

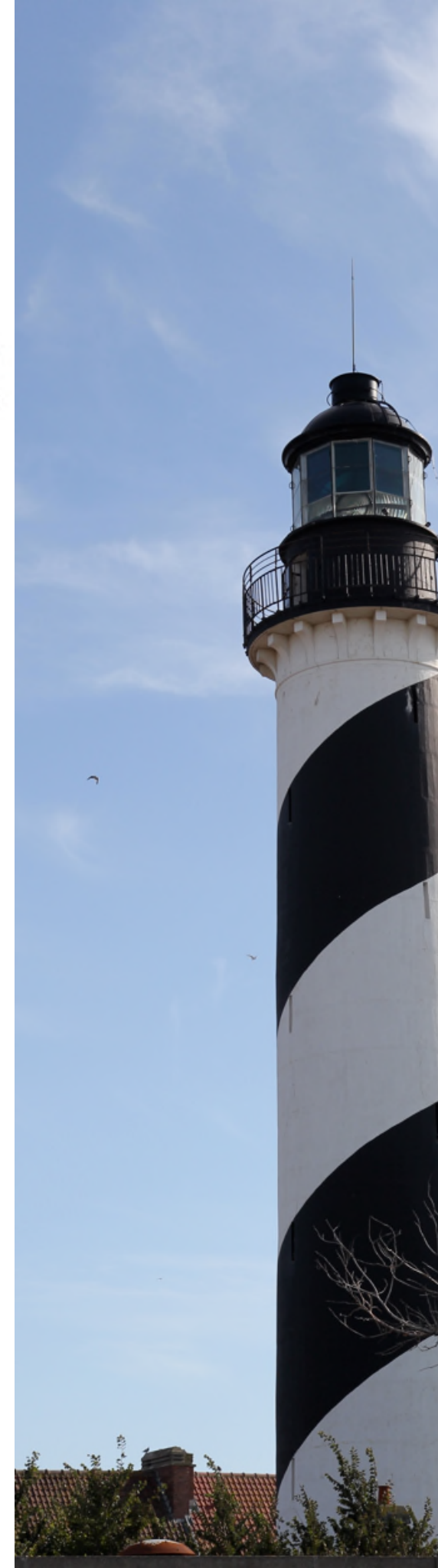
Pharo 14

<http://www.pharo.org>
stephaneducasse@



ESUG 2026





A single word

HUGE amount of work

HUGE

Multiple parts

- Process CI
- Pharo Infrastructure (cmd line, interruption..)
- Debugging
- Compiler
- Spec
- Tools
- and more ...

[All the crazyness we do]

- Debugging in gdb using Smalltalk garbageCollect as C breakpoints!
- Debugging mini images that collapse on the server without ui nor tools
- Broken tests on the server while working locally :(
- Fighting with libGit broken versionning and absence of backward compatibility
- And many more :(

Outline

- **Process CI**
- Pharo Infrastructure (cmd line, interruption..)
- Debugging
- Compiler
- Spec
- Tools
- and more ...

