I WANT YOU
TO FIGHT
BAD CODE!

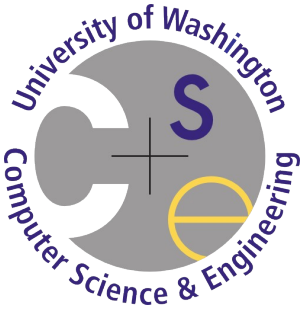
Get the tools & demos from:

`http://types.cs.washington.edu/checker-framework/2012-oscon/`

Developing and Using Pluggable Type Systems

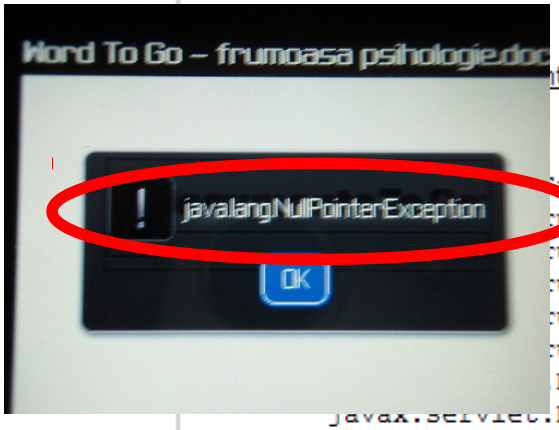
Werner M. Dietl

Michael D. Ernst



University of Washington
Computer Science & Engineering

Software has too many errors



Folder | Adobe LiveC... | poleti.pdf (appli... | Adobe Forums:... | Creating Extern...

ADOBE® LIVECYCLE® ES — Herzlich willkommen, administrator

Startseite | Dienste | Einstellungen | Info | Abmelden

HTTP Status 500 -

type Exception report

...tered an internal error () that prevented it from f...

...xception
...s.action.RequestProcessor.proces...
...uts.action.RequestProcessor.proces...
...uts.action.RequestProcessor.proces...
...uts.action.ActionServlet.process(A...
...uts.action.ActionServlet.doGet(Act...
...http.HttpServlet.service(HttpServl...
javax.servlet.http.HttpServlet.service(HttpServl...
com.adobe.framework.SetCharacterEncodingFilter.d...
com.adobe.idp.um.auth.filter.AuthenticationFilte...
org.jboss.web.tomcat.filters.ReplyHeaderFilter.d...

root cause

java.lang.NullPointerException
com.adobe.edc.ui.policy.PolicyEditAction.initPol...
com.adobe.edc.ui.policy.PolicyEditAction.doExecu...
com.cc.framework.adapter.struts.ActionUtil.execute(Unknown Source)
com.cc.framework.adapter.struts.FWAction.execute(Unknown Source)
com.cc.framework.adapter.struts.FWAction.execute(Unknown Source)
org.apache.struts.action.RequestProcessor.processActionPerform(RequestProcessor.java:431)
org.apache.struts.action.RequestProcessor.process(RequestProcessor.java:236)
org.apache.struts.action.ActionServlet.process(ActionServlet.java:1196)

Java's type system is too weak

- Type checking prevents many errors

```
int i = “hello”;
```

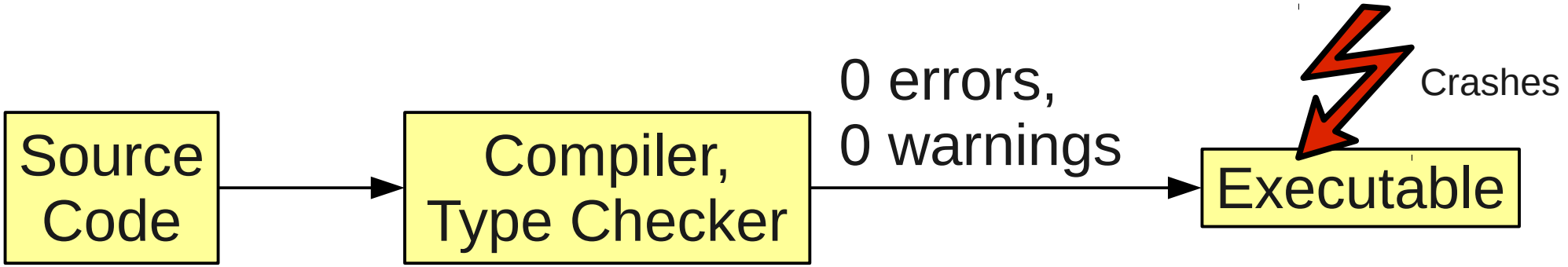
- Type checking doesn't prevent enough errors

```
System.console().readLine();
Collections.emptyList().add(“one”);
dbStatement.executeQuery(userInput);
```

