InsectJ: A Generic Instrumentation Framework for Collecting Dynamic Information within Eclipse

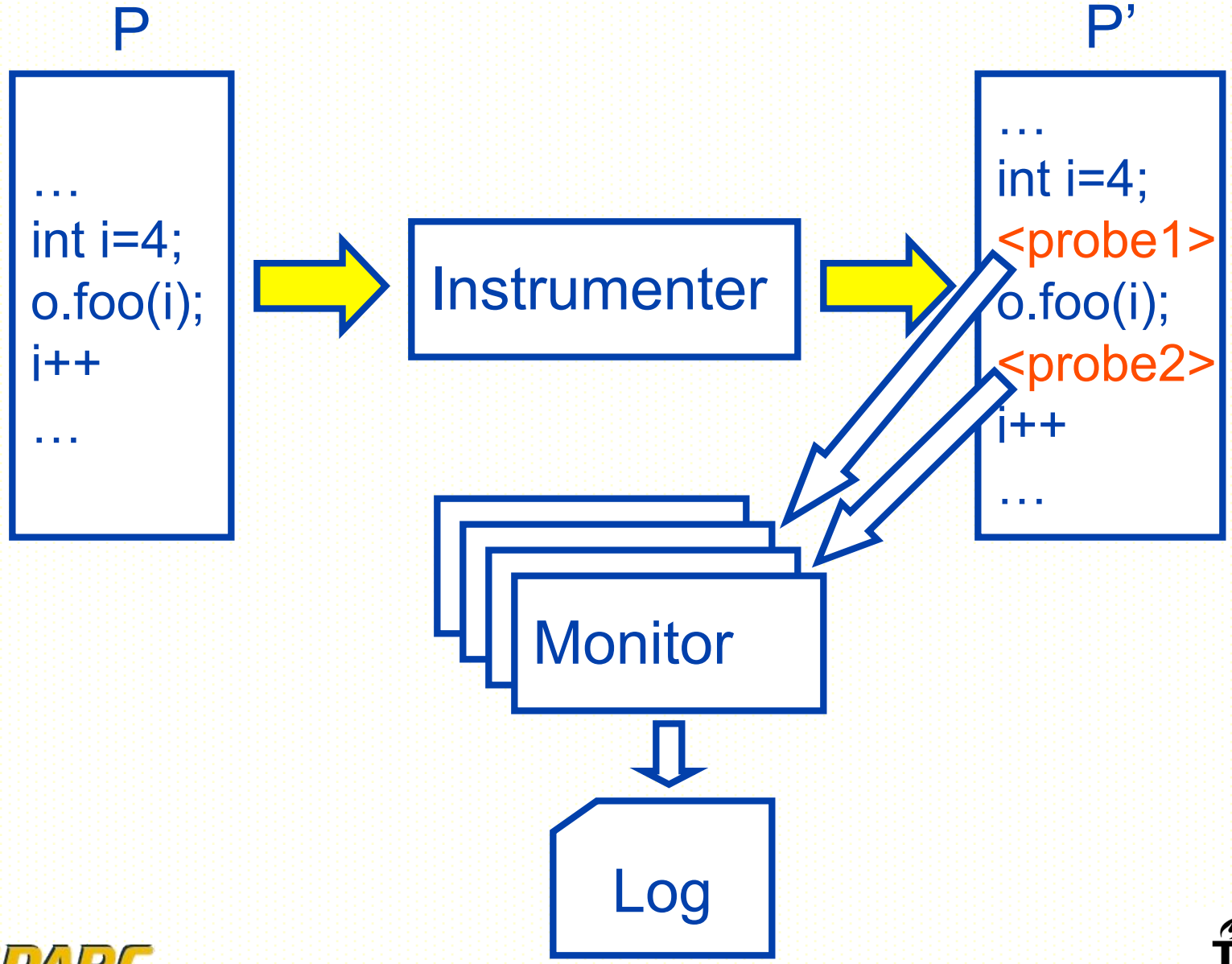
Arjan Seesing and Alessandro Orso

Georgia Institute of Technology

This work was supported in part by an IBM Eclipse Innovation Grant.

