Pharo 10 and beyond

S. Ducasse
consortium-adm@pharo.org

ESUG 2022

Remember Pharo 90!

- Full redesign of the Spec UI framework (new logic, application, style, GTK back-end)
- NewTools: new playground, new inspector, new debugger (new error infrastructure and emergency debugger)
- New composable completion framework
- General speed up
- Compiler optimisations

Remember Pharo 90!

- Better Refactorings
- Better parser for error recognition
- Comments in Microdown format (Markdown compatible)
- Fast universal FFI (Foreign Function Interface)
- Idle VM + SDL20 and back-end (extended event handling, including trackpad support)
- ARM 64 bits
- Full block closures

Now
Pharo 10

Many Bugs Fixed & Improvements

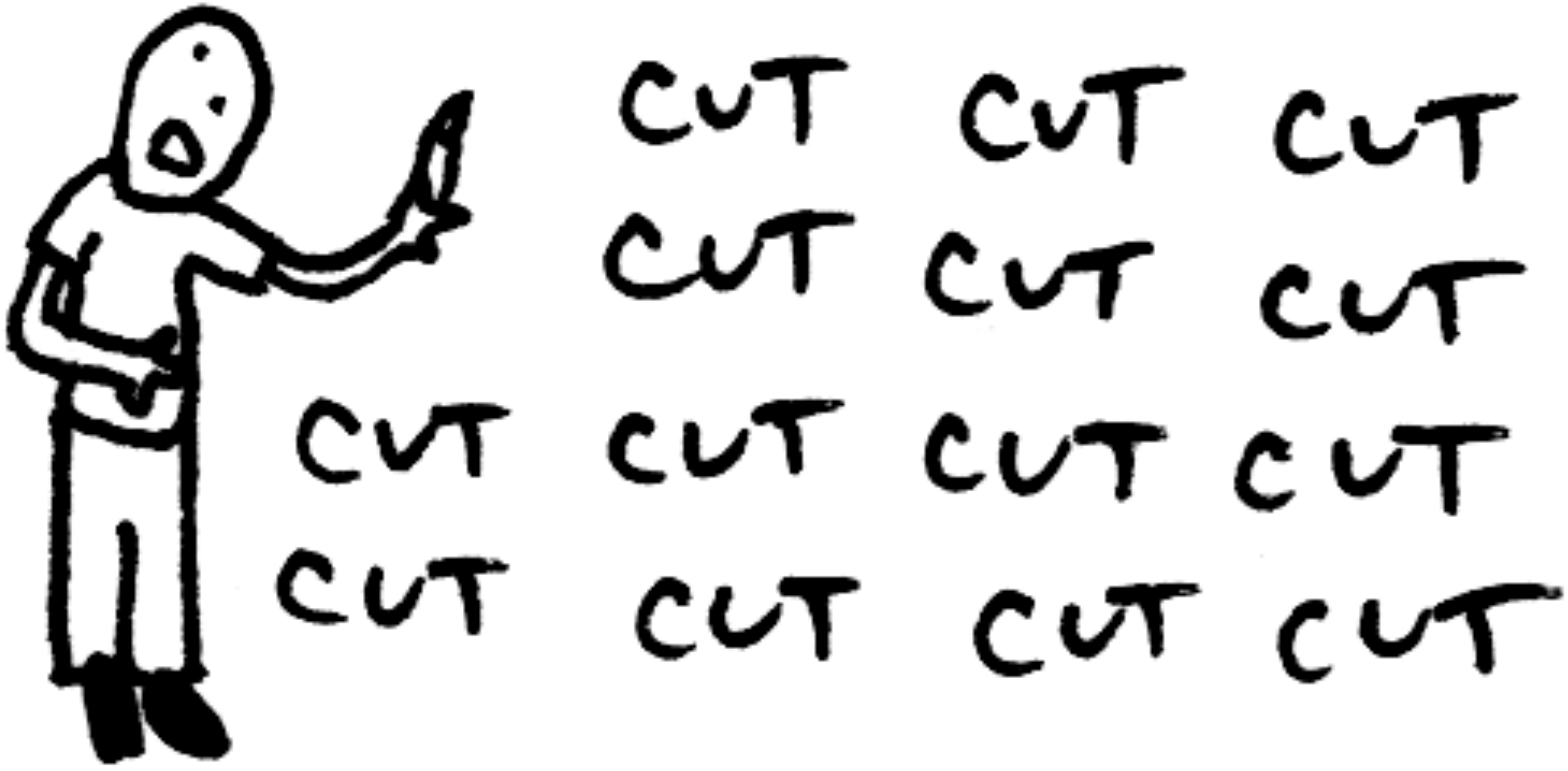
- + 1560 Pull requests
- + 1100 Issues closed
- 85 Different contributors

Many Bugs Fixed & Improvements

We wanted a SMALLER Iteration

(After a large introspection…)

- Improving the development process
- Shorter iteration to release
- **Reduced** set of objectives
- Better “Ready” definition
- Cut the **fat**


CUT CUT CUT
CUT CUT CUT
CUT CUT CUT CUT
CUT CUT CUT CUT

A side joke

“it is better, it has less classes”

said the lame OOP developer

Just smaller is trivial

- Removing code by removing functionalities is easy
- But we want **same** behavior and less code!

