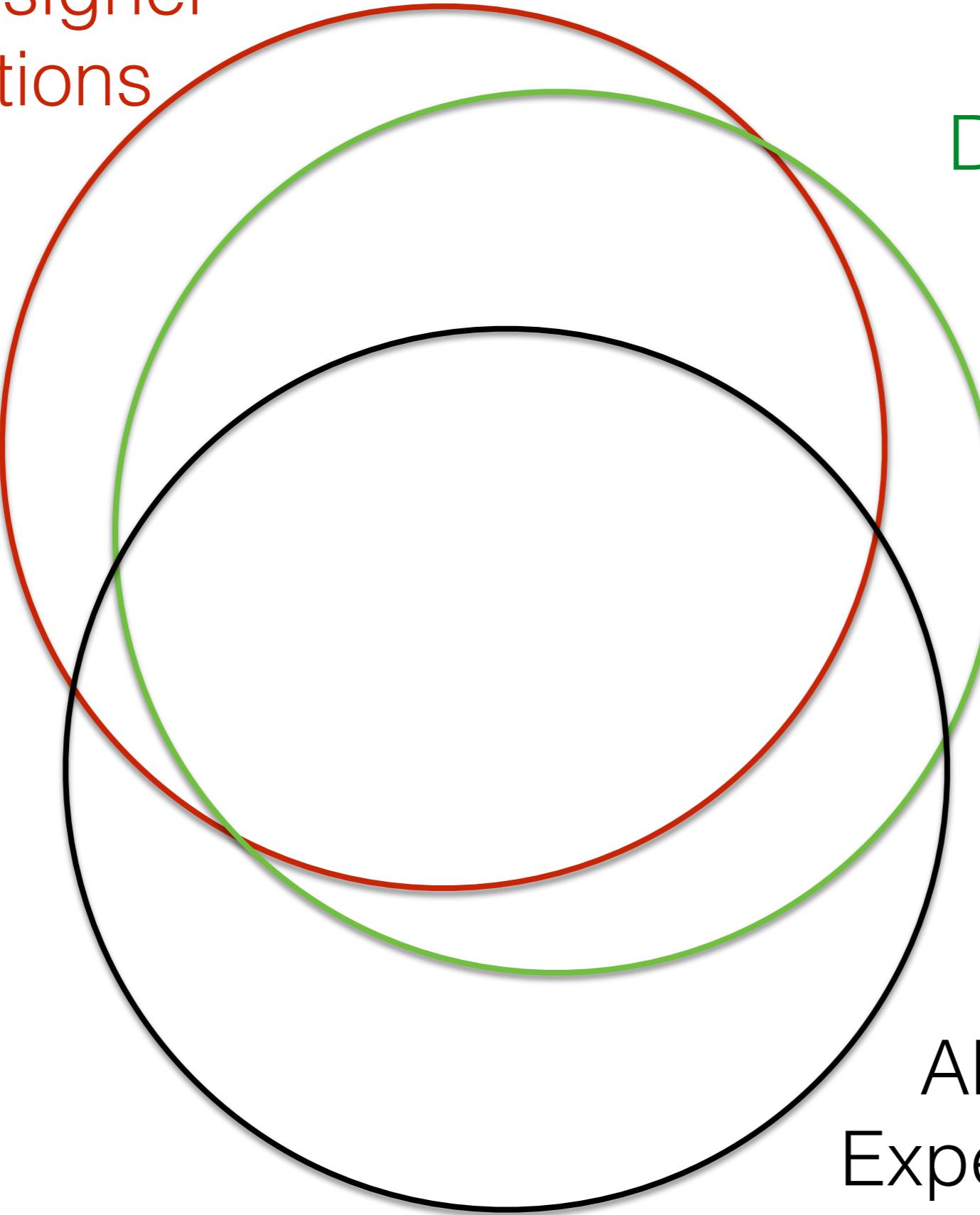


API User
Expectations/
Understanding

API Designer
Documentation

API Designer
Intentions

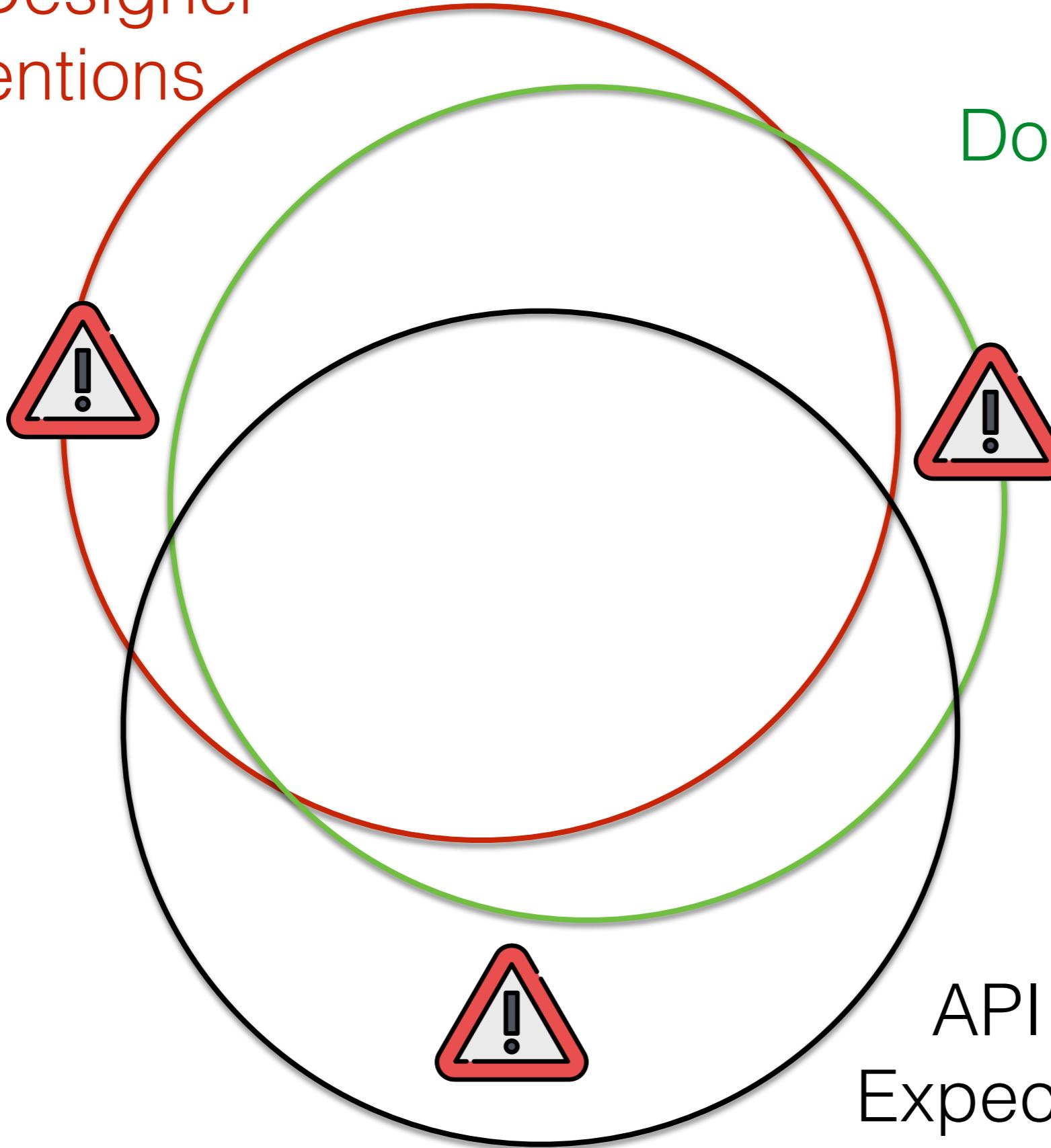
API
Documentation



API User
Expectations/
Understanding

API Designer
Intentions

API
Documentation

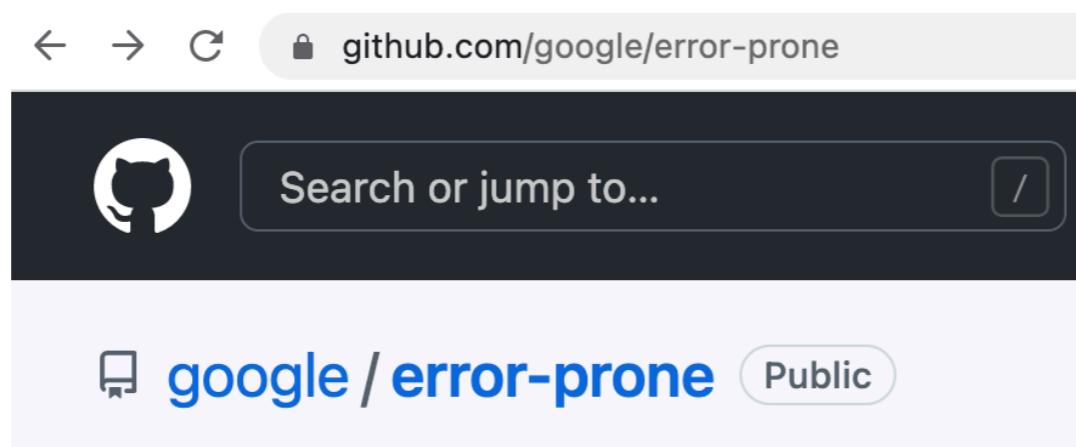


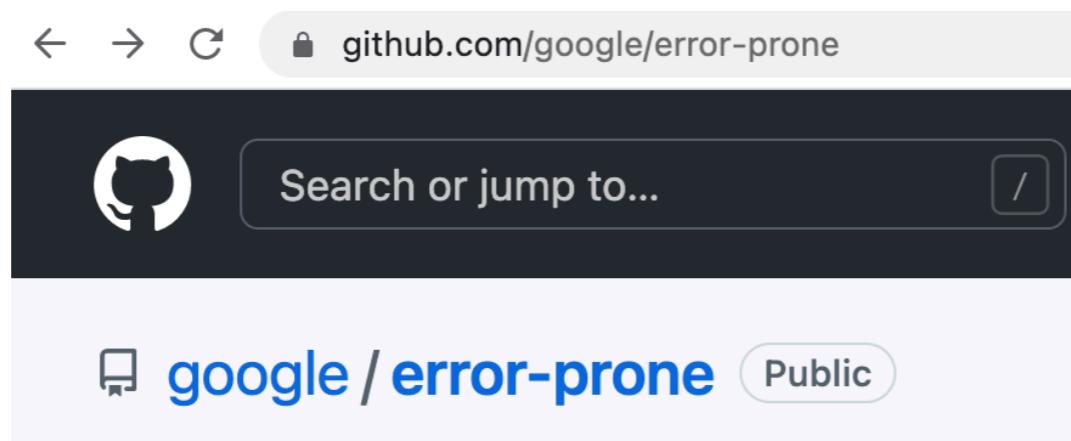