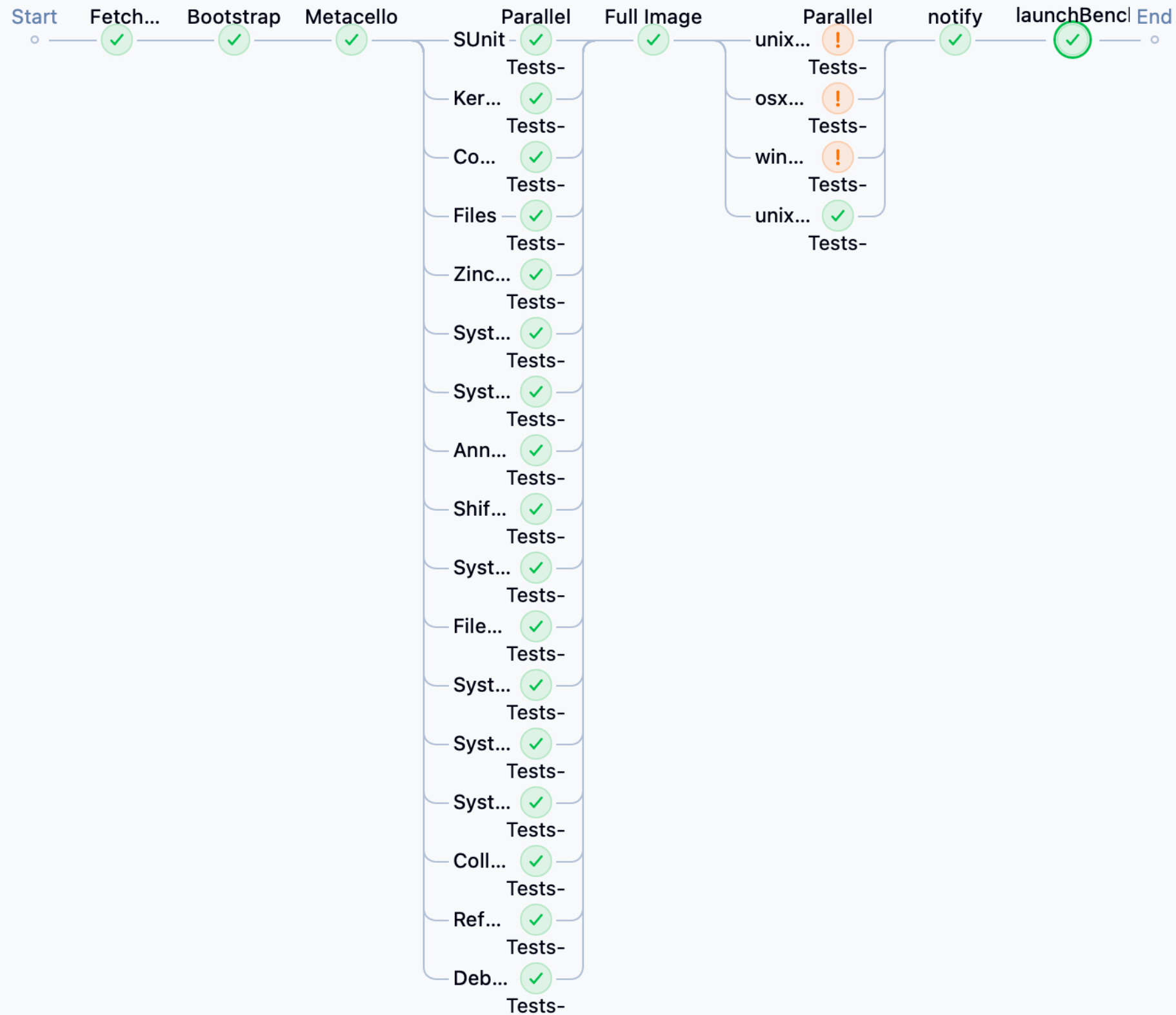
Fighting the Antagonism

- Image-loaded code vs. baseline of code to be loaded
- When subsystem is added in the image, it may introduce hidden dependencies
- Got some “ugly” tests verify dependencies
- But static and tedious when breaking

CI supported modularity

- Modularity by construction
- Load the baselines to validate them and run their tests
- Mid term goals: Validation of all baselines

CI supported modularity



Modularity by construction

- strict mode to load baseline
- No undeclared in loaded code
- Our goal is to validate all Pharo
- system be prepared
- No undeclared in your code



Outline

- Process CI
- **Pharo Infrastructure (cmd line, interruption..)**
- Debugging
- Compiler
- Spec
- Tools
- and more ...

Better interruption handling

- Remember 2025 talk of Guille Polito on memory

- The Gist of it!

when there is an endless loop, the system in urgent mode cannot allocate more objects on heavy stress and felt in “move stack pages to memory”

But there is no memory !!!! so... collapse in GC spinning wheel of death

Better interruption handling

Stack page size can be set at VM startup

Exception to be able to react to memory exhaustion

Already there but not by default

Old Command line infrastructure

Was clunky, clumpy and a bit rotten too :)

Two different handlers

Not regular

Not optional

Modernized command line infrastructure

- Unique command line handler at image startup
- All commands are now written in Clap

Unique command line handler

- Used at image startup
- ClapCommandLineHandler runs clap applications

Default low-level entry

```
perform [--help] [--save] [--no-quit] [--as-array  
<array-argument>] [<global>] [<messageSelector>]  
[<arguments>]
```

```
Pharo P14.image perform Point new  
(nil@nil)
```

You can add your own

Outline

- Process CI
- Pharo Infrastructure (cmd line, interruption..)
- **Compiler**
- Debugging
- Spec
- Tools
- and more ...

