

Wollok: en el aula y más allá

Javier Fernandes^{1,2} Nicolás Passerini^{1,2,4}

Pablo Tesone^{3,1,2,4} Débora Fortini^{1,4}

Nahuel Palumbo⁴ Juan Contardo⁴ Carlos Raffellini⁴

¹Universidad Nacional de Quilmes

²Universidad Nacional de San Martín

³Universidad Nacional del Oeste

⁴Universidad Tecnológica Nacional - F.R. Buenos Aires.

Workshop de Ingeniería en Sistemas y Tecnologías de la Información
19/09/2015

Agenda

- 1 Introduccion
- 2 Desarrollo de Wollok
- 3 Features Avanzados
- 4 Wollok Game
- 5 Experiencia en el Aula

¿Por qué es difícil aprender OOP?

- Enfoque en un lenguaje particular
- Demasiados conceptos

```
package examples;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- **Entornos de desarrollo** limitados o inadecuados
- Aprender a programar exige **aprender a abstraer**

¿Por qué es difícil aprender OOP?

- Enfoque en un lenguaje particular
- Demasiados conceptos

```
package examples;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- **Entornos de desarrollo** limitados o inadecuados
- Aprender a programar exige **aprender a abstraer**

Antecedentes

Ozono

<http://ozono.uqbar-project.org/>

- Basado en Smalltalk
- **Recorrido incremental**
 - Metamodelo simplificado: sin clases ni herencia
 - Foco en objeto - mensaje - referencia - polimorfismo
- Herramientas de visualización de código
 - Diagramas de objetos / clases

Gobstones

<http://www.gobstones.org/>

- Cuidadosa selección de los elementos sintácticos
- Elimina la necesidad de entrada-salida
- Separación entre elementos con efecto y elementos puros

Antecedentes

Ozono

<http://ozono.uqbar-project.org/>

- Basado en Smalltalk
- **Recorrido incremental**
 - Metamodelo simplificado: sin clases ni herencia
 - Foco en objeto - mensaje - referencia - polimorfismo
- Herramientas de visualización de código
 - Diagramas de objetos / clases

Gobstones

<http://www.gobstones.org/>

- Cuidadosa selección de los elementos sintácticos
- Elimina la necesidad de entrada-salida
- Separación entre elementos con efecto y elementos puros

¿Qué es Wollok?

- Lenguaje + muchas herramientas
- Optimizados para la enseñanza
- Cercanos a las herramientas profesionales *mainstream*
- Proyecto abierto




¿Qué es Wollok? - Pensado para enseñar

- Sintaxis cuidada
 - selección de keywords
 - return obligatorio
 - énfasis en objetos y mensajes (aunque no todo es objeto mensaje)
- Combina object-based con class-based programming
- APIs minimalistas (ej: colecciones)
- Import system

¿Qué es Wollok? - Cercano al mainstream

- Ambiente de objetos basado en archivos
- Framework de testing integrado
- Literales para listas y conjuntos (próximamente diccionarios)
- Manejo de excepciones

Desarrollo de Wollok

- OpenSource: LGPLv3
- Stack:  Eclipse XText + Xtend Lang
- SCM:
 - **Código:**  GitHub ([uqbar-project/wollok](https://github.com/uqbar-project/wollok))
 - **Build:** Maven + Tycho
 - **Continuous Integration:**  Travis
 - **Continuous Deployment**
 - **Coverage:** coveralls + jacoco
- Testing & TDD

Desarrollo de Wollok

Continuous Integration & Deployment

- **GitFlow**
 - Feature Branches
 - Pull-Requests
 - *dev* → *master* ← *hotfixes*
- **Integration:**
 - Travis
 - compile, test, coverage, deploy
- **Deployment:**
 - **Productos** (IDE): multiples plataformas
 - **Update Sites**
 - **WDK**
 - 2 Ambientes: Stable & Dev

Desarrollo de Wollok

Testing & TDD

- 87% Cobertura
- **Runtime**
 - Testean ejecución
 - Interprete
 - **JUnit + iDSL**
- **Estáticos**
 - **Chequeos:** XPect
 - **Type System:** JUnit + iDSL
 - **Autocomplete:** XPect
 - **Formateo:** JUnit + iDSL
- **Pendientes**
 - Quick-Fixes
 - Refactors