The real challenges

Cleaning and consolidating existing functionalities

Supporting users (deprecation)

are more **challenging** and **interesting!**

Pharo10 a.k.a. Cut the FAT

- 10% code reduction!
- Having one *good* instead of three average versions
- - 48 K LOC

Removed old/duplicated code

- Old Tools
- V3 Compiler Support
- Old Blocks & Bytecode
- VM based event handling
- Glamour / GTTools
- Spec1

Spec 2: Main Elements

- Now core is stable!
    - Core & Basic Layouts
    - Basic Presenters
    - Application Support
    - Styles / Themes
- Code Presenter

Spec 2: Extended Features

- Different layouts and composition
- Extended support for *dynamic layouts*
- New dialog building
- Transmissions
- Direct support for Roassal & Cairo
- Multiple backends (GTK / Morphic)
- Spec Tests and Testing Support

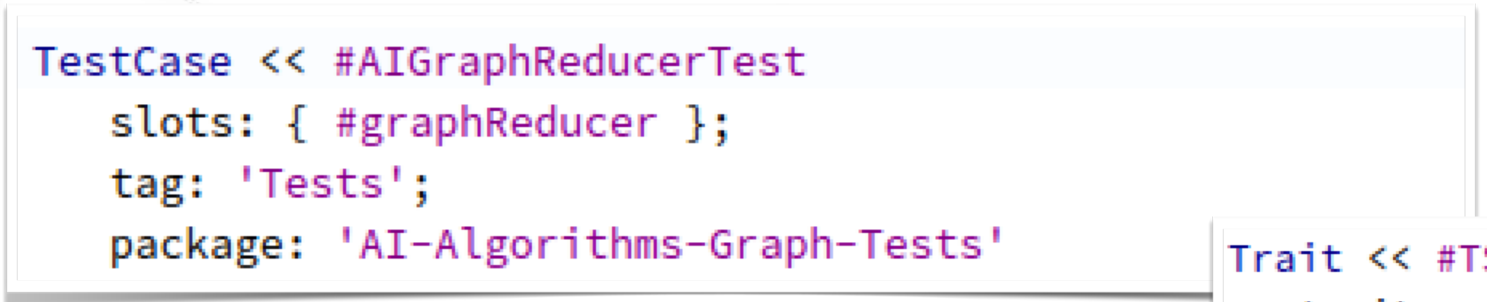
Tooling

- Migrating final tools from Spec1 to Spec2
- Improving existing ones
- Fixing issues and glitches
- Improving Refactorings, Deprecator & Rewriting tools.
- Improving Profilers

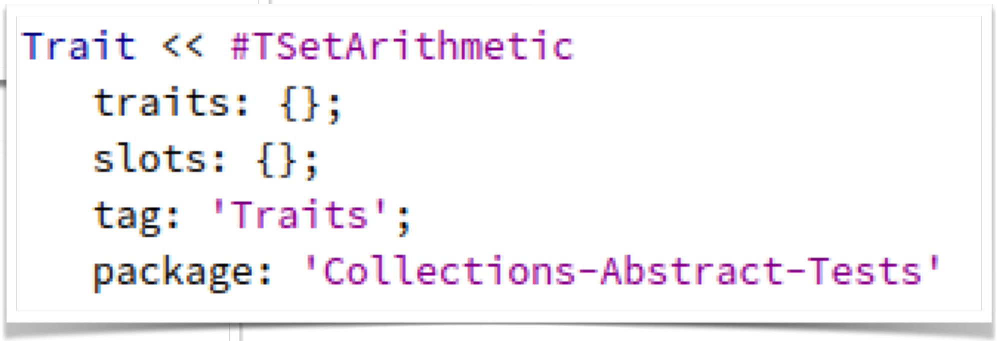
Fluid Class Syntax

- Was sketched and presented in 2017 at ESUG
- Took longer than we wanted but
    - Nice design
    - Scale well with multiple and optional parameters
    - Extensible
    - Clean and *nice* implementation
- Is the default Pharo syntax!

