

Introducing smalltalk-dev-plugin

AI-Driven Development Toolkit for Pharo Smalltalk

Masashi Umezawa

<https://github.com/mumez/smalltalk-dev-plugin>

What is smalltalk-dev-plugin?

A comprehensive toolkit that bridges **AI code editors** and the **Pharo Smalltalk** environment.

The AI Agent Problem for Smalltalk

Using AI coding agents with Smalltalk doesn't work well out of the box:

- **No knowledge of Pharo** — AI doesn't know how to interact with the environment
- **Unknown project structure** — Typical Tonel/package layout is unfamiliar
- **No testing/debugging skills** — Can't run tests or diagnose errors
- **No library access** — Can't browse existing classes or methods
- **Poor coding style** — Doesn't know Smalltalk idioms and conventions

This is a major barrier for developers who want AI assistance in Smalltalk.

Enter smalltalk-dev-plugin

We gave AI the complete skillset for Smalltalk development:

Challenge	Solution
Environment interaction	MCP tools for Pharo communication
Project structure	<code>/st:setup-project</code> with templates
Testing & debugging	<code>/st:test</code> , <code>/st:eval</code> , debugger skill
Library navigation	Finder skills, code search tools
Code quality	Validator, linter, commenter agent

Result: AI becomes a reliable Smalltalk development partner!

"AI editor as the source of truth"

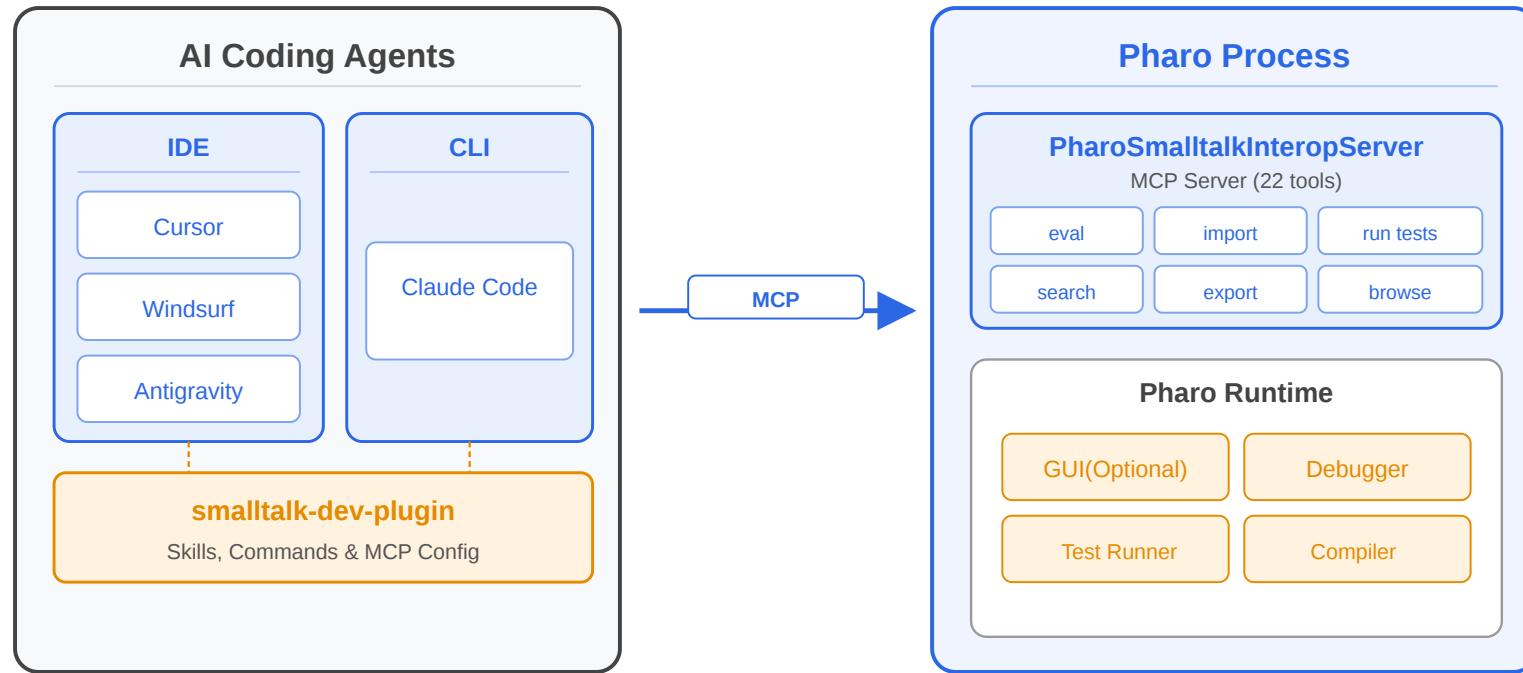
Edit Tonel files in the AI editor, import to Pharo, test, and iterate.