API User
Expectations/
Understanding



API Misuses

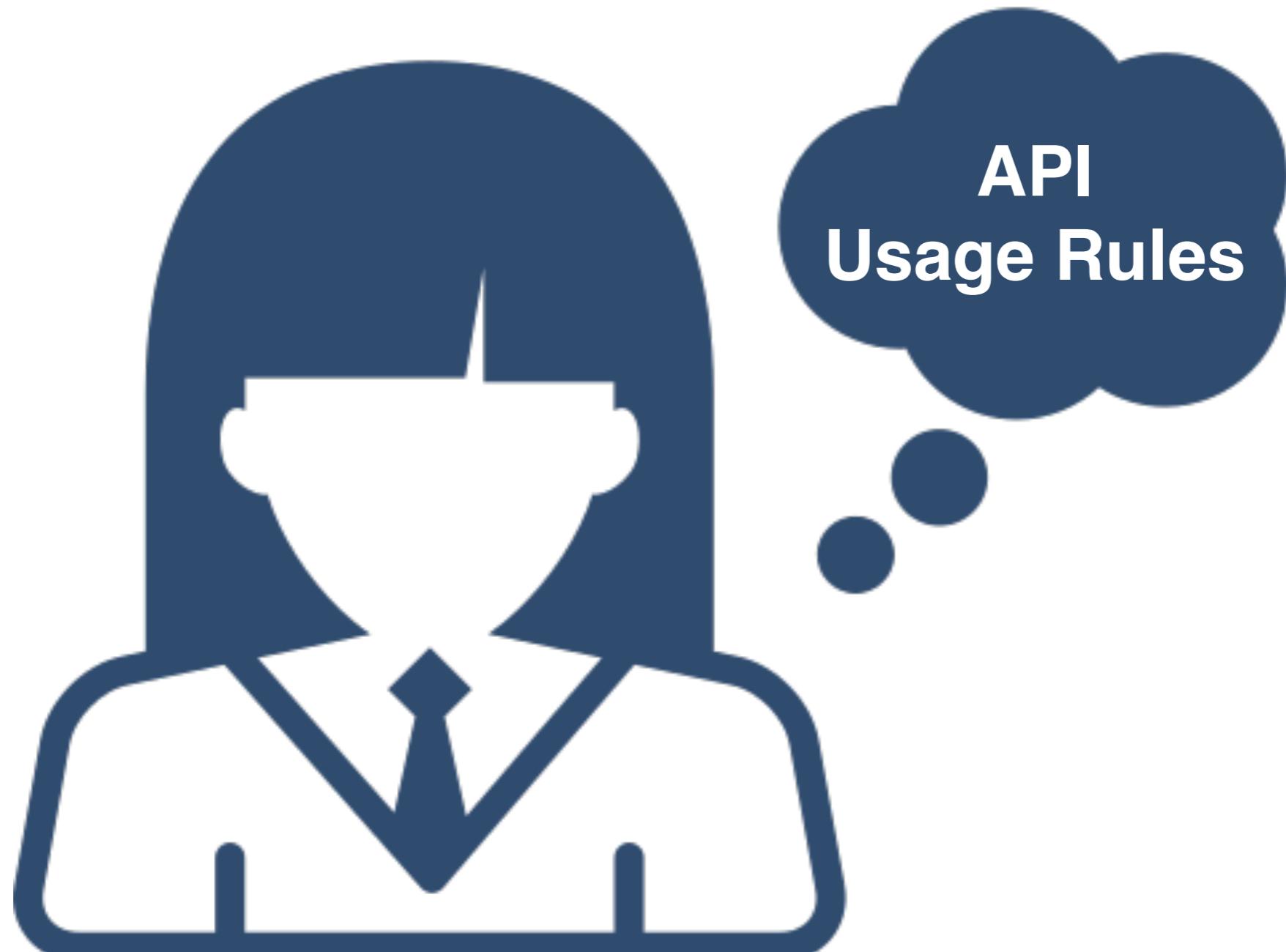
Incorrect or unexpected behavior because of not properly using the API





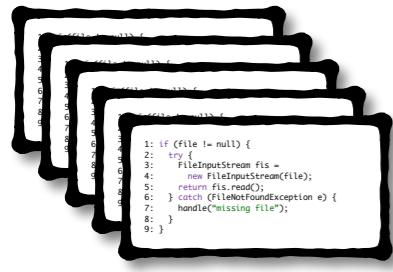
Common mistakes/bug patterns

Common mistakes/bug patterns



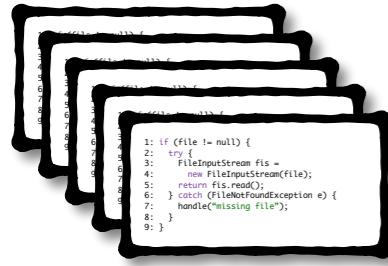
API Designer/Expert

Can we discover these
API usage rules
automatically?



Client Code

```
1: if (file != null) {  
2:     try {  
3:         FileInputStream fis =  
4:             new FileInputStream(file);  
5:         return fis.read();  
6:     } catch (FileNotFoundException e) {  
7:         handle("missing file");  
8:     }  
9: }
```