Delft University of Technology

Instrumentation Overview



Motivation

Increasing interest in programs' dynamic behavior

Instrumentation commonly used for supporting dynamic analyses

- Coverage
- Program tracing
- Profiling
- Runtime checking
- Mixed static and dynamic analyses
- …

Many issues to address

Key Instrumentation Challenges

Overhead

=> Number of probes

=> Cost of probes

Complexity

=> User unfriendliness

=> Low level details

Non-customizability

=> Ad-hoc solutions

=> Difficult to modify and adapt

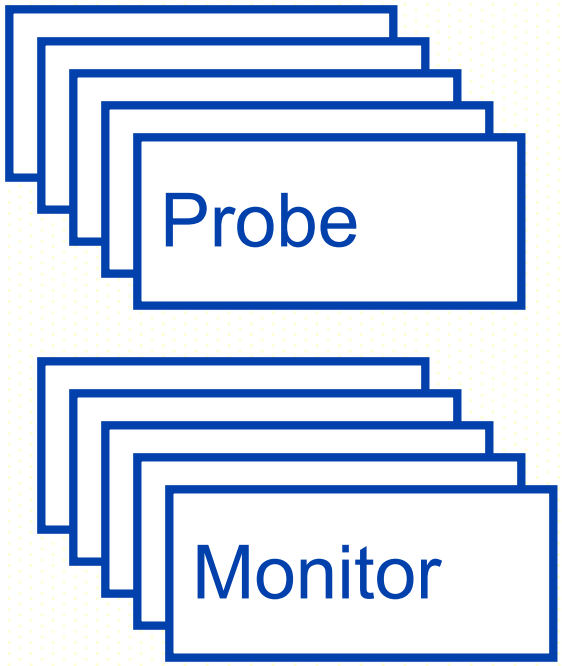
What is InsECT-J?

INStrumentation, **E**xecution, and **C**ollection **T**ool for **J**ava

Flexible, efficient bytecode instrumentation tool for collecting dynamic information

- GUI-based instrumentation (static and on the fly)
- GUI-based monitor creation
- Support for collection of new kinds of data