At a Glance

- 9 slash commands
- 4 specialized AI skills
- 27+ MCP tools for Pharo interaction
- Automated code quality checks and documentation generation

Architecture

6



- Existing AI agents talk to Pharo via MCP — no custom IDE needed
- Works with headless Pharo (CI, remote agents like Devin)

Supported AI Agents

7

Agent	Support Level
Claude Code	Full support (primary)
Cursor	Simplified setup via script
Windsurf	Simplified setup via script
Antigravity	Simplified setup via script

The rest of this presentation focuses on the Claude Code version.

Installation

Installation (1) — Install the Plugin

```
# Add marketplace from GitHub  
claude plugin marketplace add mumez/smalltalk-dev-plugin  
  
# Install the plugin  
claude plugin install smalltalk-dev
```

Installation (2) — Set Up Pharo Side

10

Install the interop server in Pharo:

```
Metacello new  
baseline: 'PharoSmalltalkInteropServer';  
repository: 'github://mumez/PharoSmalltalkInteropServer:master/src';  
load.
```

Start the server:

```
SisServer current start.
```

That's it — you're ready to go.

Setup scripts are provided for Cursor, Windsurf, and Antigravity:

```
./extra/setup-cursor.sh [target-directory]
./extra/setup-windsurf.sh [target-directory]
./extra/setup-antigravity.sh [target-directory]
```

Limitations

- Commands use `st-` prefix instead of `st:`
- Some features may differ depending on the agent
- Pharo-side setup is the same as for Claude Code

Basic Usage

/st:buddy — Main Entry Point

13

/st:buddy is a friendly assistant that routes to the right tool.

Just describe what you want in natural language:

```
/st:buddy I want to create a Person class with name and age
```

The assistant will:

1. Load the appropriate skill automatically
2. Create the Tonel files
3. Guide you through linting, importing, and testing