Client
Code

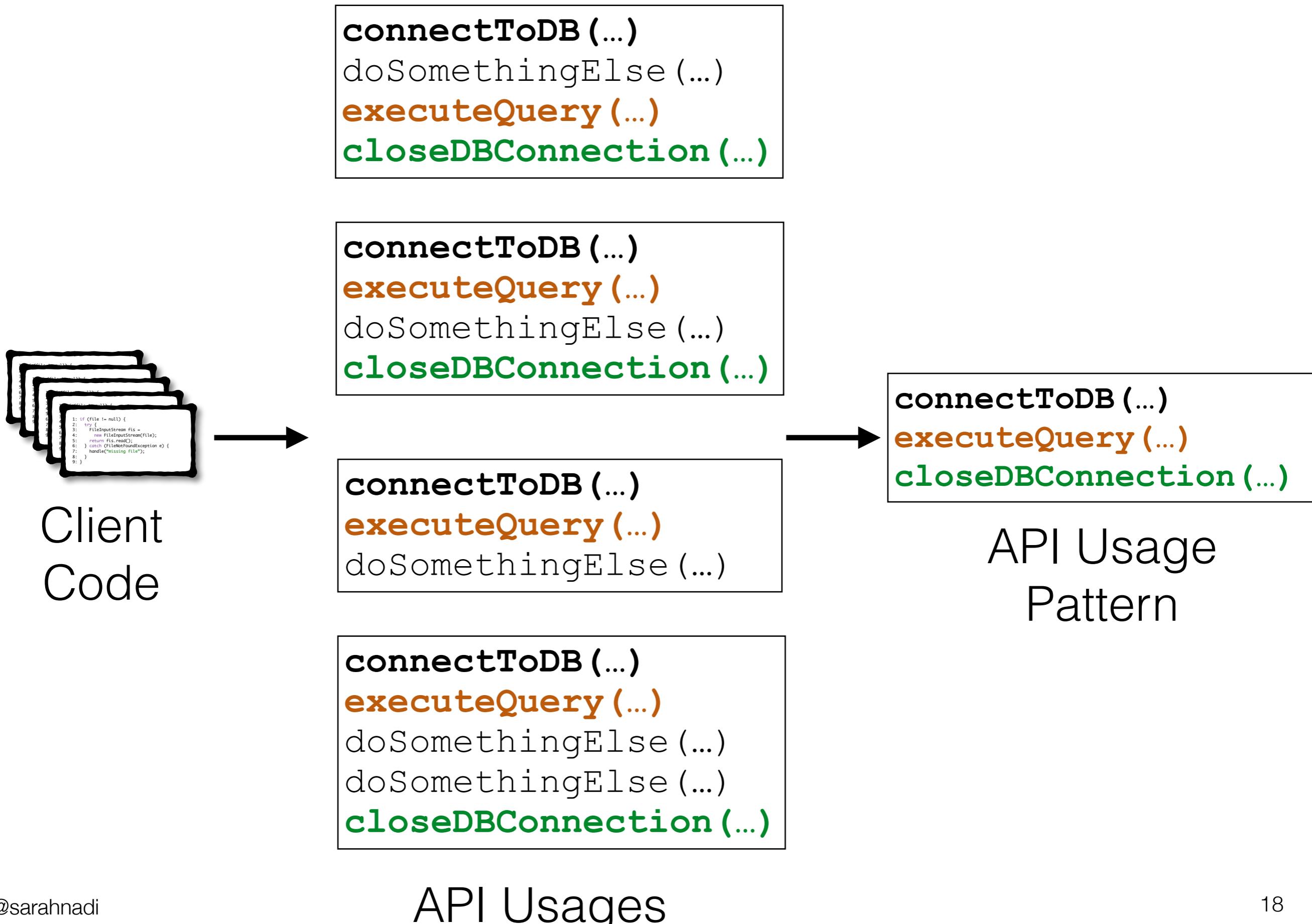


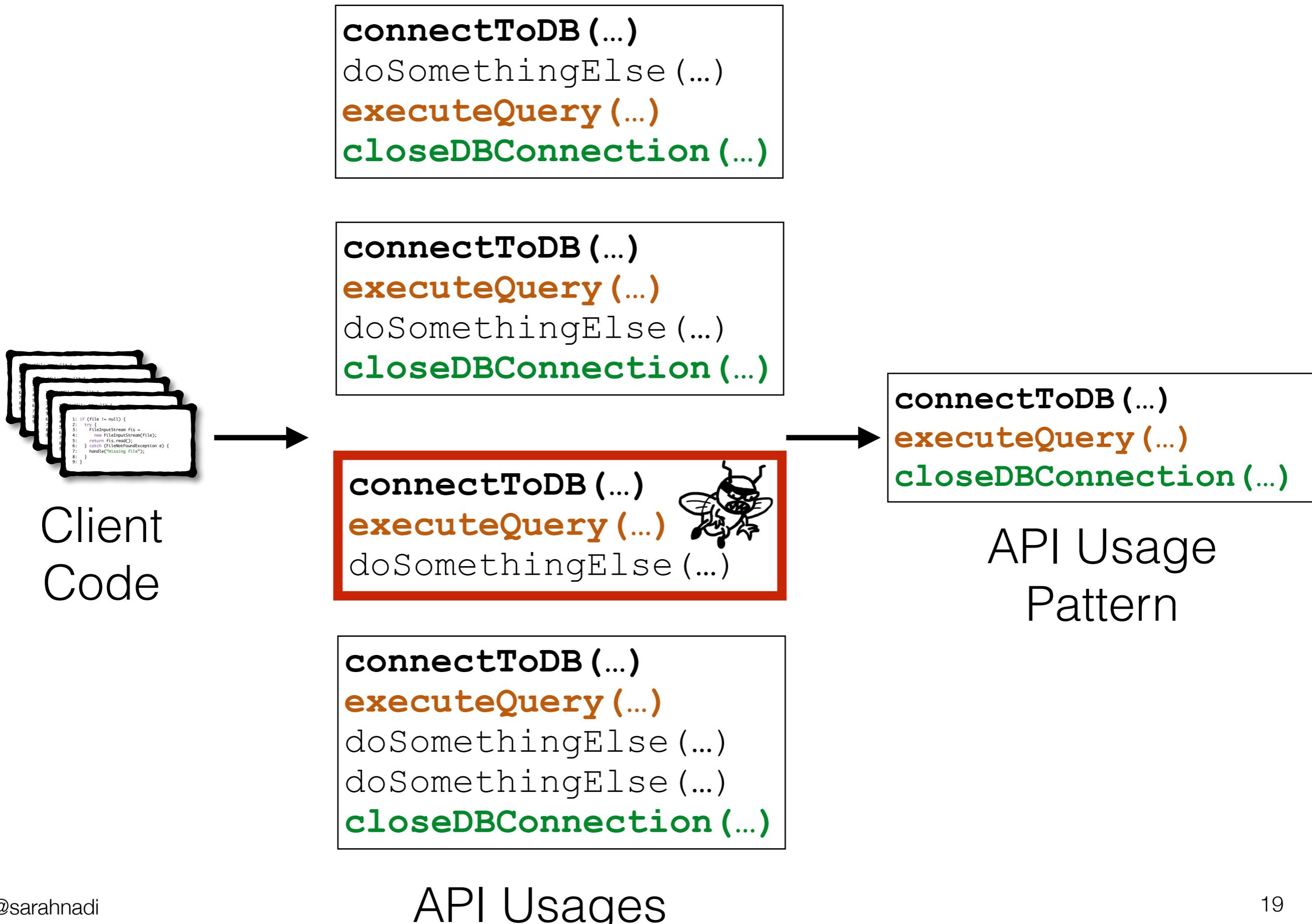
```
connectToDB (...)  
doSomethingElse (...)  
executeQuery (...)  
closeDBConnection (...)
```