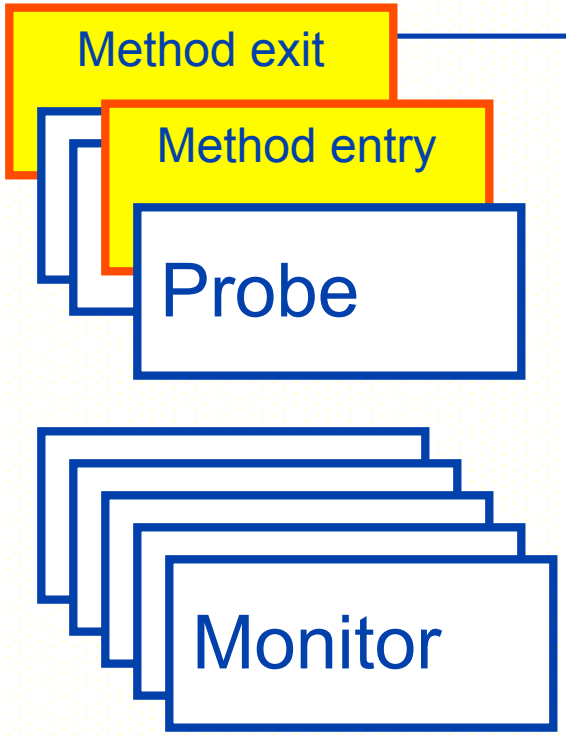
GUI-based Instrumentation



P

```
void foo {
…
}
void bar {
…
}
```

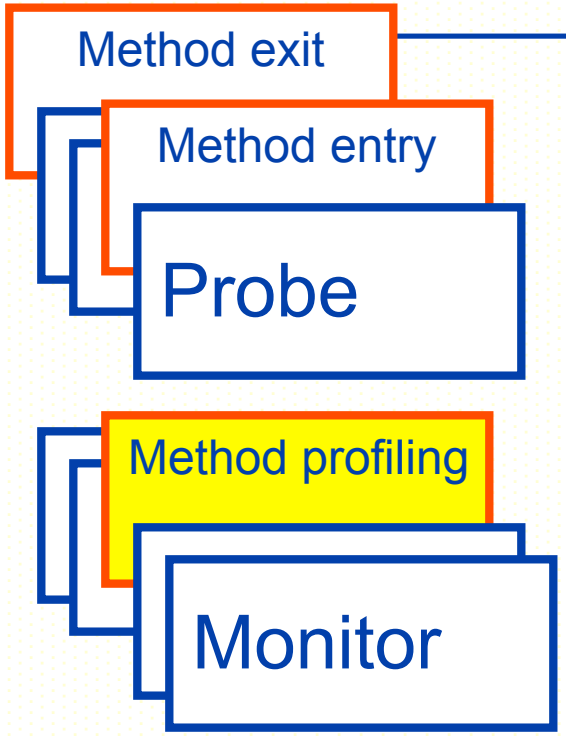
GUI-based Instrumentation



P

```
void foo {
…
}
void bar {
…
}
```

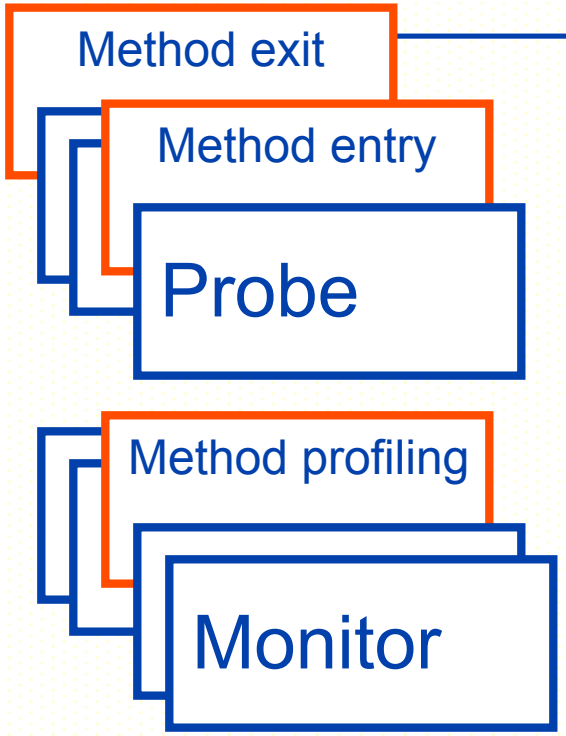
GUI-based Instrumentation



P

```
void foo {
…
}
void bar {
…
}
```

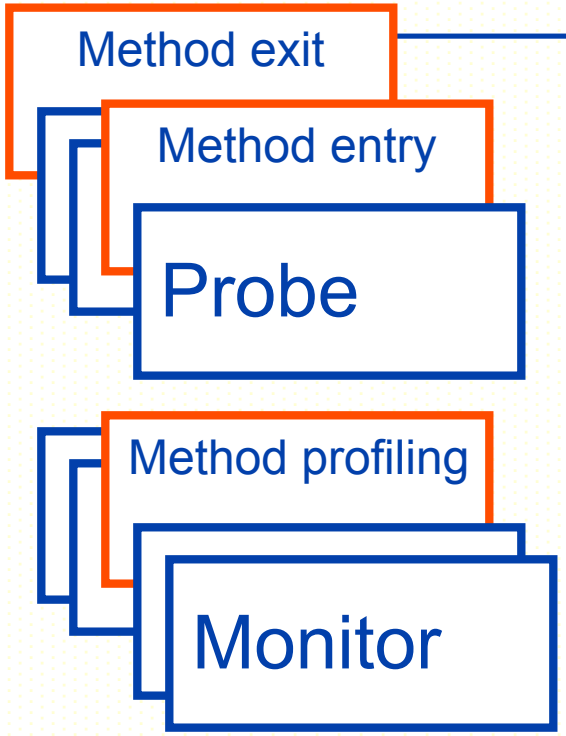
