

AJMLC: An Aspect-Oriented Implementation of jmlc

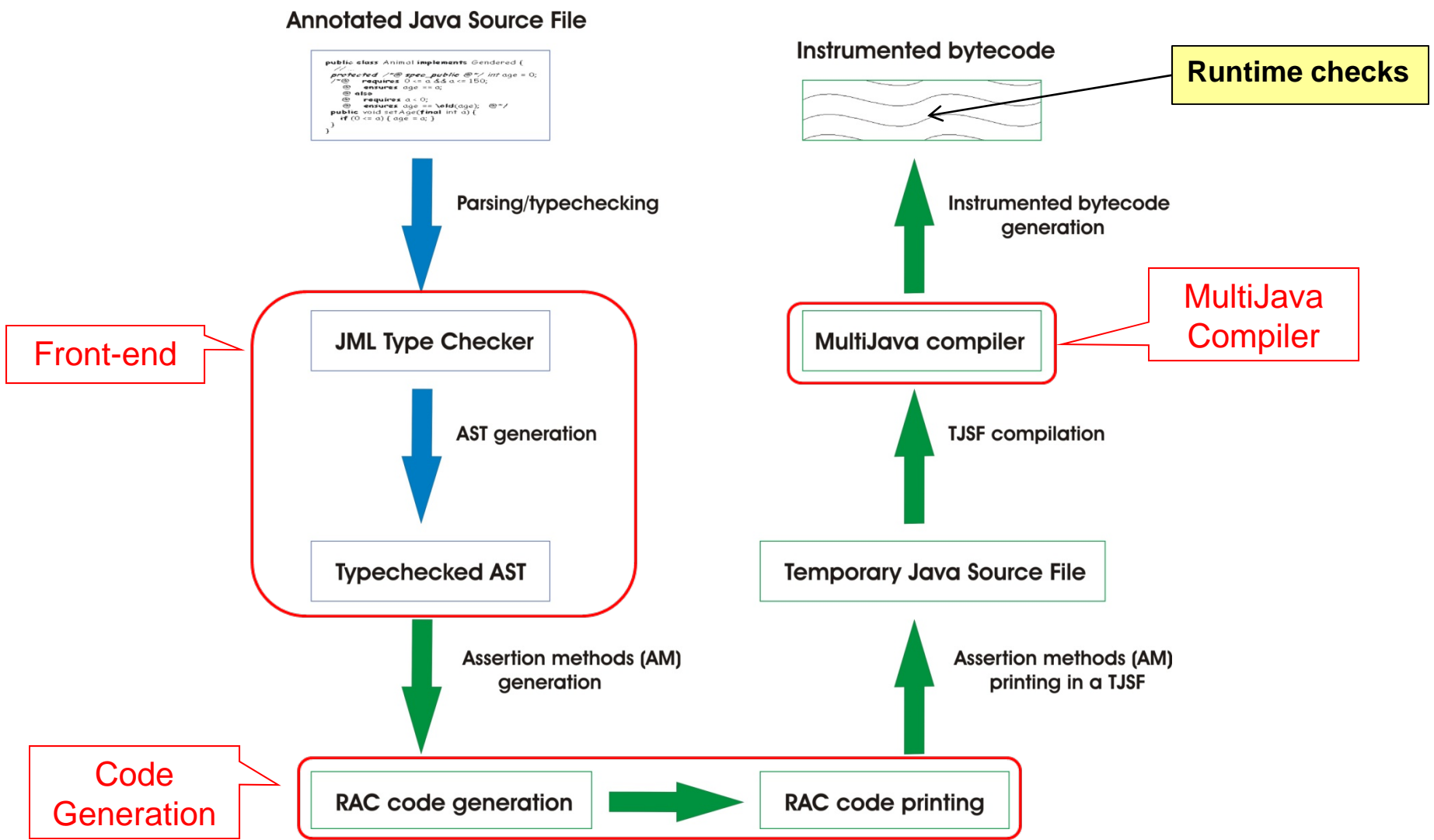
Henrique Rebêlo (hemr@cin.ufpe.br)

Federal University of Pernambuco, Recife, Brazil

Gary T. Leavens

University of Central Florida, Orlando, USA

(JML2's) jmlc: compilation passes





Main Problem

- jmlc limitation
 - jmlc doesn't work outside of Java SE
 - Example: Java ME platform
 - Data structures (e.g. *HashSet*)
 - Java reflection mechanism

Not supported
by Java ME!



jmlc with Classical Semantics...

↪ Two-valued logic

JML-5.4

- Instrumentation code
 - Source code instrumentation ✗
 - Bytecode instrumentation ✗
- Performance
 - Many reflective calls ✗
- Error reporting
 - Context information ✓
 - Server locating ✓
 - Client locating ✗





jmlc Current Status...