/st:buddy — Intelligent Routing

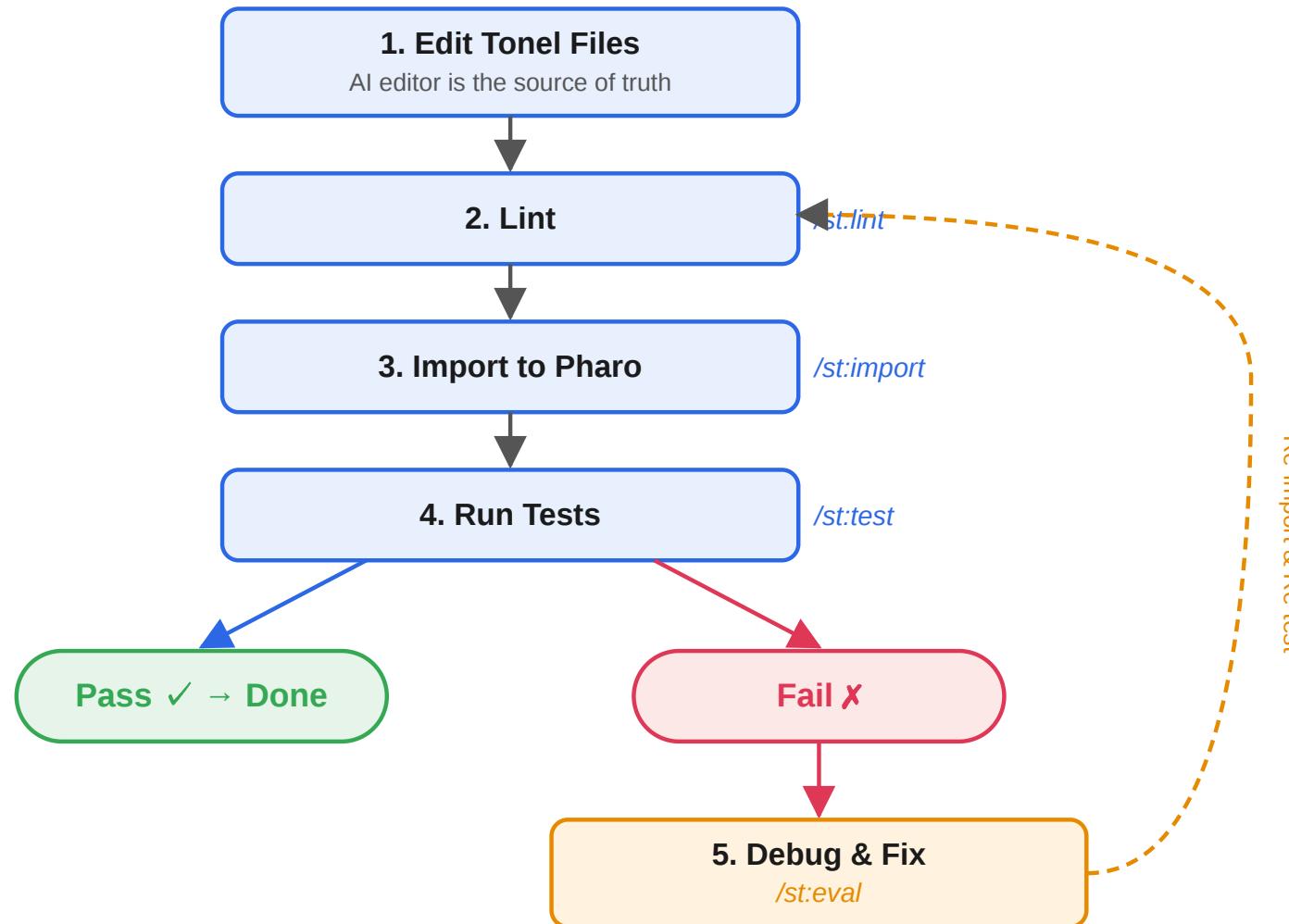
14

Your Intent	Loaded Skill
"Create...", "Add..."	smalltalk-developer
"Error...", "Test failed..."	smalltalk-debugger
"How do I use...?"	smalltalk-usage-finder
"Who implements...?"	smalltalk-implementation-finder

You don't need to remember which tool to use — just talk to `/st:buddy`.

Development Workflow

15



Components

smalltalk-developer

The core development skill. Knows Tonel format, package structure, and the full Edit → Lint → Import → Test workflow.

smalltalk-debugger

Systematic debugging specialist. Diagnoses errors, runs incremental code execution with `/st:eval`, and guides you to the fix.

smalltalk-usage-finder

Discovers how classes and methods are used. Finds examples, analyzes usage patterns, and helps you understand unfamiliar APIs.

smalltalk-implementation-finder

Finds method implementations across class hierarchies. Useful for learning coding idioms and assessing refactoring impact.

Slash Commands

19

Command	Purpose
/st:buddy	Friendly assistant (main entry)
/st:init	Initialize development session
/st:setup-project	Create project boilerplate
/st:eval	Execute Smalltalk expressions
/st:import	Import Tonel packages to Pharo
/st:export	Export packages from Pharo
/st:test	Run SUnit tests
/st:lint	Check code quality
/st:validate	Validate Tonel syntax

Sub-Agent: @smalltalk-commenter

20

Automatically suggests **CRC-style class comments** for undocumented classes.

- Triggered after file edits via hooks
- Non-intrusive — suggests, doesn't force
- Analyzes complexity and prioritizes classes that need documentation

Two MCP servers power the integration behind the scenes:

pharo-interop (22 tools)

Pharo communication — eval, import/export, tests, code navigation, etc.

smalltalk-validator (5 tools)

Tonel validation and linting.

Note: You rarely need to interact with these directly.

`/st:buddy` handles orchestration for you.