GUI-based Instrumentation

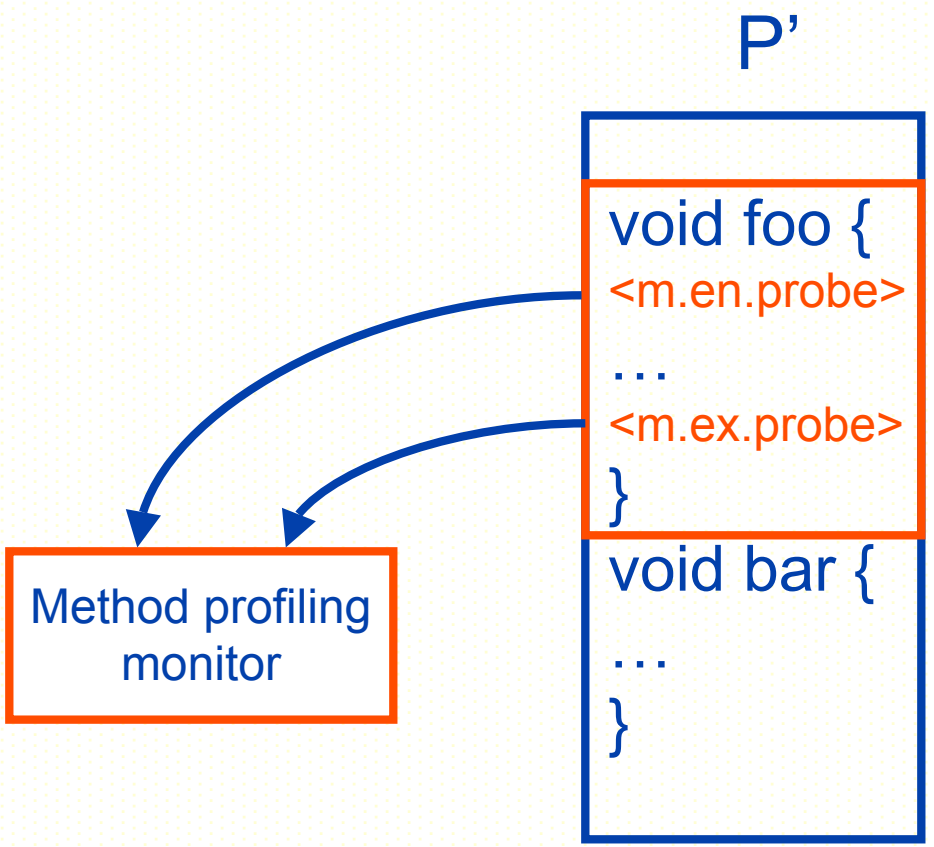


P

```
void foo {
…
}
void bar {
…
}
```

GUI-based Instrumentation

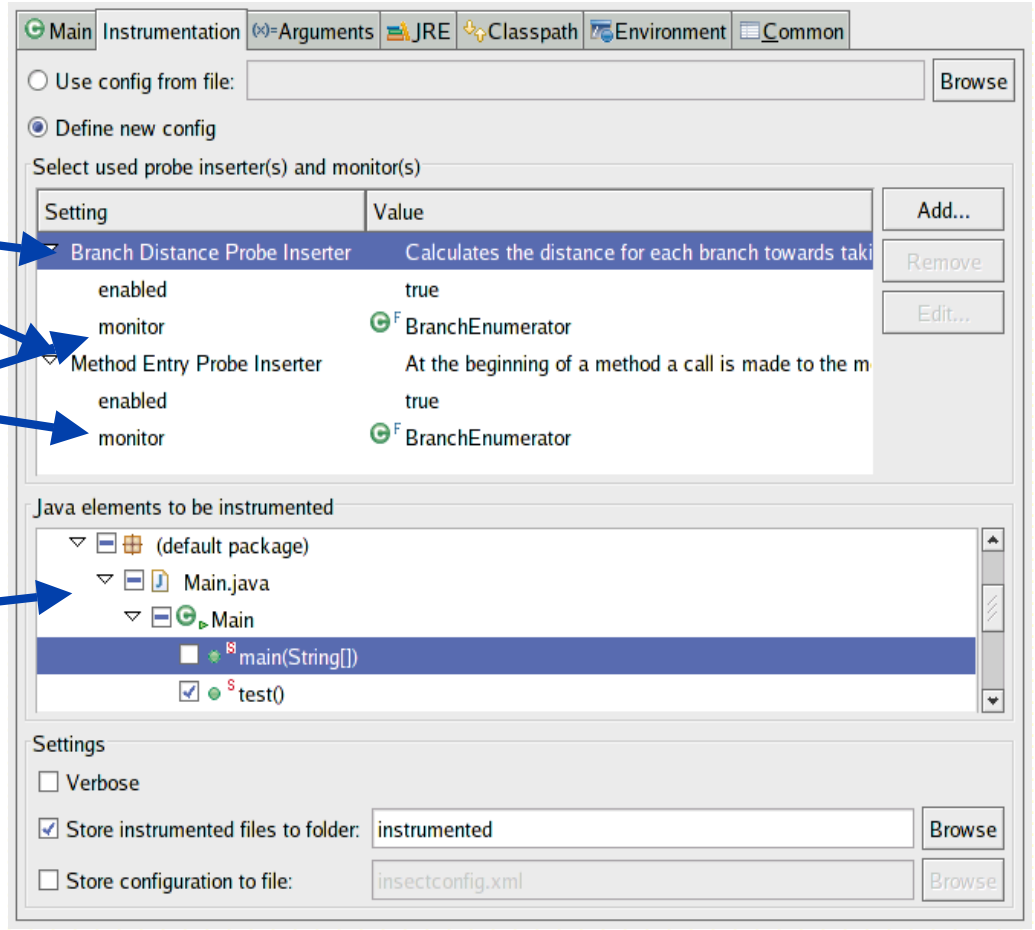




GUI-based Instrumentation

Let users select

1. Which information to collect
2. How to process the information
3. Which parts of the program to instrument



GUI-based Instrumentation

