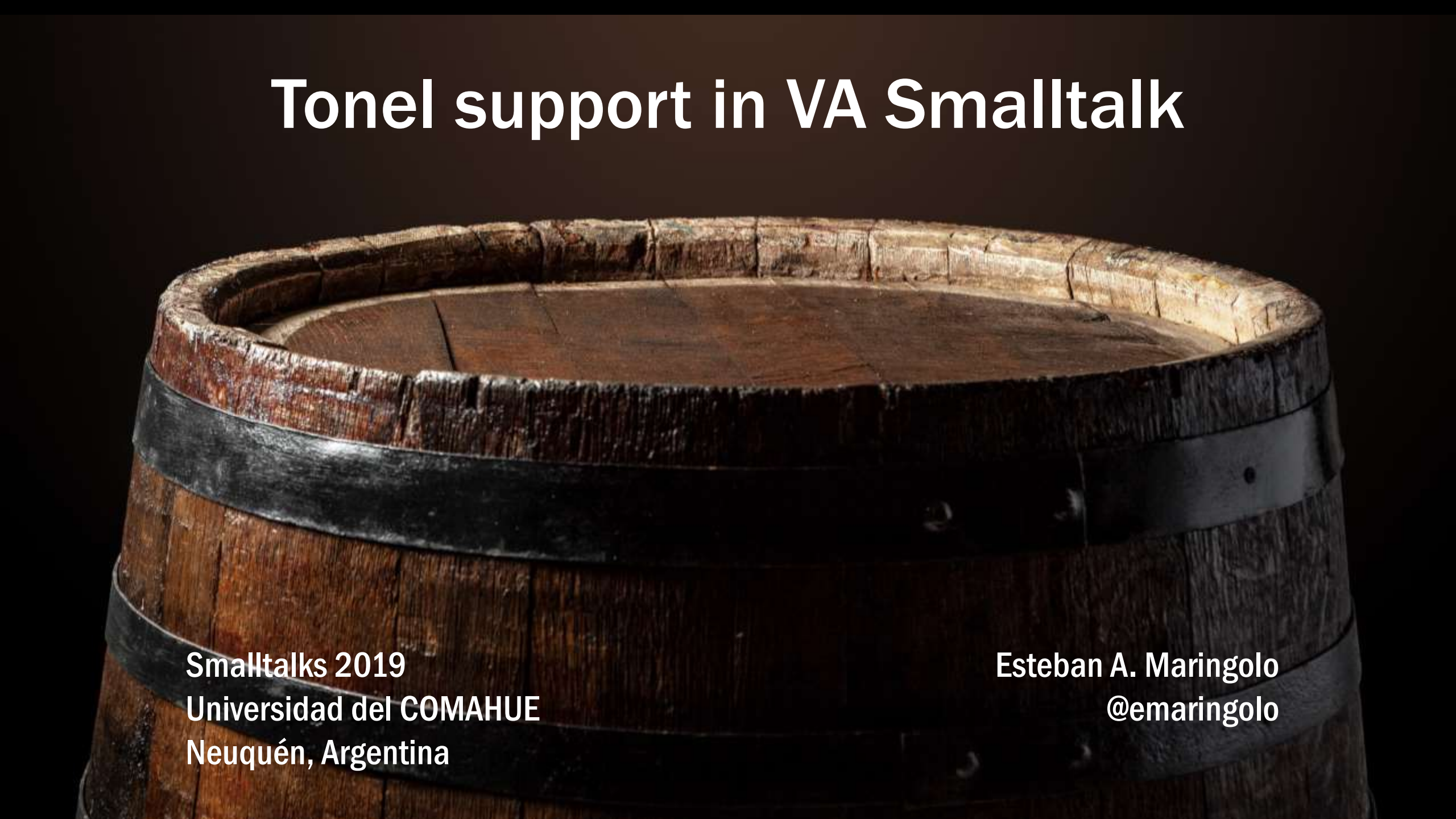
Tonel support in VA Smalltalk



Smalltalks 2019
Universidad del COMAHUE
Neuquén, Argentina

Esteban A. Maringolo
@emaringolo

Objectives

- Exchange code with other Smalltalk dialects supporting Tonel
- Enable file-based VCS for VAST Applications
- Source backup of applications in ENVY Library