Compiler cleanup

Paving the way for modules

- Modules for real: [Guille' presentation]
- We are (mentally) far from modularity
 - Yes creating a symbol as a side effect on the SymbolTable [Guille' presentation]
- Compiler should be **optional**
- Compilers versus **The** Compiler

A simple module scenario

Pharo bootstrap

running in P10

get impacted by any VM and compiler changes

What if we could run the old compiler in a given module and execute the tests (yes we different SUnit version) of the bootstrap

Can we have (another) compiler compiling the compiler?

Compiler challenges

- There is no such One Compiler!
- Hidden global
- Cannot compile IR methods without Death Recursion
- Avoid $\wedge \% \$ \& * \% (* \& \wedge \$ \%$ coupled in a strange way parameters
- Layer violations
- 10 different ways to add a method in a class!

the Ugly requestor

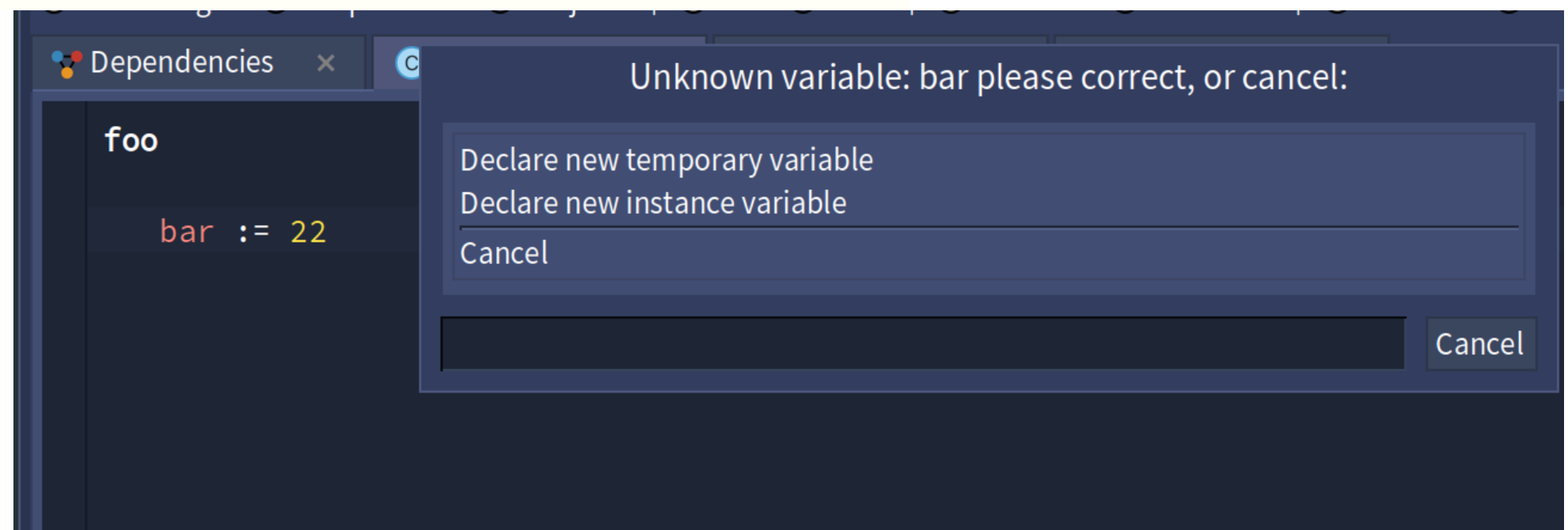
- Yes the compiler had the user interaction logic

foo

OrderedCollection new

x: 12 ; y *End of statement expected ->*OrderedCollection new

x: 12 ; y + 3+ 3



Compiler revisited

- Split responsibility: Evaluation, InteractiveAPI (requestor), and Compilation
- Cleaned API
- Cleaned clients
- Layering in place
- Compilation returns **a compilation result** (source, metadata, and compile method)
- **MethodInstallers** *silent*, announcing, traits, modules...

No omnipresent compiler!

Long term goal: Shipping Pharo Apps without a compiler!

- Debug without source
- **Debug without the compiler**
- Have different compilation semantics (plugins)
- How the debugger can debug when it has no clue about the plugin used to compile the method?
- Try and fail

DebugInfo to the rescue!

- Dwarf like metadata for debugger
- API to
 - identify key messages for debugging
 - support backwards approach
 - support forward compiler-less debugging
- Cleaning Context, CompiledMethod, InstructionStream API (oh boy!)

