Live Typing
Automatic Type Annotation that improves the Programming eXperience

Hernán A. Wilkinson - @hernanwilkinson

10Pines founder – Professor at UBA

agile software development & services

Phar

Nautilus - System Browser

Stop Dr.TDD

DrTDD
All tests are passing

Recommendation
Write a failing test

Apply recommendation | More information

Scoped | Variables

History Navigator

Type: Pkg1|^Pkg2|Pk.*Core$

- DrTDD
  - DrTDD
  - Model
  - Pharo
  - Pharo-Tests
  - Pharo-UI
  - Tests
  - Tests-Sandbox

- Developer
- DrTDD
- DrTDDMethodBrowser
- DrTDDNautilusPlugin
- DrTDDNautilusPluginIcons
- DrTDDNautilusPluginMorph
- *DrTDDPharoCommand*
- CreateNewTestCaseCommand

Hier. | Class | Com.

```
Object subclass: #NameOfSubclass
   instanceVariableNames: ''
   classVariableNames: ''
   package: 'DrTDD'
```

3/4 [24]

Format as you read W +L

“Wilklippy”

Nautilus - System Browser

Live Typing
Automatic Type Annotation that improves the Programming eXperience

Hernán A. Wilkinson - @hernanwilkinson

10Pines founder – Professor at UBA

agile software development & services

Is **Smalltalk** Cool??

YEAH!

Of Course!

Why?

Because it is **Dynamically Typed**

Because it is a **Live Environment**

But …

Looking for **senders** is done **statically**!!

we get ***more senders*** than the real ones
we need ***to manually filter*** them

Looking for **implementors** is done **statically**!!

we get ***more implementors*** than the real ones
we need ***to manually filter*** them

Renaming a message is done **statically**!!

we get ***more senders and implementors*** than the real ones
we need ***to manually filter*** them

Autocomplete in the browser is done **statically**

we get ***don't get the real messages*** an object understands

and so on…

But **Smalltalk** is Cool!!

Because it is **Dynamically Typed**

Because it is a **Live Environment**

But …

How can we get rid of this “but”?
Aren’t you tired of these problems?

~~But~~ …

I'm tired of all those problems
It's **time to act!**

What if we combine

Dynamically Typed

+

Live Environment

to get ***Automatic Type Annotation*** and ***improve the tools*** with that info?

10 Pines

What is Live Typing?

Live Typing

Automatic type annotation (done by the VM)

+

Tools to improve the development experience

(Annotation happens while the system runs)

It is **not a new** idea…

Since 1986!

A Simple Technique for Handling Multiple Polymorphism

Daniel H. H. Ingalls
Mail Stop 22-Y
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014

Abstract

Certain situations arise in programming that lead to multiply polymorphic expressions, that is, expressions in which several terms may each be of variable type. In such situations, conventional object-oriented programming practice breaks down, leading to code which is not properly modular. This paper describes a simple approach to such problems which preserves all the benefits of good object-oriented programming style in the face of any degree of polymorphism. An example is given in Smalltalk-80 syntax, but the technique is relevant to all object-oriented languages.

particular type (class), are not polymorphic, and do not depend on other types in the system.

All current object-oriented languages thus support simple polymorphism. That is to say, a variable or expression representing the receiver of a message may, dynamically, vary in type. Different but appropriate results will be produced, depending on the type of each receiver. This capability leads to a great simplification in the description of behavior of different but similar objects. Moreover, most object-oriented implementations provide an efficient message construct, so that this support for polymorphic receivers costs little more than a conventional procedure call.

Published in ECOOP '91 proceedings, Springer Verlag Lecture Notes in Computer Science 512, July, 1991.

Optimizing Dynamically-Typed Object-Oriented Languages With Polymorphic Inline Caches

Urs Hölzle
Craig Chambers
David Ungar[†]

Computer Systems Laboratory, Stanford University, Stanford, CA 94305
{urs,craig,ungar}@self.stanford.edu

Gradual Typing for Smalltalk

Esteban Allende[a,**], Oscar Callaú[a], Johan Fabry[a], Éric Tanter[a], Marcus Denker[b]

[a] *PLEIAD Laboratory*
Computer Science Department (DCC) — University of Chile
[b] *INRIA Lille Nord Europe*
CNRS UMR 8022 — University of Lille

Abstract

Being able to combine static and dynamic typing within the same language has clear benefits in order to support the evolution of prototypes or scripts into mature robust programs. While being an emblematic dynamic object-oriented language, Smalltalk is lagging behind in this regard. We report on the design, implementation and application of Gradualtalk, a gradually-typed Smalltalk meant to enable incremental typing of existing programs. The main design goal of the type system is to support the features of the Smalltalk language, like metaclasses and blocks, live programming, and to accomodate the programming idioms used in practice. We studied a number of existing projects in order to determine the features to include in the type system. As a result, Gradualtalk is a practical approach to gradual types in Smalltalk, with a novel blend of type system features that accomodate most programming idioms.