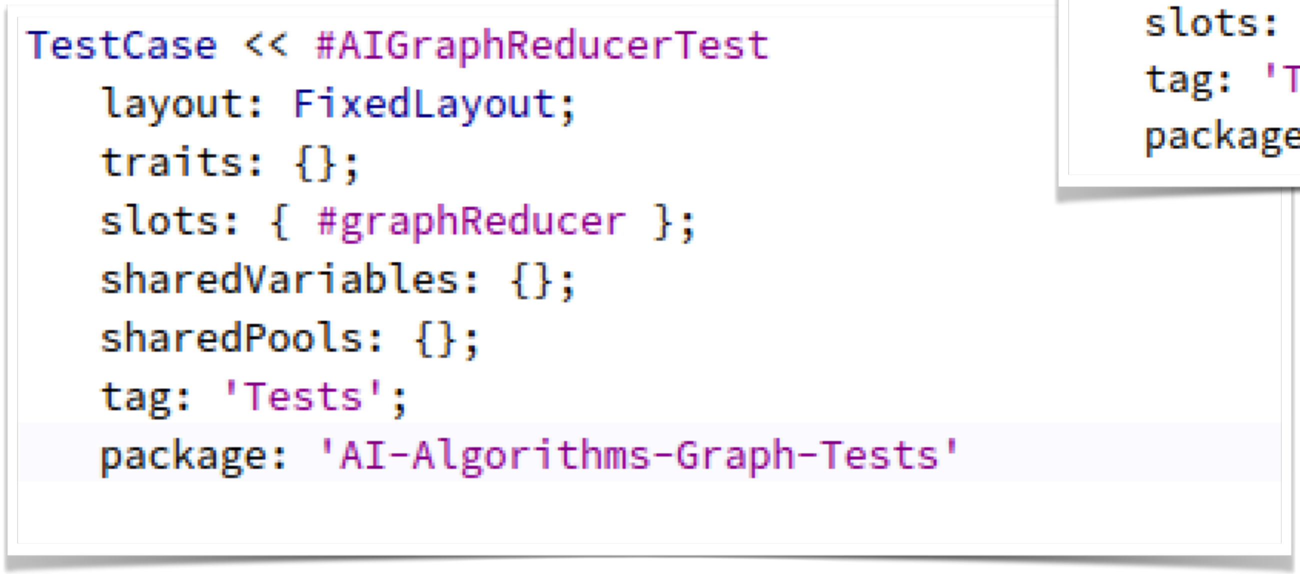
Fluid Class Syntax



```
TestCase << #AIGraphReducerTest
   slots: { #graphReducer };
   tag: 'Tests';
   package: 'AI-Algorithms-Graph-Tests'
```



```
Trait << #TSetArithmetic
   traits: {};
   slots: {};
   tag: 'Traits';
   package: 'Collections-Abstract-Tests'
```



```
TestCase << #AIGraphReducerTest
   layout: FixedLayout;
   traits: {};
   slots: { #graphReducer };
   sharedVariables: {};
   sharedPools: {};
   tag: 'Tests';
   package: 'AI-Algorithms-Graph-Tests'
```

Compiler Improvements

- Unifying objects variables into a single hierarchy
- Improved semantic analysis
  - Use Class and the Environment to lookup the variables
  - Use Variable hierarchy to model variables for name analysis
  - Improved AST Visitor
- Pragma lookup speed-up
- Compiler speed improvements

Refactorings

- New Refactorings
    - Extract setUp method
    - Remove senders of method refactoring
    - Copy package as refactoring
    - Rename package (rename manifest)
    - Merge instance var x in y
    - Move to class side method
    - Create accessors with lazy initialization

Improved Refactorings

- Deprecate method (simple version)
- Deprecate class
- Extract method refactoring
- Replace senders by another
- Rename vars in Traits, Convert temporary to instance variable
- Push up method refactoring
- Add access to pushUp and pushDown refactorings from source code
- Permute parameters when add an argument
- Abstract instance variable

Other Improvements

- Sista Bytecodes w/ Full Block Closures
- Memory management configuration
- Integration with Windows
- Zinc

Pharo 10: VM Improvements (2)

- 3 Operating Systems (OSX / Linux / Windows)
- 3 Architectures (ARM64 / ARM 32 / x86_64)
- Full Linux Packages through OBS
- Better FFI

Pharo 10: VM Improvements

- Sockets
- Clean up old code
- GC Improvements
- Logging
- Stability, Speed
- Updated Dependencies

Pharo 10

- Questions?

Pharo 11's possible points

P11 possible points [Language]

- Ephemerons
  - Using them in the image
  - Replacing Weak / Finalization mechanisms
- Concurrency
  - Cleaning up the concurrency mechanisms we have.
  - Make the image to use higher level mechanisms.

P11 possible points [Compiler]

- Clean Blocks
  - Sharing them
  - Full tool support
- Compiler Improvements
  - New Optimizations
  - Better Plugin support
  - ...

P11 possible points [UI]

- Multi Windows
- HDPI
- Bloc in preview
    - Spec Backend
    - Performance

P11 possible points [Modularity]

- Pakbot
  - Dependency management
  - Projects
- Modularization
  - Minimal Images
  - External Projects
  - Better Baselines

Not optional P11 VM points :)

- **VM**
  - **Memory Management**
  - **PermSpace**
- **New Image format**
  - **Meta-data**
- RISC-V

Thanks!!!

pharo.org

pharo-project/pharo

consortium-
adm@pharo.org

consortium

discord.gg/QewZMZa