Where to store the metadata?

140,000 methods

2k per method does not fit in the image

Store the metadata in the sources file?

Well did you ever look at the source management code...



Source challenges

- Loading image logic not symmetrical to saving
- Multiple loading code logics
- Multiple saving code logics

Painful to debug... and code without source :(

Only one is more

- MASSIVE cleanup of logic
- CompiledMethods do not need to know “the” compiler!
- One way to load
- One way to save code
- Versionned format! (duh!)
- Open doors for the future

Modernized source next steps

- Possibility to plug
 - alternate approaches (e.g., database,...)
 - Package metadata
- Support for debugger without compiler

Outline

- Process CI
- Pharo Infrastructure (cmd line, interruption..)
- Compiler
- **Debugging**
- Spec
- Tools
- and more ...

Challenges of mini images debugging

How to debug mini images?

- No UI
- Not all libraries loaded

```
Error: CRITICAL: Data ingestion pipeline failed in worker thread.  
UndefinedObject>>DoIt  
OpalEvaluator>>evaluate:  
[ ] in ClapCodeEvaluator>>evaluateExpression  
FullBlockClosure(BlockClosure)>>on:do:  
ClapCodeEvaluator>>evaluateExpression  
ClapCodeEvaluator>>evaluate  
ClapCodeEvaluator>>execute  
ClapCodeEvaluator(ClapApplication)>>executeOrPrintHelp  
[ ] in ClapContext>>executeToExit:  
FullBlockClosure(BlockClosure)>>on:do:  
ClapContext>>executeToExit:
```

CmdLine debugger API

Prompt based textual stdin/out debugger :)

```
----- Debug phase started -----  
For help, type "help"  
(debug) help  
===== available commands =====  
quit ( or 'q' ) : kills the debugger  
step ( or 's' ) : execute next stepInto  
printCode ( or 'pc' ) : print current method code  
print ( or 'p' ) <expression> : evaluate the expression  
history ( or 'hist' ) : Give the history of last commands  
over ( or 'o' ) : execute next stepOver  
printStack ( or 'ps' ) : print the current Stack  
help ( or 'h' ) : print this help  
(debug) step  
Stepped to: [ ] in UndefinedObject>>DoIt(PC: 19)  
(debug) step  
Stepped to: [ ] in UndefinedObject>>DoIt(PC: 20)  
(debug) printCode  
[ --->1+1<--- ]  
(debug) !!  
[ --->1+1<--- ]
```

Outline

- Process CI
- Pharo Infrastructure (cmd line, interruption..)
- Compiler
- Debugging
- **Spec**
- Tools
- and more ...

Spec news

- [alpha] Code editor / Code presenter work
- Refactoring command migration
- Morphic layout fixes
- Better List/Tree filtering behavior
- Dialog/window/modal behavior
- Action/command cleanup

Pharo 14 Tools

Migrating to Spec

Setting Browser

Finder

MethodBrowser

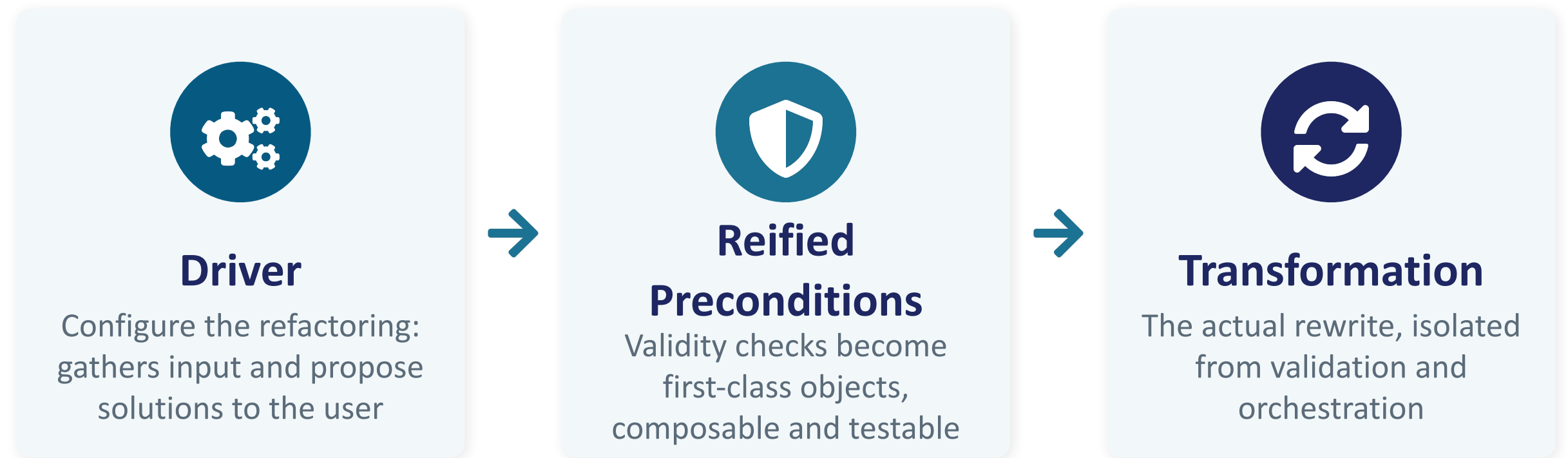
File Browser/Finder/Transcript improvements