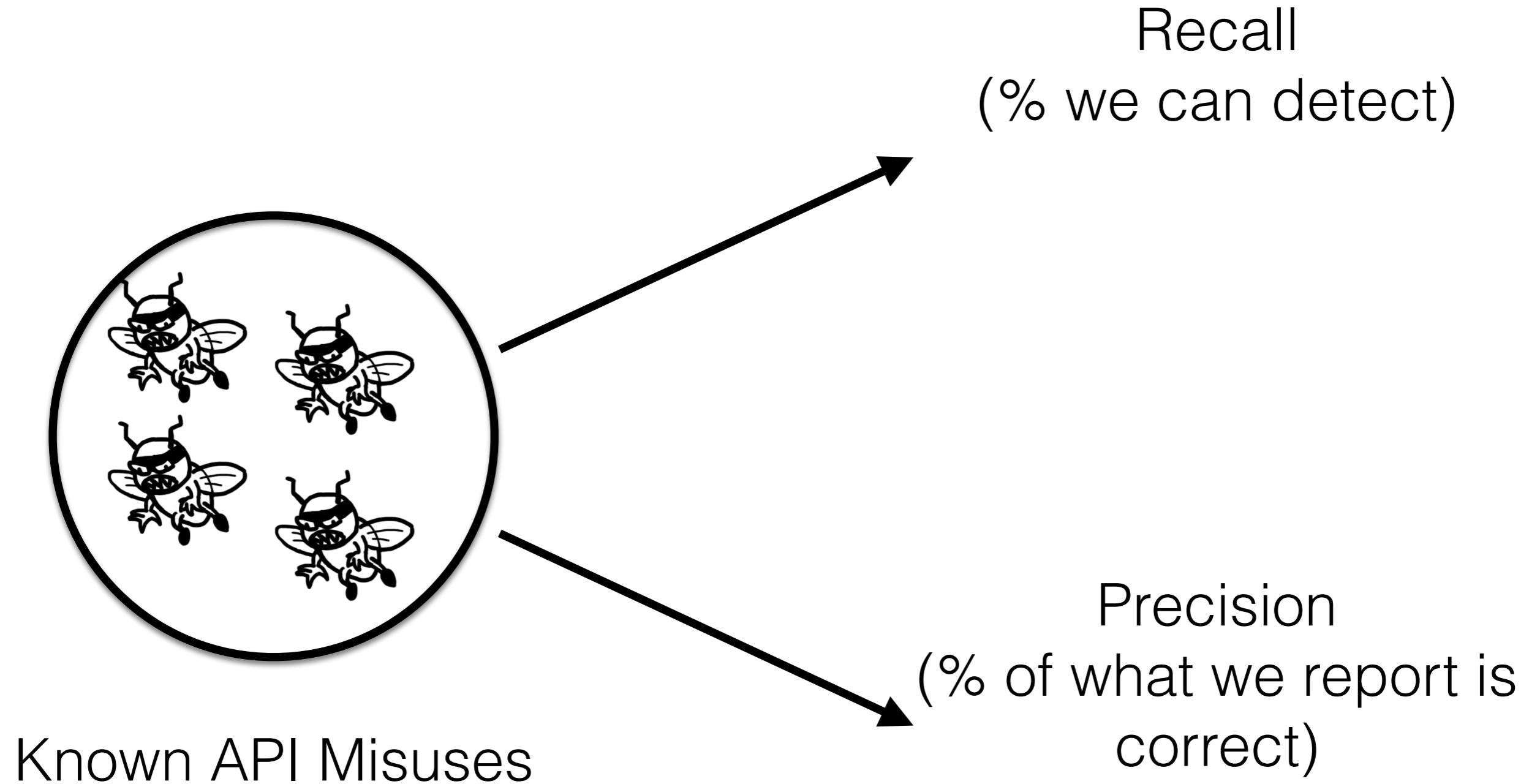
```
connectToDB (...)  
executeQuery (...)  
doSomethingElse (...)  
closeDBConnection (...)
```