↪ Three-valued logic

JML-5.5

■ Instrumentation code

- Source code instrumentation 
- Bytecode instrumentation 






still need enhancement

■ Performance

- Many reflective calls 

■ Error reporting

- Context information 
- Server locating 
- Client locating 



Solution: Aspect-Oriented jmlc

↳ New assertion semantics (Strong Validity)

AJML-1.0

- Instrumentation code
 - Source code instrumentation ✓
 - Bytecode instrumentation ✓
- Performance
 - No reflective calls ✓
- Error reporting
 - Context information ✓
 - Server locating ✓
 - Client locating ✓



ajmlc: compiler passes

Annotated Java Source File

```
public class Animal implements Generated {  
    // protected // @ spec_public @? int age = 0;  
    // @ requires 0 <= a && a <= 150; @  
    // @ ensures age == a;  
    // @ also @ requires a < 0;  
    // @ ensures age == ValidAge; @?/  
    public void setAge(int a) {  
        if (0 <= a) { age = a; }  
    }  
}
```

Parsing/typechecking

JML Type Checker

AST generation

Typechecked AST

Aspect Assertion methods (AAM) generation

Aspect RAC code generation

Instrumented bytecode



Instrumented bytecode generation

AspectJ compiler

TASF compilation

Temporary Aspect Source File

Aspect Assertion methods (AAM) printing in a TJSF

