Whiley: a Platform for Research in Software Verification

David J. Pearce and Lindsay Groves

School of Engineering and Computer Science
Victoria University of Wellington

`http://whiley.org`

Documentation: What's the problem?

```
Bytecode Load(int register, JvmType type)
```

Creates an object representing a bytecode which loads a value of a given type from a given register (e.g. `iload`, `aload`, etc). **Note:** the register index must be between 0 and 255.

- This documentation **is informal**
- This documentation is **not enforced**
- Can we do anything **about this**?

What is Whiley?

```
Bytecode Load(int idx, JvmType type)
    requires idx >= 0 && idx <= 255
```

- A language designed specifically to simplify **verifying software**
- Several trade offs e.g. **performance for verifiability**
  - *Unbounded Arithmetic, value semantics, etc*
- **Goal**: to statically verify functions meet their specifications

Overview of Whiley (Brief)

Overview: Types in Whiley

- **Primitives**:
  - e.g. `any`, `null`, `bool`, `int`, `real`, `char`
- **Collections** (lists, maps, sets):
  - e.g. `[int]`, `{string}`, `{int=>string}`
- **Records** and **Tuples**:
  - e.g. `{int x, int y}`, `(int,int)`
- **Unions** and **Negations**:
  - e.g. `int|null`, `!int`

Flow Typing

```
int sum([int] items):
    r = 0
    for item in items:
        r = r + item
    return r
```

- A **flow-sensitive** approach to type checking
- Types declared only for **parameters** and **returns**
- Variables can have **different types!**
- Conditionals and/or assignments cause **retyping**

Flow Typing: Example 1

```
define Circle as {int x, int y, int r}
define Rect as {int x, int y, int w, int h}
define Shape as Circle | Rect

real area(Shape s):
    if s is Circle:
        return PI * s.r * s.r
    else:
        return s.w * s.h
```

- Variables are automatically **retyped** by type tests
  - (even on the false branch)

Flow Typing: Example 2

```
null|int indexOf(string str, char c):
    ...

[string] split(string str, char c):
    idx = indexOf(str,c)
    if idx is int:
        below = str[0..idx]
        above = str[idx..]
        return [below,above]
    else:
        return [str]
```

- Here, **union type** protects against `null` dereference!

Verification with Whiley

Verification: Example 1

- A very **simple** example:

```
int f(int x) ensures $ >= 0:
    return x
```

- Above is invalid and **does not verify**. Can **fix** it like so:

```
int f(int x) requires x > 0, ensures $ >= 0:
    return x
```

- This will now **verify**

Verification: Example 2

- Another **simple** example:

```
int abs(int x) ensures $ >= 0:
    if x >= 0:
        return x
    else:
        return -x
```

- Above code is valid and **will verify**

- Verifying compiler **reasons precisely** about information flow

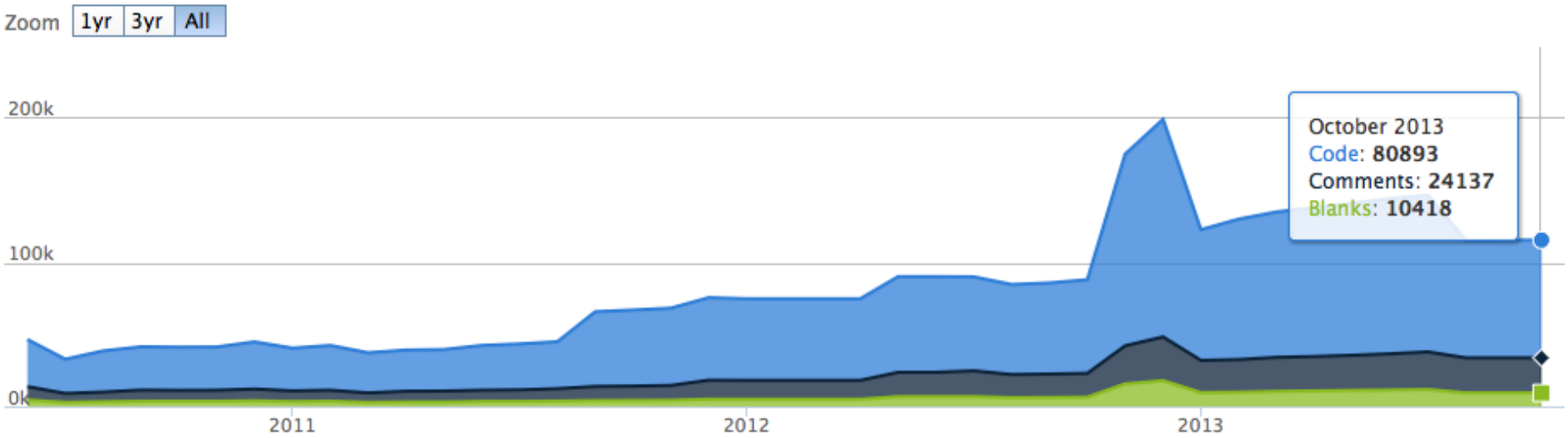
Verification: Example 3

```
null|int indexOf(string str, char c):
    i = 0
    while i < |str| where i >= 0:
        if str[i] == c:
            return i
        i = i + 1
    return null
```