Live Examples

Example 1 — Starting from Scratch

23

A simple use case: starting from an empty directory as a Smalltalk beginner.

```
/st:buddy I'm a Smalltalk beginner. I want to create a Money class  
that can perform arithmetic operations across different currency units.  
Please help me from the project initialization.
```

Example 1 — What Happens Next

```
Claude Code — Example 1: Money Class

e-money/src/SampleMoney/ExchangeRateProvider.class.st"
  "success": true,
  "file_path": "/home/une/git/st-dev-sample-money/src/SampleMoney/ExchangeRateProvider.class.st",
  "t3: lines (ctrl+o to expand)

● plugin:smalltalk-dev:smalltalk-validator - lint_tonal_smalltalk_from_file (MCP)(file_path:
  "/home/une/git/st-dev-sample-money/src/SampleMoney-Tests/MoneyTest.class.st")

  "success": true,
  "file_path": "/home/une/git/st-dev-sample-money/src/SampleMoney-Tests/MoneyTest.class.st",
  "t3: lines (ctrl+o to expand)

● plugin:smalltalk-dev:smalltalk-validator - lint_tonal_smalltalk_from_file (MCP)(file_path: "/home/une/git/st-dev-sample-money/src/BaselineOfSampleMoney/BaselineOfSampleMoney.class.st")

  "success": true,
  "file_path": "/home/une/git/st-dev-sample-money/src/BaselineOfSampleMoney/BaselineOfSampleMoney.class.st",
  "t3: lines (ctrl+o to expand)

  Scanning... (in 20s = 11.2k tokens)

>| accept edits on (shift+tab to cycle) - esc to interrupt
```

- Video
- Generated source
- Demo

Example 2 — Graph Algorithms

25

A more complex use case for someone familiar with Smalltalk.

```
/st:buddy I want to create GrNode and GrArc to represent directed graphs  
and solve shortest path problems. Nodes have a name, arcs have a score.  
Let's start this as a GraphGear project.
```

Example 2 — What Happens Next

Claude Code — Example 2: Graph and Shortest Path

```
1 Class {
2     #name : 'GrArc',
3     #superclass : 'Object',
4     #instVars : [
5         'source',
6         'target',
7         'score'
8     ],
9     #category : 'GraphGear'
10 }
```

→ 51 lines (ctrl+e to expand)

Now let me implement GrGraph with Dijkstra's algorithm.

```
Write(src/graphGear/GrGraph.st)
```

Implementing GrGraph with Dijkstra. (to 260 - 1.14k tokens)

- ✓ Implement GrNode class
- ✓ Implement GrArc class
- ✗ Implement GrGraph class with Dijkstra
- Implement BaselineOfGraphGear
- Implement test classes
- Lint, import, and run tests
- ✓ Create project directories and package.st files

accept edits on (shift+tab to cycle) - esc to interrupt

- Video
- Generated source
- Demo

Example 3 — GUI Programming

Building a GUI application with Spec2.

```
/st:buddy I want to make a simple to-do list using the Spec2 framework.  
Include a checkbox for each item, plus an input field  
with Add/Remove buttons at the bottom.  
Only checked items can be removed. Start developing.
```

Example 3 — What Happens Next

```
Claude Code -- Example 3: Spec2 to-do list

90 + | filteredItems presenters filterText |
91 + filterText := filterField text asLowercase trimBoth.
92 + filteredItems := filterText
93 +           ifEmpty: [ items ]
94 +           ifNotEmpty: [
95 +             items select: [ :item |
96 +               item text asLowercase includesSubstring: filterText ] ].
97 + presenters := filteredItems collect: [ :item |
98 +   TodoListPresenter new
99 +     model: item;
100 +     yourself ].
```

Now let me list and import the updated code.

```
#login:smalltalk-dev:smalltalk-validator - lint_tonel_smalltalk_from_file (MCB)(file_path:
  "/home/ume/git/st-dev-sample-gui/src/TodoList/TodoListPresenter.class.st")
```

```
{ "success": true,
  "file_path": "/home/ume/git/st-dev-sample-gui/src/TodoList/TodoListPresenter.class.st",
  "line_numbers": 100}
```

#Running. (18s, 17246 tokens, 1311Kb)

I tip: Try setting environment variable COLORTERM=truecolor for richer colors

```
accept edits on (shift+tab to cycle) - use to interrupt
```

- Video
- Generated source
- Demo

smalltalk-dev-plugin makes Pharo Smalltalk development with AI assistants practical and productive.

Just type `/st:buddy` and start building.

Feedback and contributions are welcome!

<https://github.com/mumez/smalltalk-dev-plugin>