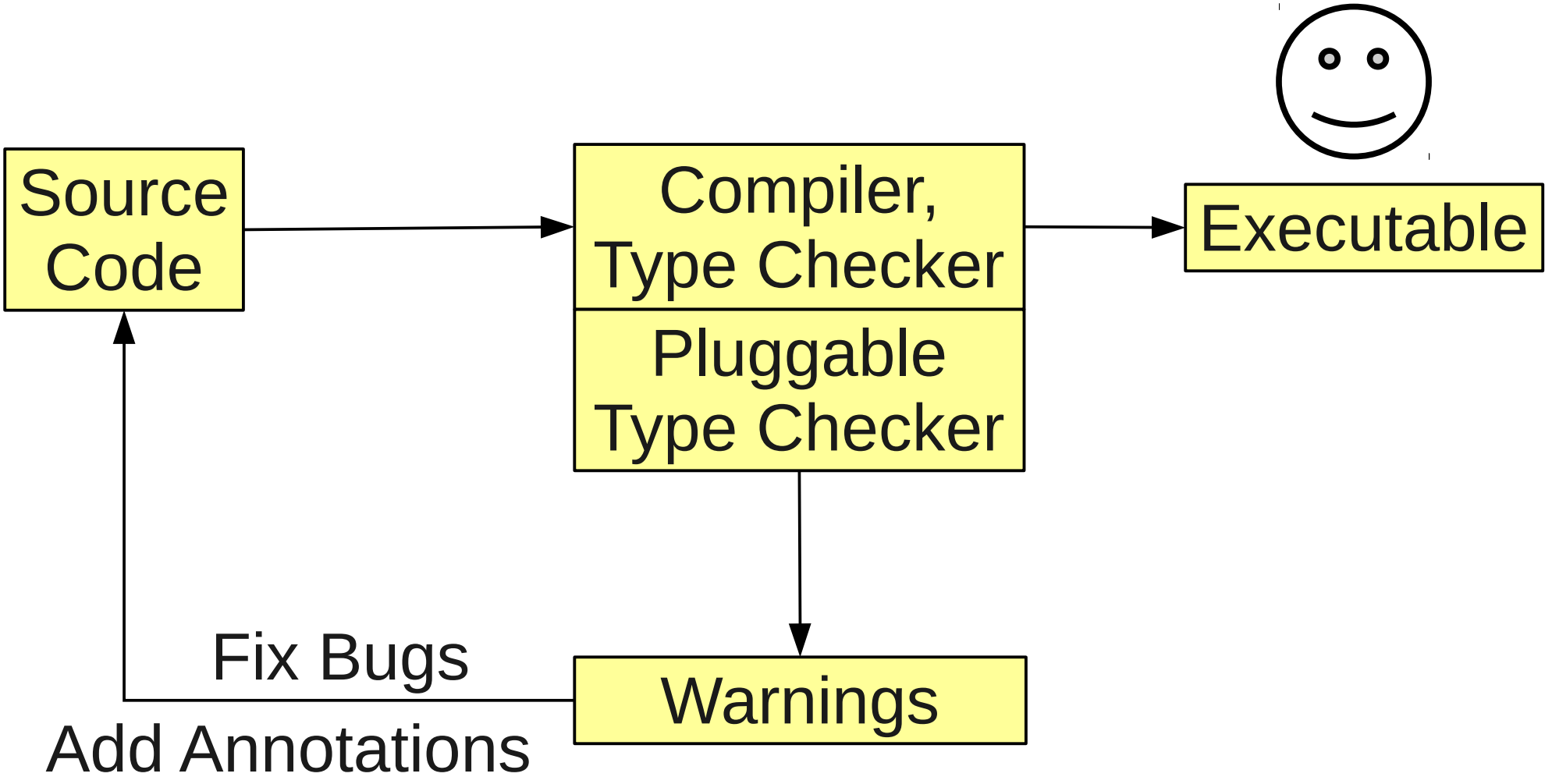
Better type systems can help!

- Null-pointer exceptions [Fähndrich & Leino '03]
- Unwanted mutations [Tschantz & Ernst '05]
- Concurrency errors [Boyapati et al. '02, Cunningham et al. '07]
- ... many more!