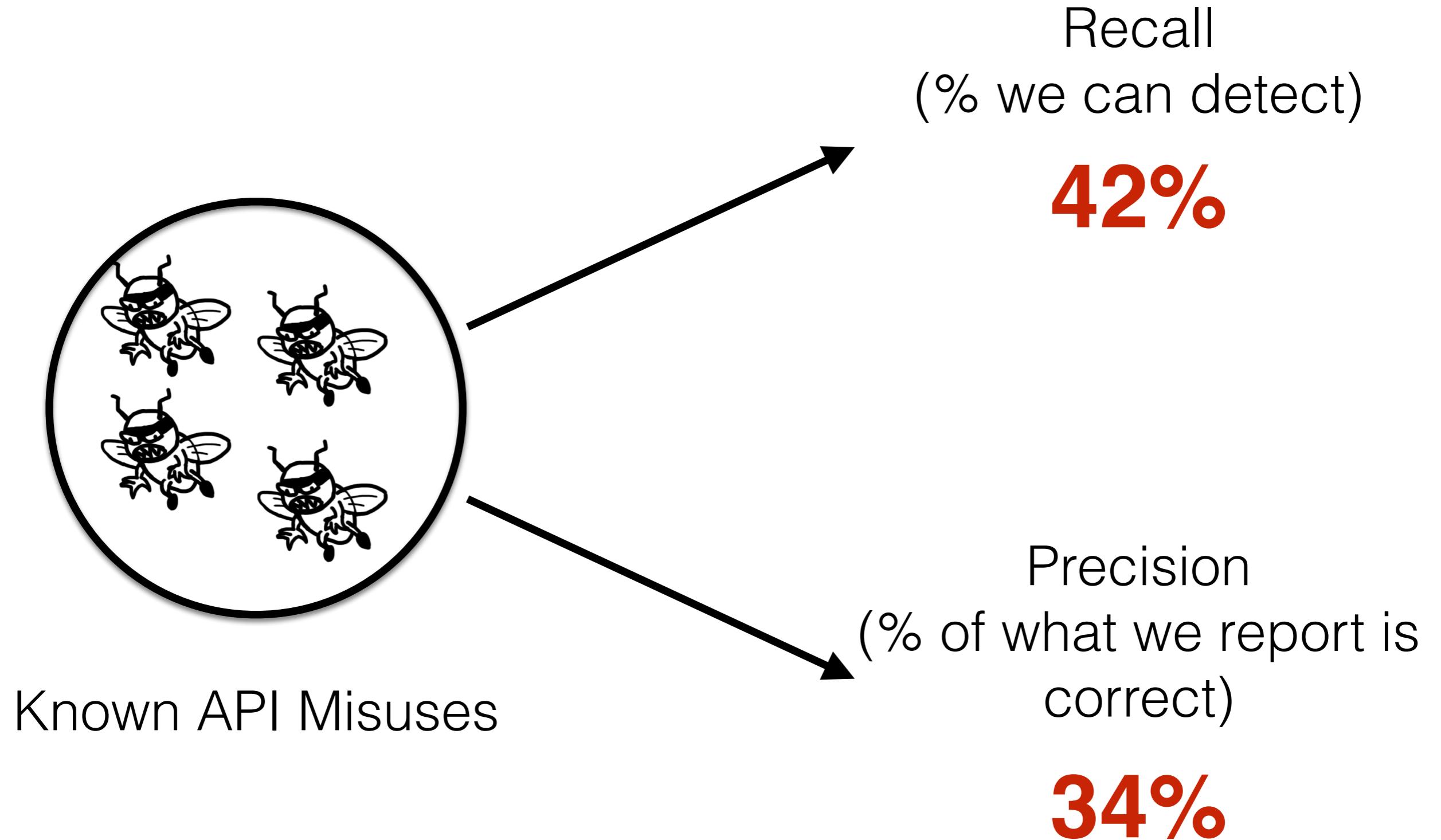
```
connectToDB (...)  
executeQuery (...)  
doSomethingElse (...)
```

```
connectToDB (...)  
executeQuery (...)  
doSomethingElse (...)  
doSomethingElse (...)  
closeDBConnection (...)
```









Pattern Mining

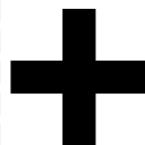
- + Automated and can mine patterns with no human intervention
- In real world setting, suffers from **low precision and recall**

Writing API Usage Rules

- + Rules are going to be accurate
- Takes too much human effort

Pattern Mining

- + Automated and can mine patterns with no human intervention
- In real world setting, suffers from **low precision and recall**