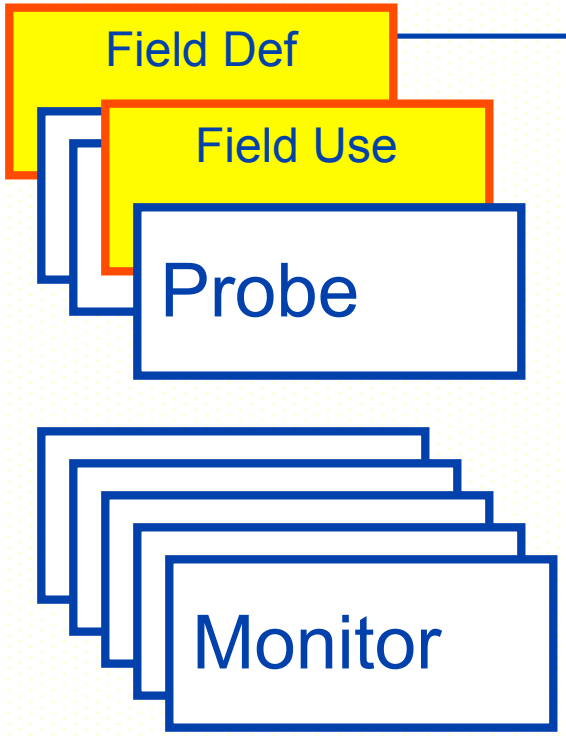
Extensible probe library

- Basic block
- Branch
- Cast
- Call
- Return
- Field read/write
- Method entry/exit
- Throw
- Catch
- Acyclic paths (WIP)
- …

Extensible monitor library

- Block coverage
- Block profiling
- Branch coverage
- Branch profiling
- Method profiling
- Cast monitoring (down-casting)
- DefUse tracing
- Throw/catch coverage
- …