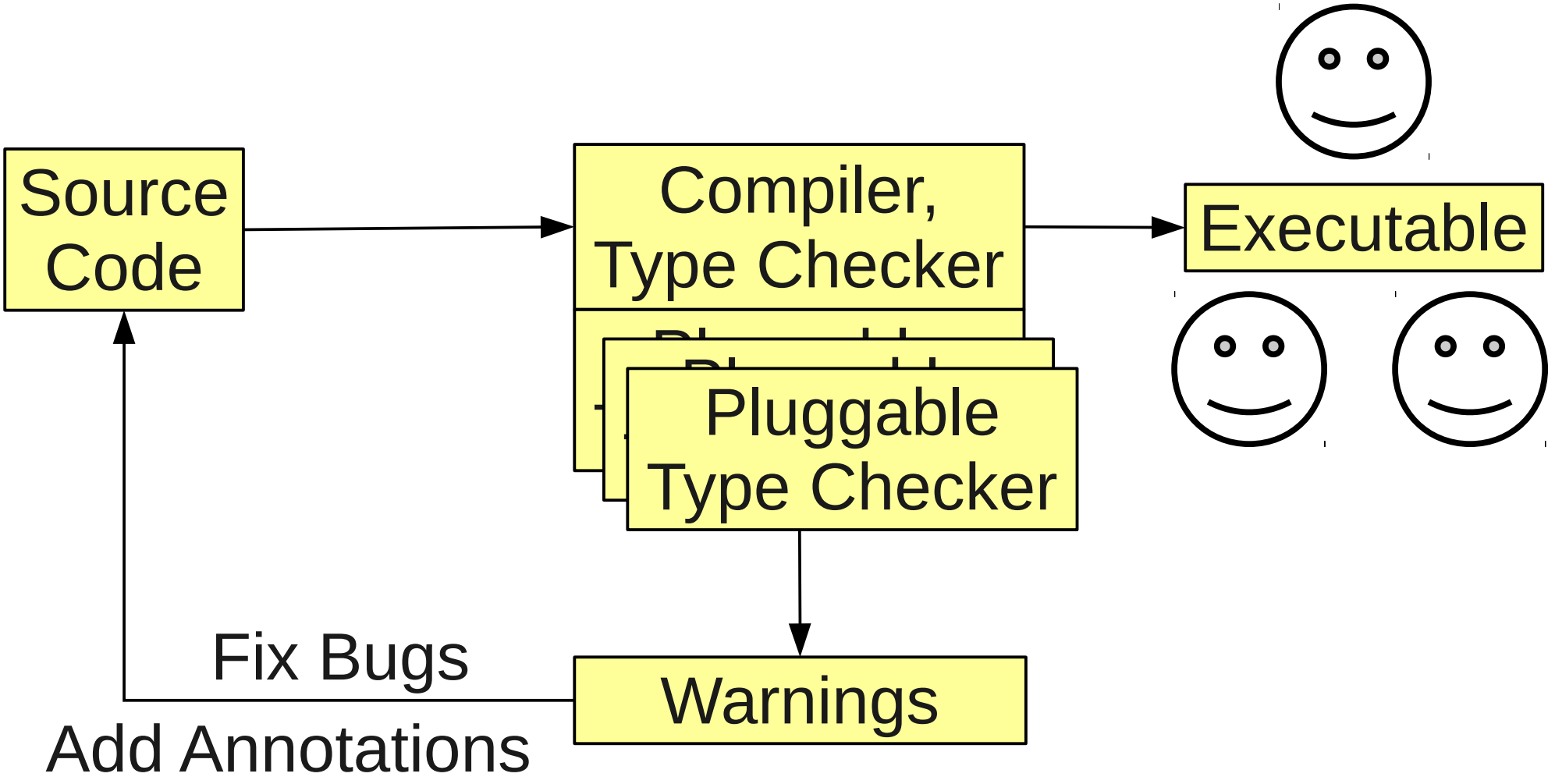
Static type systems



Pluggable type checkers



Pluggable type checkers



Java 8 extends annotation syntax

- Annotations on all occurrences of types

```
@Untainted String query;
List<@NonNull String> strings;
myGraph = (@Immutable Graph) tmpGraph;
class UnmodifiableList<T>
  implements @Readonly List<@Readonly T> {}
```

- Stored in classfile
- Handled by javac, javap, javadoc, ...
- You can use it with Java 5/6/7!
  - Backward compatible: write in `/*@comments*/`

The Checker Framework

- A framework for pluggable type checkers
- “Plugs” into the OpenJDK compiler
- Easy to use

```
javac -processor EncryptionChecker …
```

- Eclipse plug-in, Ant and Maven integration

Example: Regular expressions

```
String regex = getUserInput();
Pattern pat = Pattern.compile(regex);
Matcher mat = pat.matcher(content);

if (mat.matches()) {
  println("Group: " + mat.group(4));
} else {
  println("No match!");
}
```

Regular expression type system

- What runtime exceptions do you wish to prevent?
  `PatternSyntaxException` and `IndexOutOfBoundsException`.
- What properties of data should always hold?
  Indicate strings containing valid regexs and group counts.
- What operations are legal and illegal?
  `Matcher.group` only on regex with minimum group count.