StPulse to replace Spotter

Smoother Refactorings

REFACTORING ENGINE

A Better Flow for Refactorings



The result: each step owns its own logic — clearer code, safer transformations, and refactorings that are easier to extend.

New method browser

- Spec-based (for Toplo migration)
- Support for refactorings
- Test execution
- H/V layout views
- Tabs

New Method Browser: Browsing, Rebuilt

The screenshot shows a window titled "Senders of createPath: [48]" with a toolbar at the top containing icons for "Current image", "Browse", "References", "Senders", "Implementors", "Versions", "Commit history", "Copy to clipboard", "Reuse window", "Run Tests", and "Flip".

The left pane displays a table of senders:

Location	Selector	Package
AthensSurfaceExamples class (examp	example3stroke	[Athens-Examples]
AthensSurfaceExamples class (examp	example3	[Athens-Examples]
AthensSurfaceExamples class (examp	example10	[Athens-Examples]
AthensSurfaceExamples class (examp	example5	[Athens-Examples]
AthensTiger (converting)	readParts	[Athens-Examples]
BracketMorph (*Athens-Morphic)	createClosedPolygonPathFrom:on:	[Athens-Morphic]
EllipseMorph (*Athens-Morphic)	asAthensShapeOn:	[Athens-Morphic]
HiSimpleRenderer (rendering)	athensPathForDescendingLink:on:	[Hiedra]
HiSimpleRenderer (rendering)	athensPathForNodeOn:	[Hiedra]
HiSimpleRenderer (rendering)	athensPathForAscendingLink:on:	[Hiedra]
PolygonMorph (*Athens-Morphic)	drawArrowOnAthensCanvas:at:from:	[Athens-Morphic]
PolygonMorph (*Athens-Morphic)	asAthensCurvedOpenPathOn:	[Athens-Morphic]
PolygonMorph (*Athens-Morphic)	asAthensCurvedPathOn:	[Athens-Morphic]
PolygonMorph (*Athens-Morphic)	asAthensLinePathOn:	[Athens-Morphic]
RSAthensRenderer (creating path)	getOrCreatePathOf:	[Roassal]
RubAdornmentDisplayer (drawing)	drawOnAthensCanvas:	[Rubric]
RubSegmentMorph (drawing)	drawOnAthensCanvas:	[Rubric]
RubSegmentMorph (drawing)	drawBorderOnAthensCanvas:usingEr	[Rubric]
RubUnderlinedSegmentMorph (dra	drawUnderlineOnAthensCanvas:	[Rubric]
RubSurfaceSelectionShape (drawing)	drawOnAthensCanvas:	[Rubric]
RubTextSegmentIconDisplayer (draw	drawOnAthensCanvas:	[Rubric]
SDL2AthensDrawingExample (updati	redraw	[OSWindow-SDL2-Examples]
SDL2TouchExample (drawing)	drawMoves	[OSWindow-SDL2-Examples]
SDL2TouchExample (initialization)	initializeFromSurface	[OSWindow-SDL2-Examples]
SDL2TouchGestureExample (drawir	generateCircleOfSize:	[OSWindow-SDL2-Examples]
SDL2TouchGestureExample2 (draw	generateCircleOfSize:	[OSWindow-SDL2-Examples]
SDL2TouchGestureExample2 (draw	displayAbortMessage	[OSWindow-SDL2-Examples]
SpAthensAdapterTest (accessing)	circle:	[Spec2-Backend-Tests]