Quick recap of Tonel

- Plaintext based source code persistence
- Uses its own syntax and parser
  - Sections with STON syntax
- Support for these types:
  - Package
  - Class
  - Trait
  - Extension

```
[comment] type {
  typeDefinition
}
{ methodMetadata } method [
methodBody ]
```

```
Class {
     #name : ‘ClassName',
     #superclass : ‘Object',
     #instVars : [‘iv1',‘iv2’],
     #classVars : ['CV1'],
     #pools : [‘SomePool'],
     #classInstVars : [],
     #category : ‘Category name'
}

{ #category : 'Accessing’ }
  ClassName >> accessor [^iv1 ifNil: [‘foo’]]
```

But...

- Designed within Pharo
- With Pharo needs in mind
- Without considering other dialects

VAST Specific features

- Multiple method categories
- Public/Private methods
- Application prerequisites
- Applications/Subapplications hierarchy
  - Subapplication conditions (shadows)
- SharedPool declaration pragmas

Semantic equivalences

VA Smalltalk

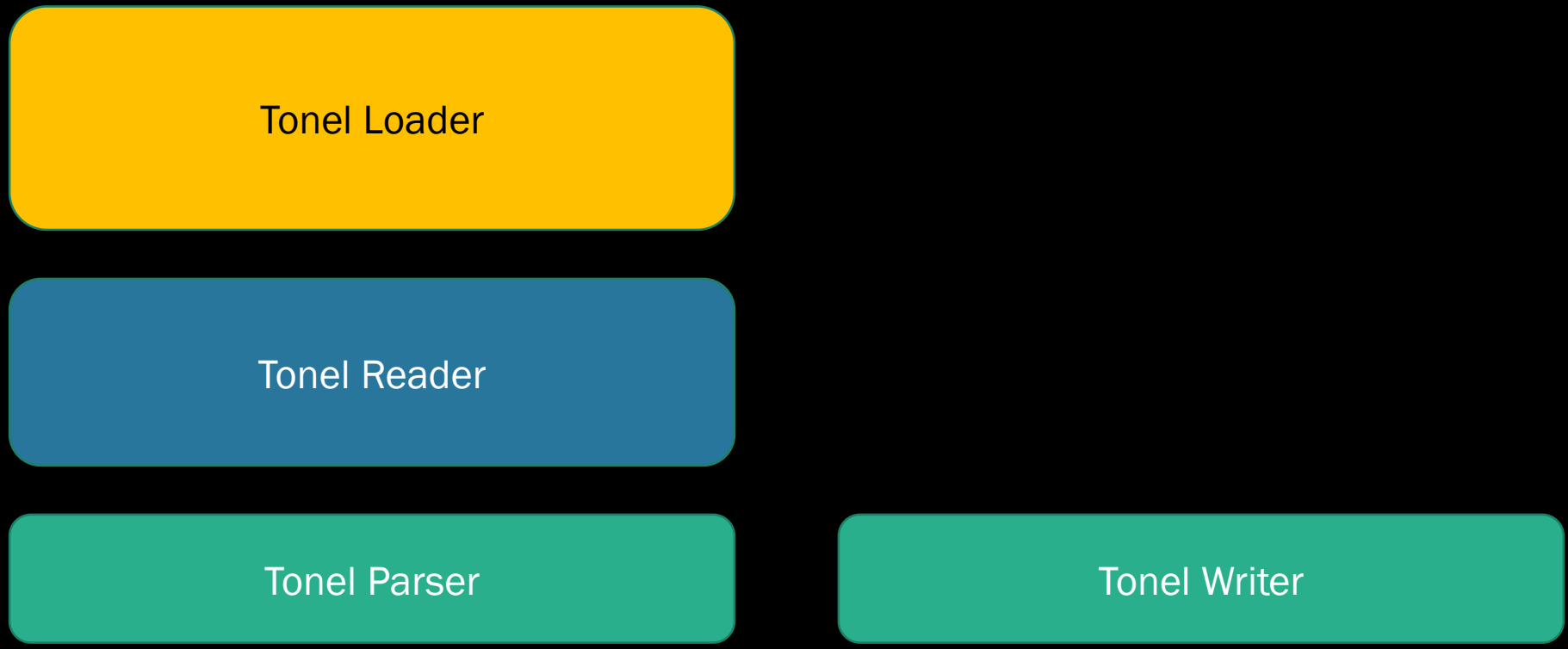
- Application
- Subapplication (hierarchy)
- Subapplication conditions
- Extension
- (none)
- SharedPools defined by pragmas