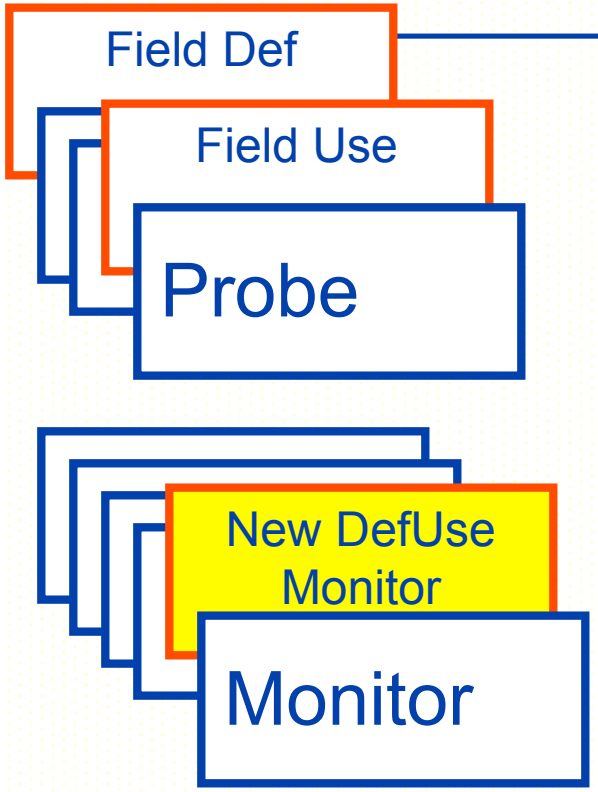
GUI-based Monitor Creation



P

```
void foo {
…
}
void bar {
…
}
```

GUI-based Monitor Creation

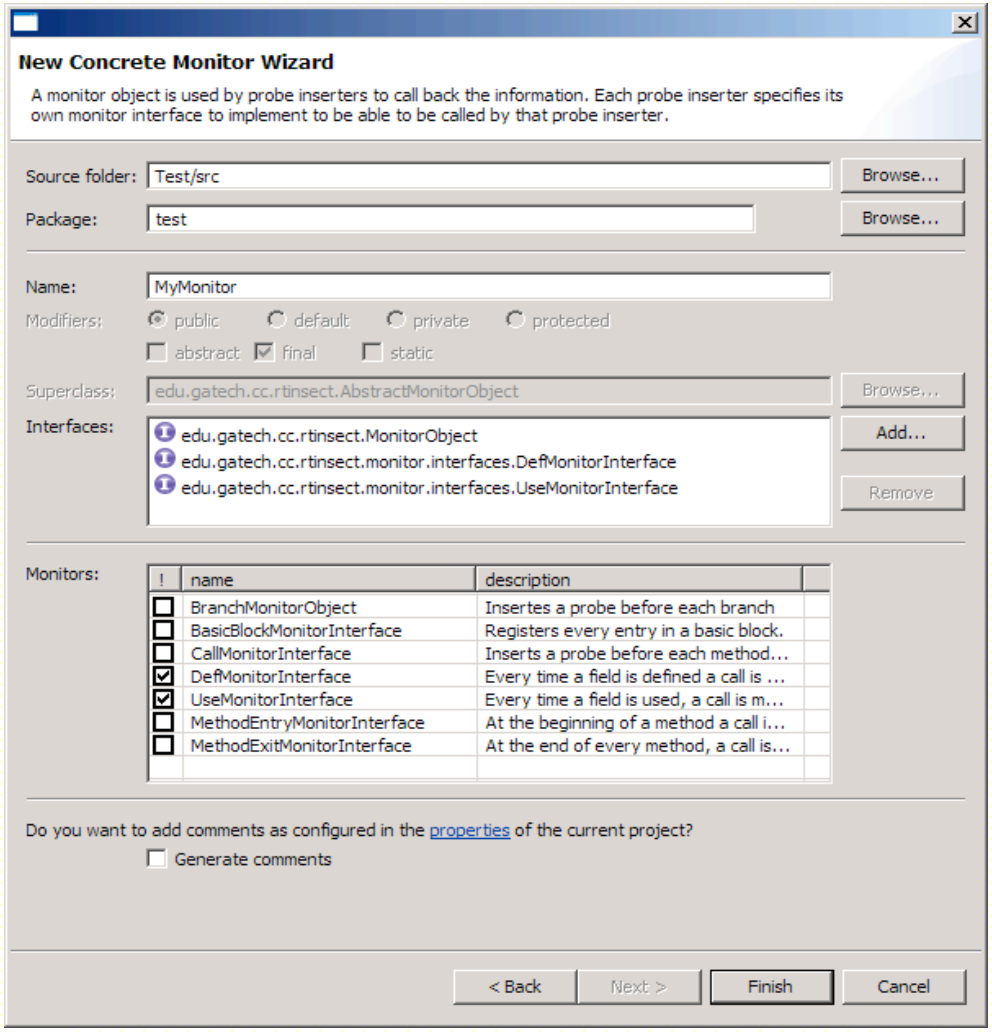


P

```
void foo {
…
}
void bar {
…
}
```

GUI-based Monitor Creation

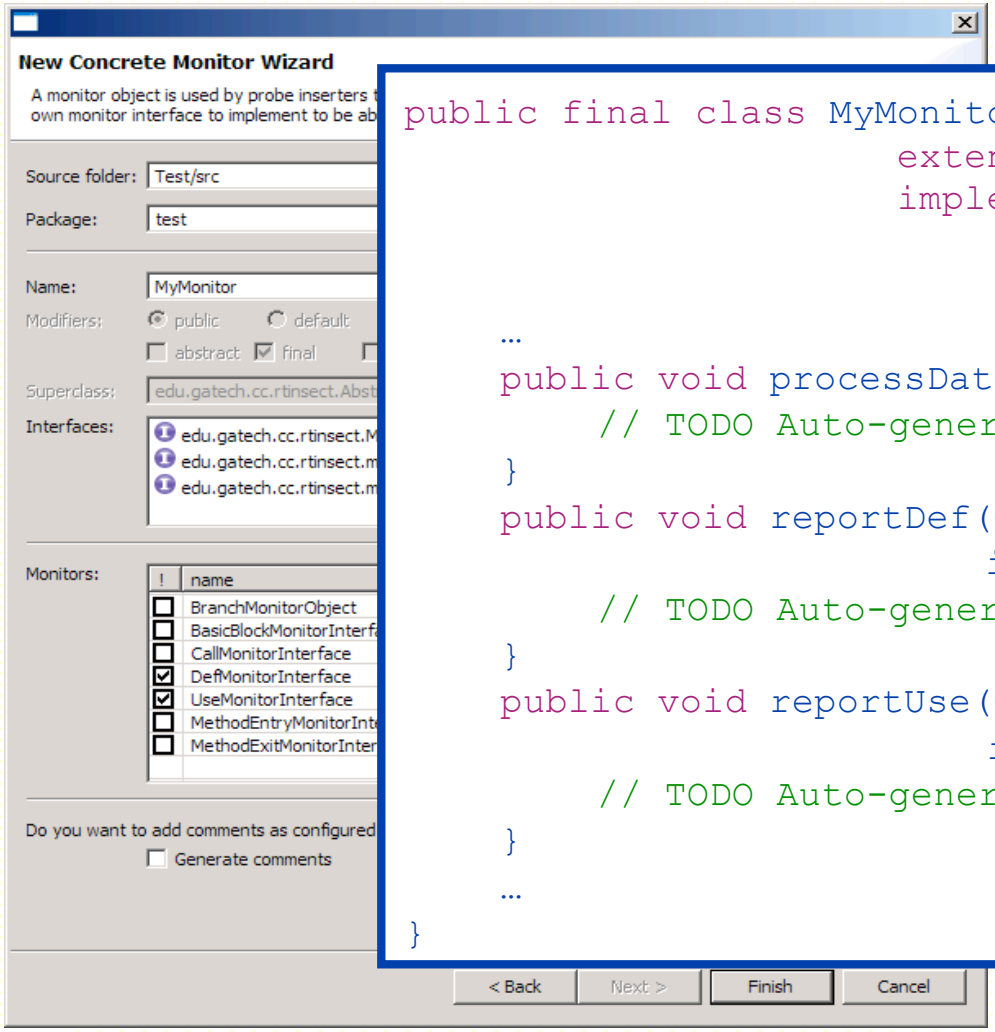
Automated monitor stub creation through a wizard



- Monitors implement interfaces for the corresponding probe(s)
- Monitor wizard creates monitor stub classes

GUI-based Monitor Creation

Automated monitor stub creation through a wizard

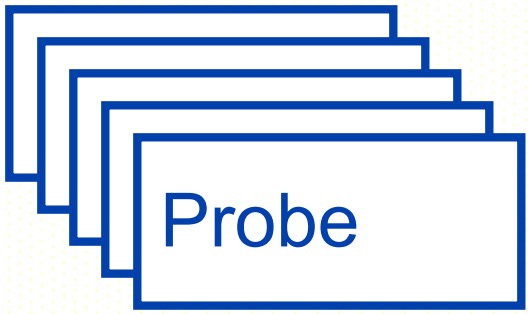


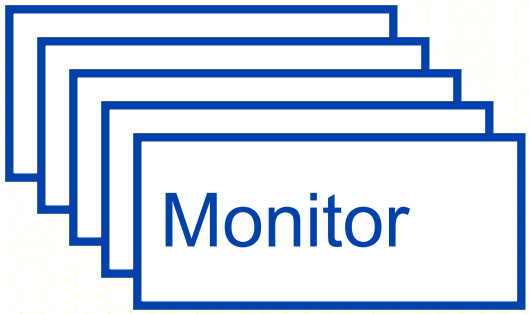
- Monitors implement

```
public final class MyMonitor
                    extends AbstractMonitorObject
                    implements MonitorObject,
                               DefMonitorInterface
                               UseMonitorInterface {

    …
    public void processData() {
        // TODO Auto-generated method stub
    }
    public void reportDef(Object newValue, Object oldValue,
                          int srcLine, int probeId) {
        // TODO Auto-generated method stub
    }
    public void reportUse(Object value, int srcLine,
                          int probeId) {
        // TODO Auto-generated method stub
    }
    …
}
```