- Above code is valid and **will verify**
- Verifying compiler proves array indices **always within bounds**

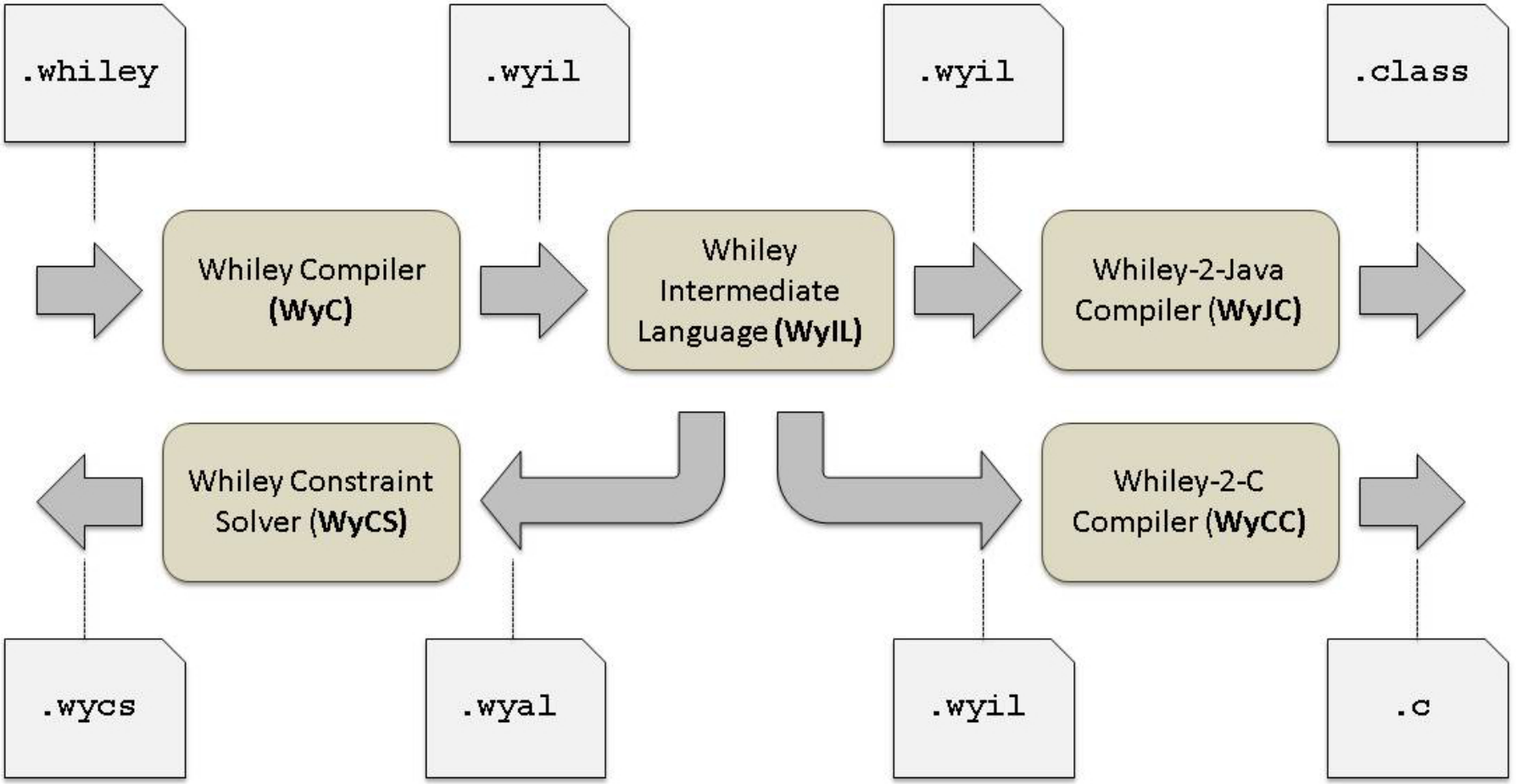
About Whiley

History of Whiley

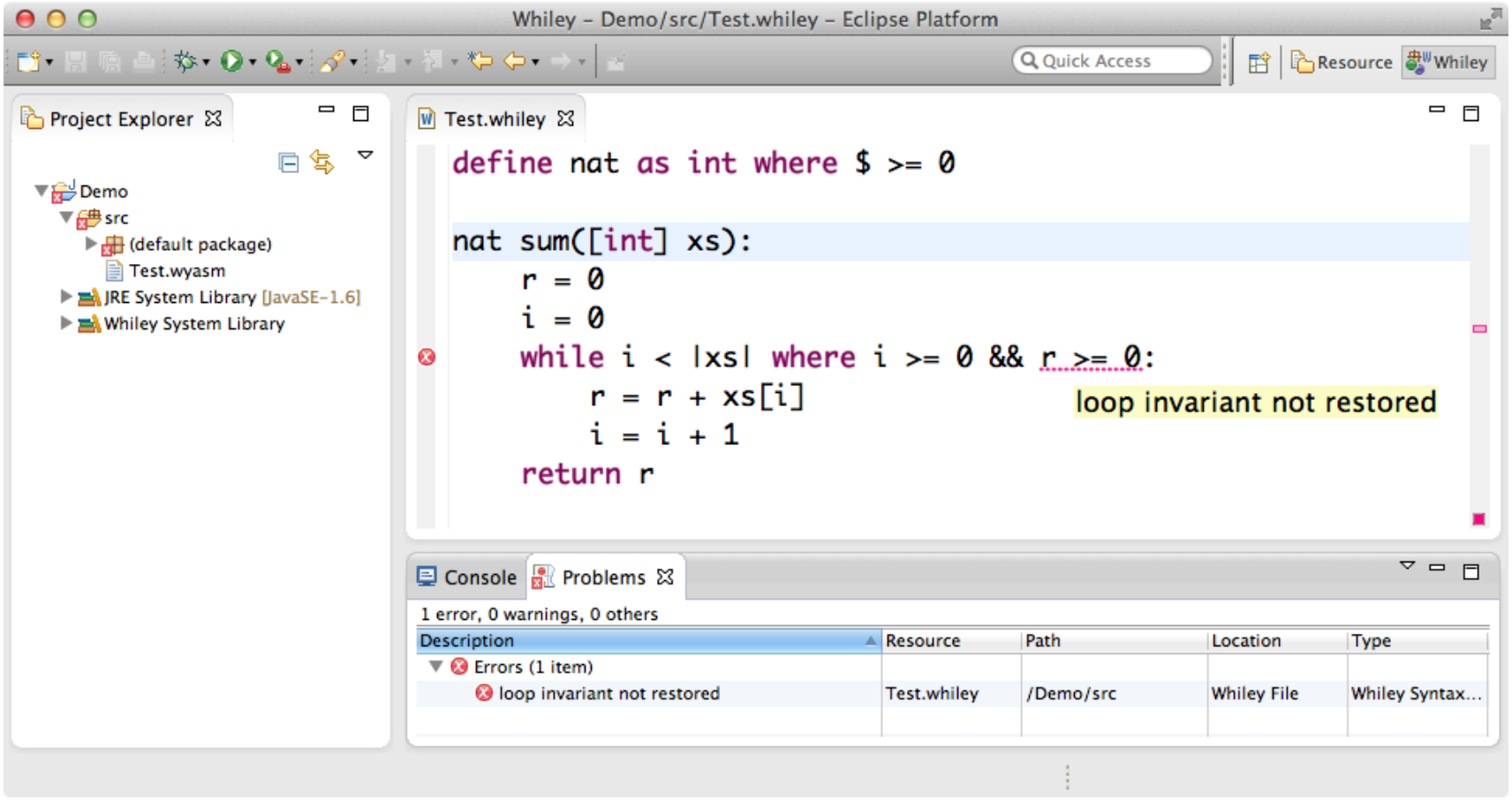


- 2009 — **Initial** version of Whiley released (GPL Licence)
- 2010 — **GitHub** repository and **`http://whiley.org`** go live
- 2010 — **Version 0.3.0** released (BSD Licence)
- 2013 — **Latest version** 0.3.20 (approx 81KLOC)
- 2014 — **Version** 0.4.0 released?

Architecture of the Whiley Compiler



Eclipse plugin for Whiley!



- **Update Site:** `http//whiley.org/eclipse`

http://whiley.org