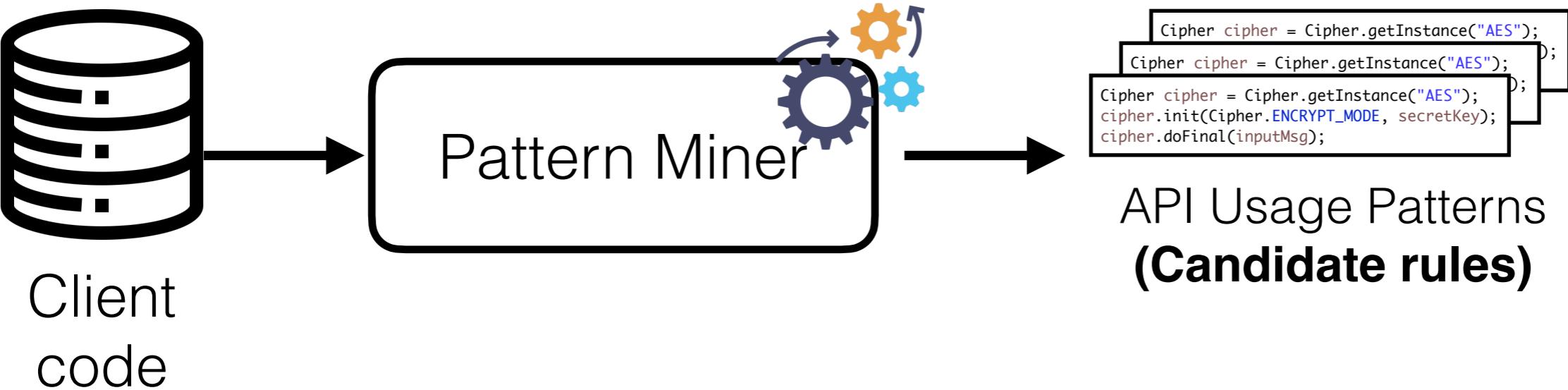


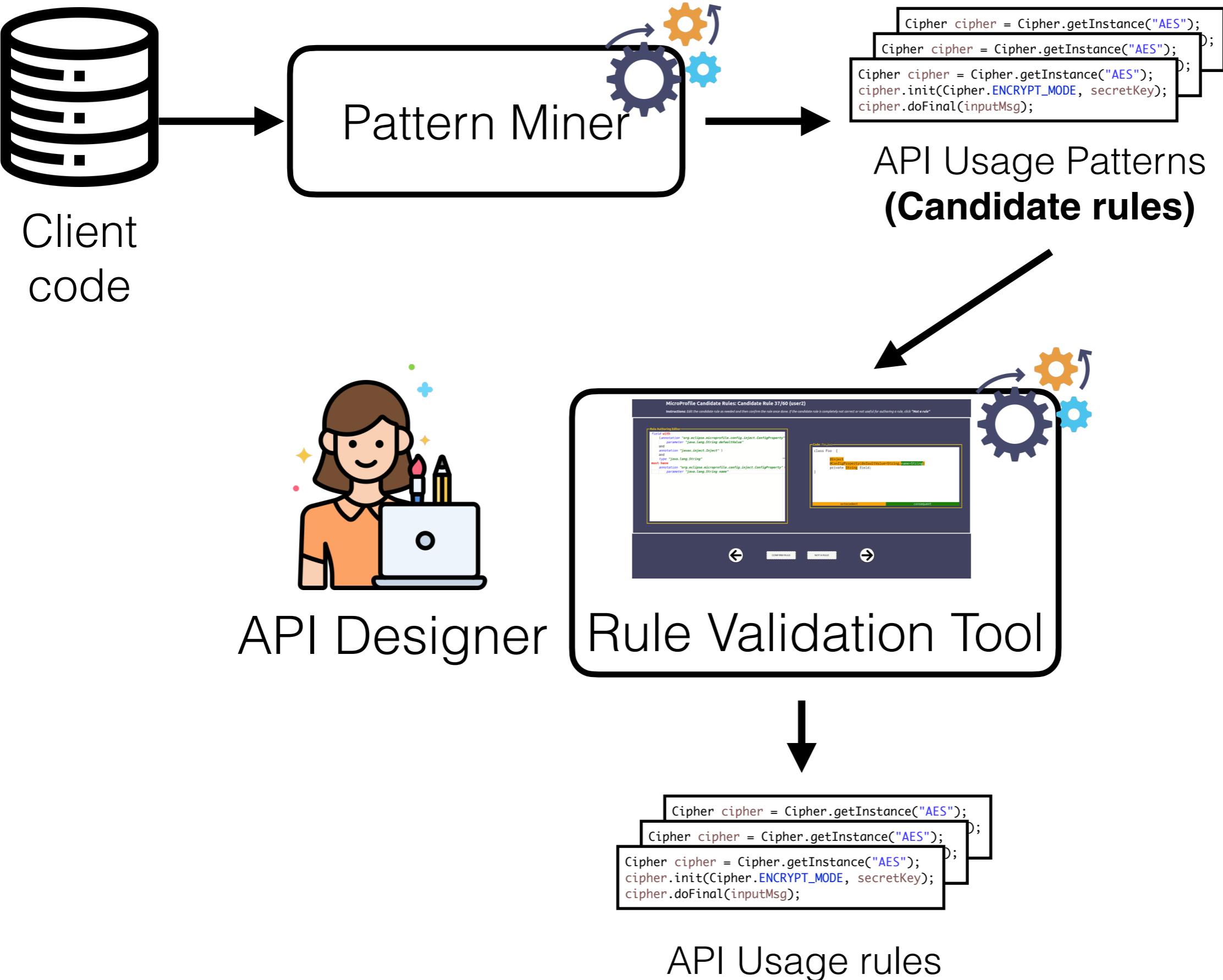
Writing API Usage Rules

- + Rules are going to be accurate
- Takes too much human effort

Make it easier for the API designer to make expectations explicit/checkable

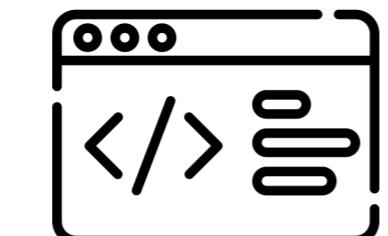
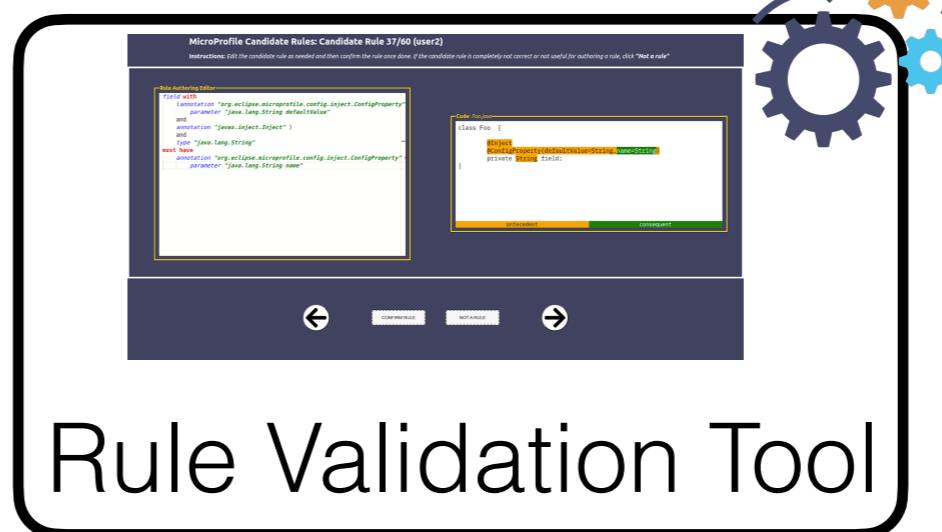
Make it easier for API users to know when they have made mistakes







API Designer Rule Validation Tool



Dev code

```
Cipher cipher = Cipher.getInstance("AES");  
Cipher cipher = Cipher.getInstance("AES");  
Cipher cipher = Cipher.getInstance("AES");  
cipher.init(Cipher.ENCRYPT_MODE, secretKey);  
cipher.doFinal(inputMsg);
```

API Usage rules

Code Checker





API Designer

Rule Validation Tool + localhost/home ?

Candidate Rule 1/67 : 67 rules left to label (mansur)

Instructions: Edit the candidate rule as needed and then confirm the rule once done. If the candidate rule is completely not correct or not useful for authoring a rule, click "Not a rule"

Rule Authoring Editor

```
field with
  (annotation "org.eclipse.microprofile.config.inject.ConfigProperty" wi
    parameter "java.lang.String defaultValue"
  and
  annotation "javax.inject.Inject" )
  and
  type "java.lang.String"
must have
  annotation "org.eclipse.microprofile.config.inject.ConfigProperty" wit
    parameter "java.lang.String name"
```

Code Preview: Foo.java

```
class Foo {
  @Inject
  @ConfigProperty(defaultValue=String, name=String)
  private String field;
}
```

anteecedent consequent



CONFIRM RULE

BEST PRACTICE

NOT A RULE





API Designer

Rule Validation Tool + localhost/home ?