Support for Collection of New Kinds of Data



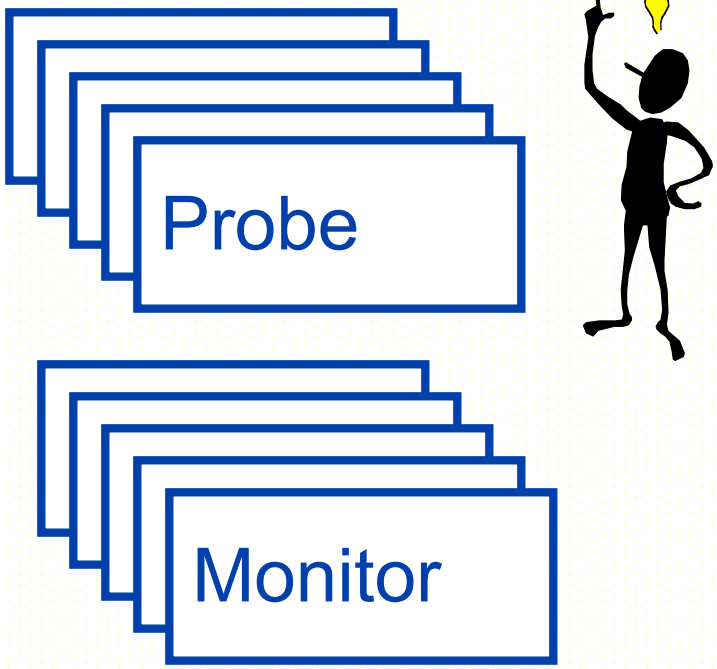


P

```
void foo {
…
}
void bar {
…
}
```


Delft University of Technology

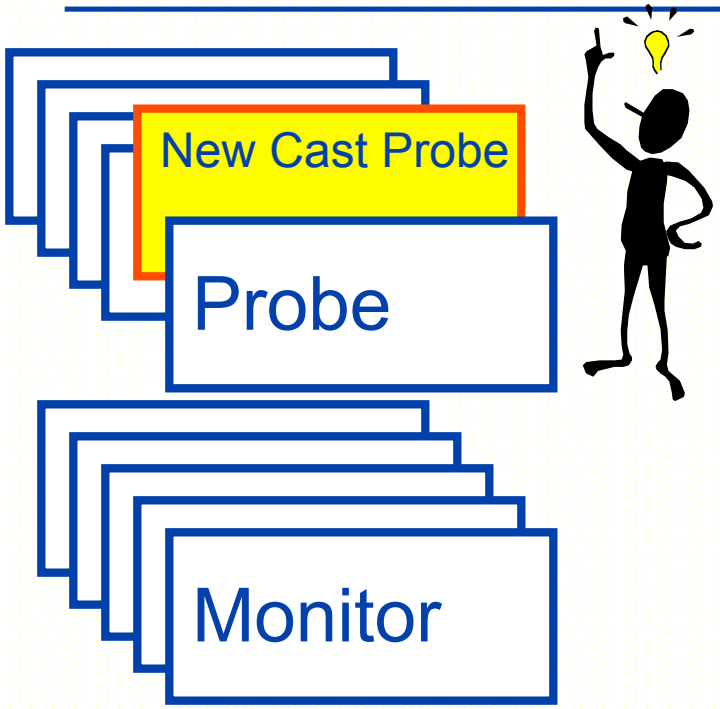
Support for Collection of New Kinds of Data



P

```
void foo {
…
}
void bar {
…
}
```

Support for Collection of New Kinds of Data



P

```
void foo {
…
}
void bar {
…
}
```

Support for Collection of New Kinds of Data

- Allow for easier creation of new types of instrumentation
  - Create extension to Probe plug-in
  - Define monitor interface
  - Implement probe inserter that (1) collects information of interest and (2) calls monitor
- Provide higher-level abstractions for common operations
- Simple instrumentations require only a few lines of code
  (e.g., cast probe is 6 lines of code)

InsectJ Architecture

InsectJ implemented as a set of plug-ins

- Core plugin:
  - Instruments based on a configuration file
  - Provides common functionality for monitors and probe inserters
- Probe inserter plug-ins:
  - Collect different kinds of dynamic data
- Monitor Classes:
  - Predefined
  - User-defined

How Did Eclipse Help (or Did not)

The good:

- Powerful java parser
- Extensible
- Extensive, high-quality documentation
- Developer friendly
- Deployment/technology transfer

The bad:

- Complicated API
- Hard to reuse some high-level elements (JDT-UI)
- Error prone plug-in build process

Future Work

- Instrumentation at lower granularity
  - Statements
  - Context
  - …
- Addition of new probes (e.g., local defs and uses, assignments)
- Use of analysis to optimize (e.g., points-to analysis)
- Tighter integration with Eclipse
  - Visualization
  - Navigation
  - Information report

For More Information

- InsectJ's web site

  http://www.cc.gatech.edu/~orso/software/insectj
- See demo and poster tonight
- Send us email {orso|zeikerd}@cc.gatech.edu