The right pane shows the source code for the selected sender:

```
1 renderOn: canvas
2
3 | path pathBlock |
4
5 canvas surface clear: Color black.
6 pathBlock := [ :builder |
7     builder
8         absolute;
9         moveTo: pt1;
10        curveVia: pt2 and: pt3 to: pt4
11    ].
12
13
14 "First , we draw the curve using Cairo"
15 (canvas setStrokePaint: Color green).
16 canvas drawShape: (canvas createPath: pathBlock).
17
18 "draw a polygon, connecting control points"
19 (canvas setStrokePaint: Color blue) width: 0.5.
20 canvas drawShape: (canvas createPath: [ :builder |
21     builder
22         absolute;
23         moveTo: pt1;
24         lineTo: pt2;
25         lineTo: pt3;
26         lineTo: pt4
27     ]).
28
29
30
31 canvas setPaint: Color red.
32
```

Complishion on steriods

- **Package dependencies** to better choices
- **Coding history** mixed with your **last selections** :)
- [preview] Typo tolerance
- [See Complishion talk]

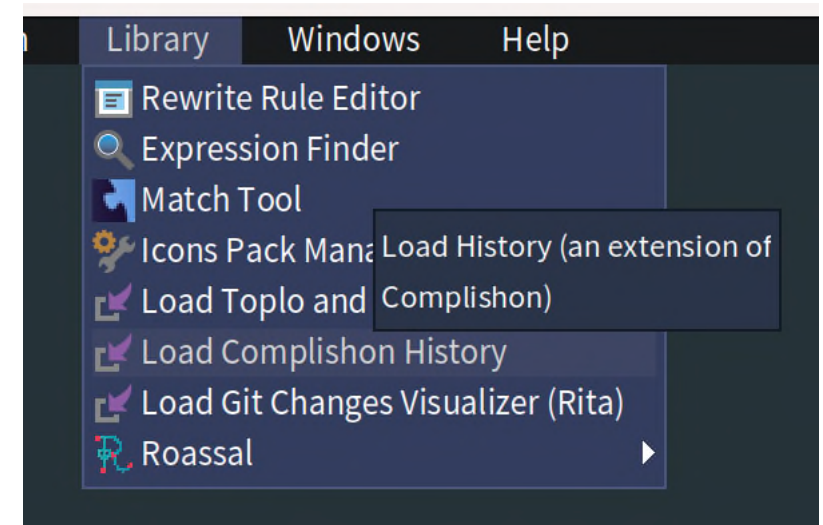
Past activities

Optional but super cool

Take your **recent class/method addition/removal**

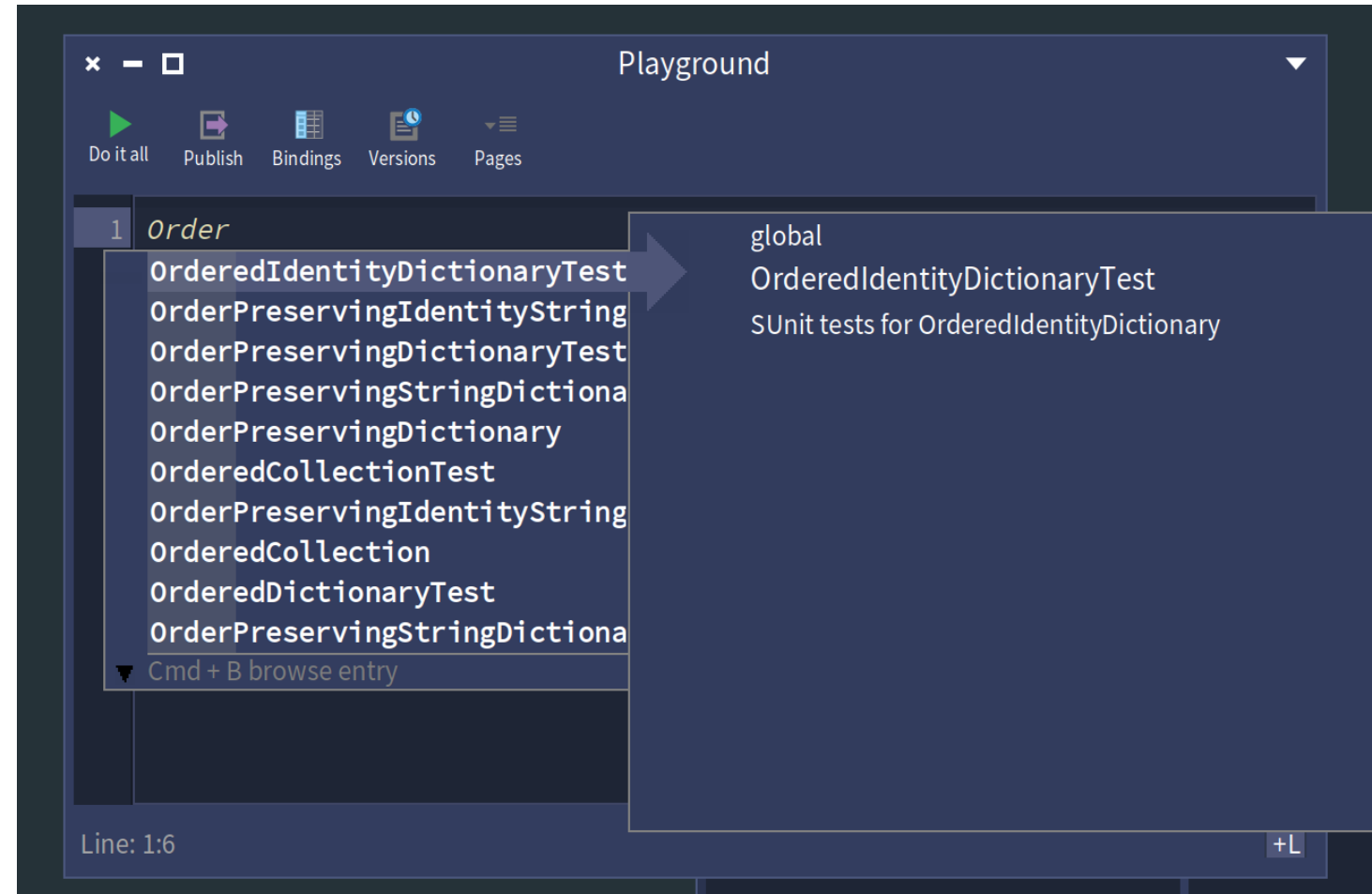
Take your **recent selections** in the menu

Use a recently use window



Without With

- Selecting OrderedCollection



Iceberg

- New Spec merge browser replaces
- optional command-line Git backend
- Metacello/direct Git URL fixes
- Repository lifecycle and libgit resource cleanup
- Repository browser UX improvements
- dirty-package, diff, snapshot, and history performance work
- UIManager and deprecated API cleanup
- LibGit chase game: version numbering is not backward compatible

Tests

- TestRunner object
 - responsible of running the tests (shuffling, report generation)
 - To be improved and more used
- Pass on HD report