Testing & TDD

Runtime

Testeo del Intérprete

```
class PostFixOperationTestCase extends AbstractWollokInterpreterTestCase {
```

```
    @Test
```

```
    def void testPlusPlus() {'''
```

```
        program p {
```

```
            var n = 1
```

```
            n++
```

```
            assert.that(n == 2)
```

```
        }''' .interpretPropagatingErrors
```

```
    }
```

Testing & TDD

Cheques Estáticos

Testeo de Cheques Estáticos

```
/* XPECT_SETUP org.uqbar.project.wollok.tests.xpect.WollokXpectTest END_SETUP */
```

```
class Golondrina {  
    var energia = 100
```

```
    method energia() {
```

```
        // XPECT errors --> "Cannot assign a variable to itself. It does not have any effect" at "energia"
```

```
        energia = energia
```

```
}
```

Features Avanzados

- Debugger
- Tests
- Sublime Plugins
- I18N

Features Avanzados

Debugger

Debugger

- UI integrada a Eclipse Debug
- Breakpoints: agregar, remover, deshabilitar, etc
- Step, into, out
- Inspeccionar variables
- Diagrama de Objetos

Features Avanzados

Tests

Tests

The screenshot shows the 'Wollok Tests' tab in the IDE. At the top, there are statistics: 'Tests: 2', 'Run: 2', and 'Errors: 1'. Below this is a large red bar indicating a failure. The test file path is shown as `../wollok-dev/wollok-tests/src/wollok/tests/test.wtest`. The test cases listed are 'Energia del Alpiste es 100' (passed) and 'Energia del Maiz es 200' (failed, highlighted in blue). The error message at the bottom reads: 'Assert Error: Expected [100] but found [200] - Expected [100] - Actual [200]'.

Features Avanzados

Soporte para Sublime

Soporte para Sublime

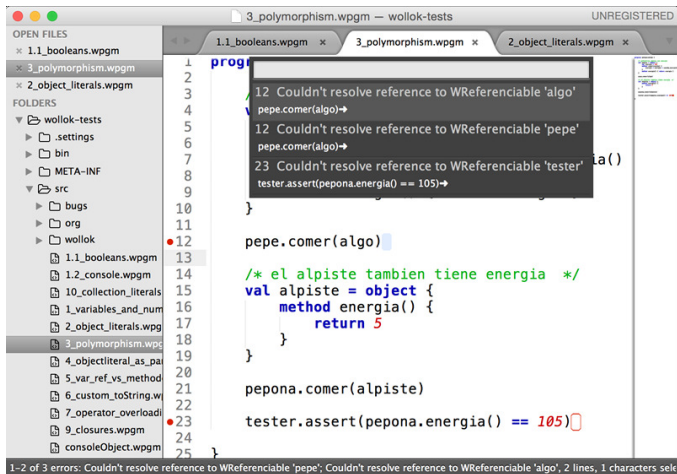
- WDK
 - No IDE
 - ~ 70MB (vs ~ 140)
 - Headless: wchecker, winterpreter, wtest
- Syntax highlight
- Templates
- Linter

Sublime Support

Syntax Highlight

```
1 program polymorphism {
2
3     // golondrina pepona con energia
4     val pepona = object {
5         var energia = 100
6         method comer(comida) {
7             energia = energia + comida.energia()
8         }
9         method energia() { return energia }
10    }
11
12    /* el alpiste tambien tiene energia */
13    val alpiste = object {
14        method energia() {
15            return 5
16        }
17    }
18
19    pepona.comer(alpiste)
20
21    tester.assert(pepona.energia() == 105)
22
23 }
```

Sublime Support Linter



Wollok Game

- a
- b
- c

Wollok Game

FarmVille

FarmVille - Demo

Wollok Game

Sokoban

Sokoban - Demo

Wollok Game

Futuro

- + Tipos de **Juegos**
 - Survival
 - Por turnos
- + Tipos de **Interacciones**
- Features Gráficos
 - Animaciones
 - Fondos infinitos
 - Distintos vistas (lateral, isométrica, etc)

Experiencia en el Aula

- S
- a
- S

Muchas gracias