Example: Encrypted communication

```
void send(@Encrypted String msg) {…}

@Encrypted String msg1 = ...;
send(msg1);   // OK

String msg2 = ...;
send(msg2);   // Warning!
```

Encryption type system

- What runtime exceptions do you wish to prevent?
  Invalid information flow.
- What properties of data should always hold?
  Separate encrypted and plain strings.
- What operations are legal and illegal?
  Forbid sending unencrypted data.

Our experience

- Checkers reveal important latent bugs
  - Ran on >3 million LOC of real-world code
  - Found hundreds of user-visible bugs
- Annotation overhead is low
  - Mean 2.6 annotations per kLOC

Null-pointer crash in Google Collections

```
class ForMapWithDefault {
  @Nullable Object defaultValue;

  public int hashCode() {
    return map.hashCode() +
      defaultValue.hashCode();
  }
}
```

java.lang.NullPointerException

- Found 9 such crashes, despite:
  - 45000 tests (2/3 of the LOC)
  - Uses FindBugs @Nullable annotations, no FindBugs warnings

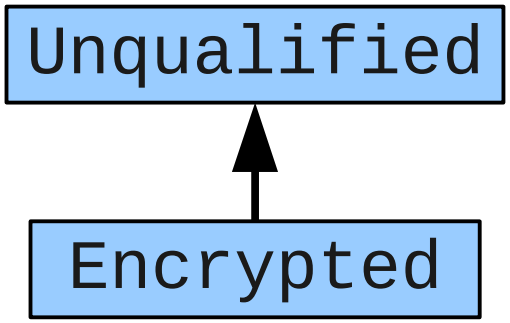
Building checkers is easy

Example: Ensure encrypted communication

```
void send(@Encrypted String msg) {…}

@Encrypted String msg1 = ...;
send(msg1);   // OK

String msg2 = ....;
send(msg2);   // Warning!
```



The complete checker:

```
@TypeQualifier
@Target(ElementType.TYPE_USE)
@SubtypeOf(Unqualified.class)
public @interface Encrypted {}
```

Building complex checkers is possible

Nullness Checker is actually 3 checkers:

- Nullness itself
- Correct object initialization
- Correct usage of keys in map accesses

Refined defaulting:

- Refined flow-sensitive inference
- Heuristics for `Map.get` behavior

SQL injection demo

Goal: no SQL injection attacks possible

- Uses @Tainted and @Untainted annotations

Open-source blogging software

1. Download `personalblog.zip` demo
2. Go into directory
   `personalblog-demo`
3. Requires 8 annotations; we wrote 6
4. Follow me along!

Brainstorming new type checkers

- What runtime exceptions do you wish to prevent?
- What properties of data should always hold?
- What operations are legal and illegal?

- Type-system checkable properties:
  - Dependency on values
  - Not on program structure, timing, ...

Possible type systems

- String normalization (address, dates, ...)
- File existence, legal operations
- Units of measurement and precisions
- Positive/negative numbers
- Network transfer completed
- Type state systems
- String interning
- Bitfields, legal drinking age, fake enumerations

A sampling of type checkers

Property you care about:	Annotation to use:
• Tainting	`@Tainted`
• Java type signatures	`@BinaryName`
• Null dereferences	`@Nullable`
• Concurrency	`@Lock, @GuardedBy`
• Mutability & side effects	`@Immutable`
• Fake enumerations	`@SwingCompassDirection`
• Internationalization	`@Localized`
• Regular expressions	`@Regex`
• Object encapsulation	`@Rep, @Peer, @Any`
• Energy efficiency	`@Approx, @Precise`
• Equality tests	`@Interned`

Your turn to improve your code!

1. Choose a project you care about
   - Or, try pircbot (download from tutorial page)
2. Improve it
   - Apply an existing checker to your code, or
   - Create a new domain-specific type checker

Checker Framework: Much More!

- Powerful framework to develop sophisticated type checkers
- Inference tools
- Annotation tools to insert annotations
- Specification files for libraries

What to do next

- Improve your projects using type checkers
- Develop your own type checkers
- Contribute to the Checker Framework project
- Problems or suggestions? Give us feedback!

I WANT YOU TO FIGHT BAD CODE!

Get the tools & demos from:

`http://types.cs.washington.edu/checker-framework/2012-oscon/`