Pharo

- Package
- (none)
- (none)
- Extension
- Trait
- SharedPools as classes

Tonel Tools architecture

(Not really)



How it works

- The reader parses code and creates the object model
- The loader works on a per Application basis
- If there is an App/SubApp hierarchy the loader will build the hierarchy
  - (based on the metadata)
- Source gets through VAST compiler before going into the EM Library
- Writer reads from image, not EM Library

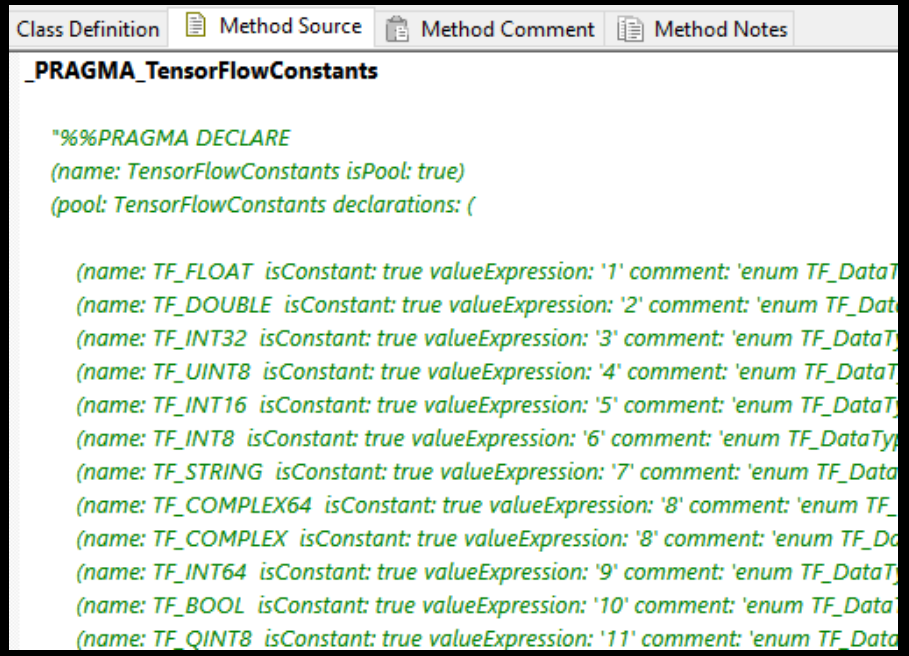
Bridging the gap

- Extra metadata written in type info
  - Parent application (`#vaParent:`)
  - Subapplication conditions (`#vaSubApplications: []`)
  - Prerequisite names (`#vaPrerequisites: []`)
- Extra metadata written in methods
  - Multiple categories (`#vaCategories: []`)
  - Method visibility (`#vaVisibility:`)


MIND THE GAP
MIND THE GAP
MIND THE GAP

Pools, the shared ones

- In Pharo they're classes with class variables
  - Initialized by message sends (usually class side #initialize)
- In VAST they're instances of SharedPool
  - Defined by a "declaration pragma"
  - Values can be initialized in the definition
  - Can have constant values
  - Are localizable by NLS subsystem
- "Compromise"



Caveats

- VAST will write #va* metadata and honor it back when reading
  - VAST-to-VAST, VAST-to-Pharo will work fine
- Pharo will ignore that metadata when reading
  - So it won’t have it when writing it back
- Limit to Lowest Common Denominator
  - Applications without SubApplications (no shadows!)
  - All methods public
- SharedPool conversion
  - From method to class when writing
  - From class to methods when loading
    - One method to define the pool
    - One to initialize its values
- Work in progress

Next steps

- Map Configuration Maps into something (BaselineOf...?)
- Strategy based behavior
  - Versioning
  - Prerequisite choosing
  - Interactive (GUI), Headless
- ENVY Library version migration to git commits
- Heavy use testing
- https://github.com/vasmalltalk/tonel-vast/issues

Questions?

Thanks.

https://github.com/vasmalltalk/tonel-vast/