Aspect RAC code printing

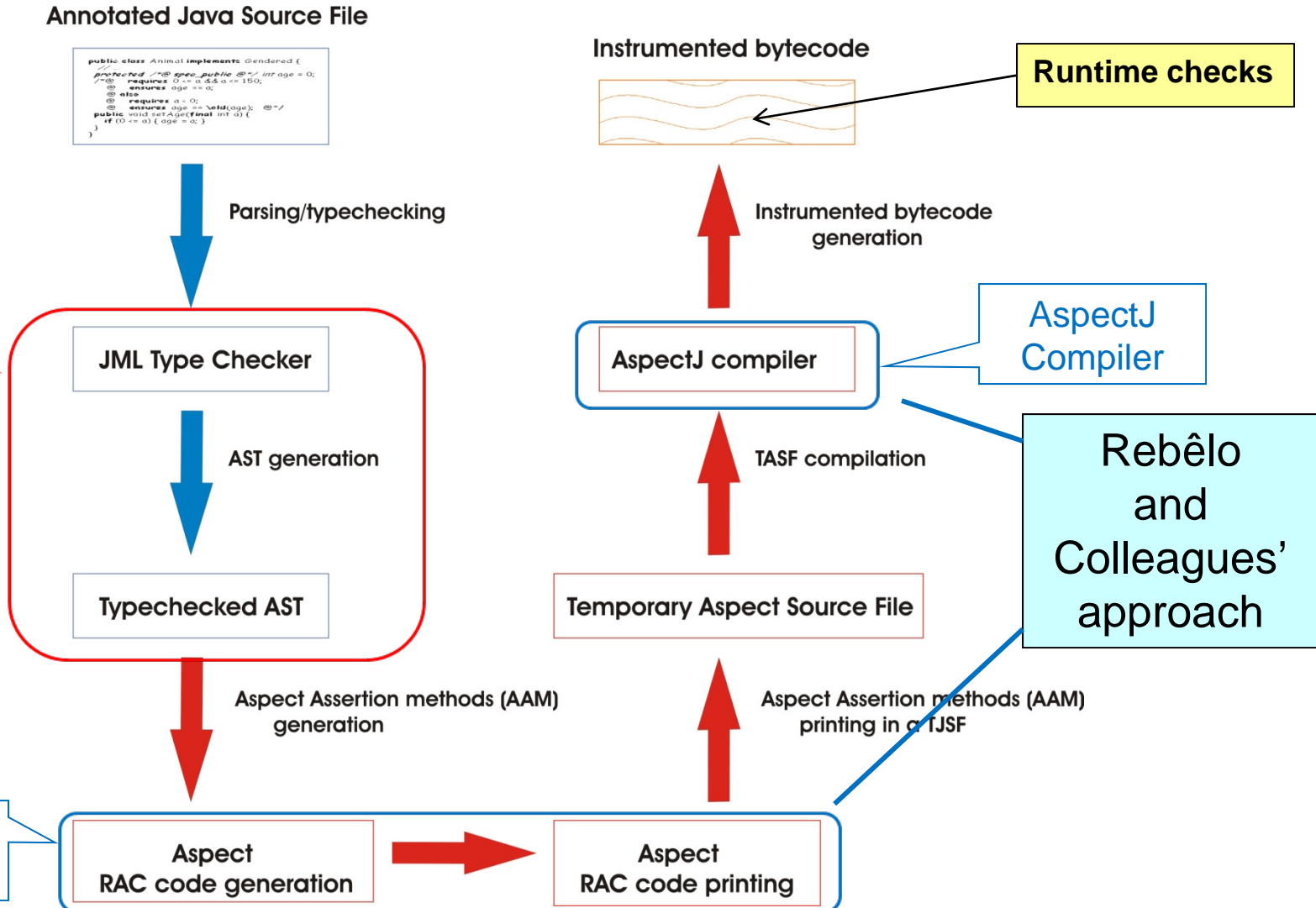
Runtime checks

Front-end

Aspect Code Generation

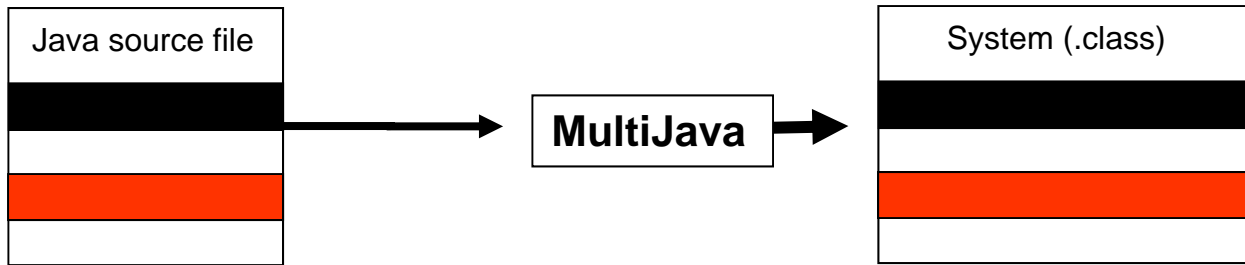
AspectJ Compiler

Rebêlo and Colleagues' approach

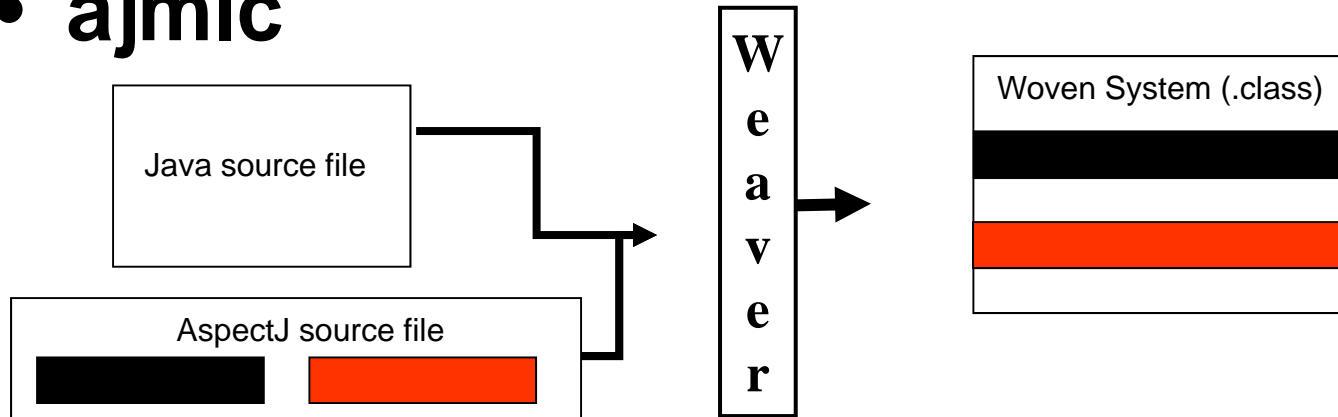


generated code: jmlc VS ajmlc

- **jmlc**



- **ajmlc**



ajmlc: Better Error Messages

jmlc-5.6

```
1: class C {
2:
3:   //@ pre x > 0;
4:   void m(int x) {
5:     ...
6:   }
7: }
```

```
1: class Client {
2:
3:   static void main(){
4:     C c = new C();
5:     c.m(-10);
6:   }
7: }
```

```
c:\JML\JML-5.6\bin>jmlrac Client
Exception in thread "main" org.jmlspecs.jml:
ror: by method C.m
    at Client.main(Client.java:293)
```

ajmlc-1.0

```
<terminated> Client [Java Application] C:\Java\jdk1.5.0_09\bin\ja
Exception in thread "main" org.jmlspecs.ajm
    at AspectJMLRac_C.ajc$before$Aspect
    at C.m(C.java:5)
    at Client.main(Client.java:5)|
```

Further Information



- Ajmlc release now available!
- Publications
- ...

Contacting Henrique...



Ph.D. student Henrique Rebêlo

Specialist on **AOSD**, **DbC**, Static Metrics, **JML**,
Software Architecture, Product Lines, Empirical Studies

email: **hemr@cin.ufpe.br**

Informatics Center

Federal University of Pernambuco, UFPE

Brazil