- Support server test local machine running
 - Sorting then shuffling with the same seed

VM side

- Merge several feature branches
- Starting to work on VM migration to Pharo 13
- Change of Clang versions (from 9 to 21) - Ouch
- OBS revisited

VM summary

- Frame unification and stack management improvements, delivering substantial interpreter and execution engine cleanups.
- RISC-V JIT support, expanding platform coverage and future-proofing VM development.
- New memory management capabilities, including direct old-space allocation and numerous GC and allocation improvements

VM summary

- Extensive compiler and Slang/C translation work, many correctness fixes, new tests, and improved inspection tools.
- Platform and build modernization, including SDL updates, improved CI, cross-build support, and updated third-party dependencies.
- Large-scale cleanup and refactoring efforts across the VM, interpreter, Cogit, PICs, and infrastructure.

Enhancements

- Clean blocks and debugger/evaluation behavior
- ClassBuilder, trait, and superclass correctness
- Morphic/UISManager dependency reduction
- Calypso, FastTable, text, graphics, Zinc, and Zodiac fixes
- Many more ...

Backward compatible

We are **REALLY** concerned
about backward compatibility

Most of the work is backward
compatible

Toplo/Bloc on Preview

Missed our target to have Toplo in P14 :(

But we made a lot of progress

See the presentation about Bloc/Toplo

Consortium news



- Spesenfuchs increased its level! Thank you
- Qualitative and Quantitative Data Fruits is platinumium



- We are honored, committed and feel responsible of delivering for you

Consortium news

- New support process
 - Systematically report support consumption
- Evaluating support for migration from proprietary Smalltalks

PharO