Candidate Rule 1/67 : 67 rules left to label (mansur)

Instructions: Edit the candidate rule as needed and then confirm the rule once done. If the candidate rule is completely not correct or not useful for authoring a rule, click "Not a rule"

Rule Authoring Editor

```
field with
  (annotation "org.eclipse.microprofile.config.inject.ConfigProperty" wi
    parameter "java.lang.String defaultValue"
  and
  annotation "javax.inject.Inject" )
  and
  type "java.lang.String"
must have
  annotation "org.eclipse.microprofile.config.inject.ConfigProperty" wit
    parameter "java.lang.String name"
```

Code Preview: Foo.java

```
class Foo {
  @Inject
  @ConfigProperty(defaultValue=String, name=String)
  private String field;
}
```

anteecedent consequent



CONFIRM RULE

BEST PRACTICE

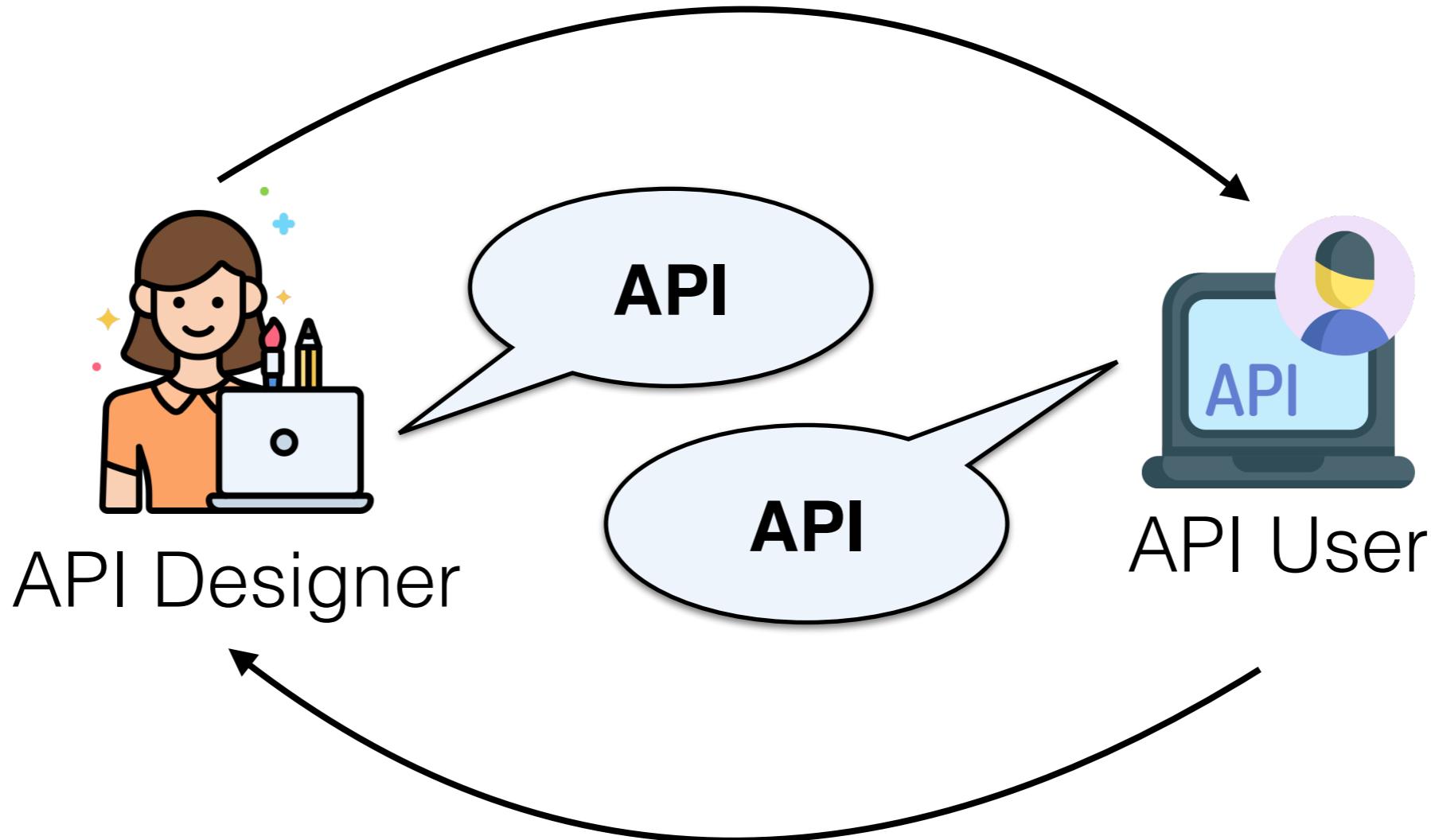
NOT A RULE





API User

```
$ mvn violation-detector: scan
[WARN] For rule: QueryGraphQLAPIRule
[WARN]   class with function with annotation "Query" \
[WARN]     must have annotation "GraphQLApi"
[WARN] Class FooBar is missing the following element(s):
[WARN]   [@org.eclipse.microprofile.graphql.GraphQLApi]
[WARN] Location: (line 22, col 1) - (line 77, col 1)
```



Avoid buggy software by having an
(automated) conversation between
API designers & API users



Mansur Gulami



Batyr Nuryyev



Ajay Kumar Jha



Karim Ali



Sven Amann



Mira Mezini



Hoan Nguyen



Hoan Nguyen



Yee-kang Chang



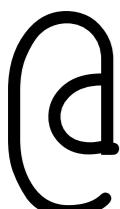
Emily Jiang

Automatically avoiding API misuses

Try it yourself!



Mine Java API usage patterns
(control & data flow)



Mine & validate Java annotation
usage patterns +
generate static analysis checks



<https://sarahnadi.org/smr/api-misuse/>



@sarahnadi



sarahnadi.org