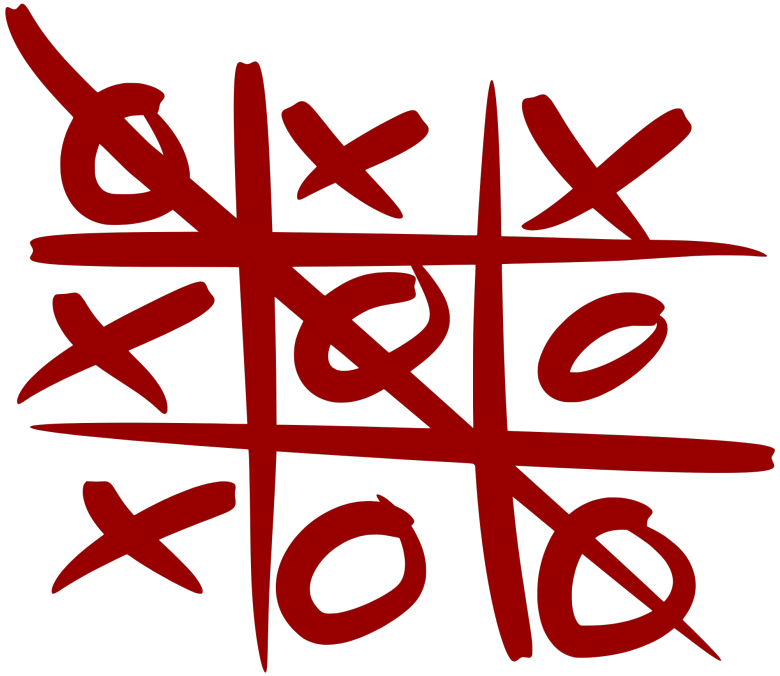
Keywords: Type systems, gradual typing, Smalltalk

It is not a new idea but…
It is a **particular** one with a **working implementation**

Live Typing is not about
type checking
in the “classic way”

10 Pines

Tools Examples



TicTacToe

10 Pines

Showing/Managing Types

Autocompletion

DynamicType
(SelfType, ClassType, InstanceType)

Thank you Jan Vrany & Marcus Denker

Actual Implementors

("a much better use of the human brain"-Tudor et al ☺)

Actual Senders
Sure and Possible
(Per Message Send analysis)

Refactorings with **Actual Scope**

Type Checker

(we know for sure when ***nil*** is assigned!)

Conclusion

With **no extra effort**,
we were able to **get rid off** of
most of the **disadvantages**

(you don’t have to maintain the types, it does not interfere when reading code, etc.)

Bret Victor:
‘we are used to play computers’

Unknow Smalltalker:
‘we are used to play memory type games’

Live Typing makes types explicit to you, you do not have to remember or infer them

It is a very **simple** technique that **heavily improves** the **programming experience**

feenk: When you start using it, you don’t want to loose it

It does not change the **syntax**
It does not stop you from **compiling**
It does not force you to **use it**
Types are not in the **source code**

I humbly believe it
respects and **honors**
the Smalltalk **spirit**

My Goal?

To say that **Smalltalk** is not **Dynamically Typed** anymore, but **Lively Typed**

(you func…. guys want types? You have types for free!)

The implementation has **bigger challenges** than in a **Statically Typed Language**

Statistics

Performance

	Typed VM	Stack VM	Difference
Aconcagua Tests	37 ms	22 ms	1.6 x
Chalten Tests	2400 ms	2204 ms	1.08 x
Refactoring Tests	56382 ms	39650 ms	1.42 x
TicTacToe Tests	3 ms	2 ms	1.5 x
Some Kernel Tests	220 ms	151 ms	1.45 x
Average			**1.41 x**

The important thing is that you do not notice it when you are programming

10 Pines

Memory

Live Typing Image*	Common Image	Difference
22 MB	10 MB	2.2 x

(*) Default configuration

- All type arrays size equals 10
- Instance Variables Usage: 8.8%
- Method Variables Usage: 8.4%
- Method Return Usage: 8.02%

10 Pines

Future Work

In the VM side

- Annotate types in closure parameters and variables (under development)
- Support for Parameterized types (Generics) is needed for collections, association, etc. (Collection<T>, Association<K,V>, etc.) (under development)
- Implement it on the JIT VM (Currently it is implemented on the Stack VM)
  (under development – need a lot of help!!)
- To think about:
  - Change the COMPILER (not the VM) to generate and initialize the PIC at compile time!!

10 Pines

In the Image side

- 🅰 Support for Parameterized types (on development)
- 🅰 Add more type cast cases in the Type Checker
- 🅰 Check for parameter types (Freeze annotated types before)
- 🅰 Use Type Checker infrastructure to improve even more the autocomplete
- 🅰 Import type info from production images to development images
- ❓ Things to try/think about:
  - ❓ Improve Type Checker to warn about dead code
  - ❓ Check for soundness in parameters and method returns
  - ❓ Delete method with transitive closure of actual sends in that method

10 Pines

Thanks!

@HernanWilkinson – hernan.wilkinson@10pines.com – www.10pines.com