Static vs Dynamic Types in Software Development

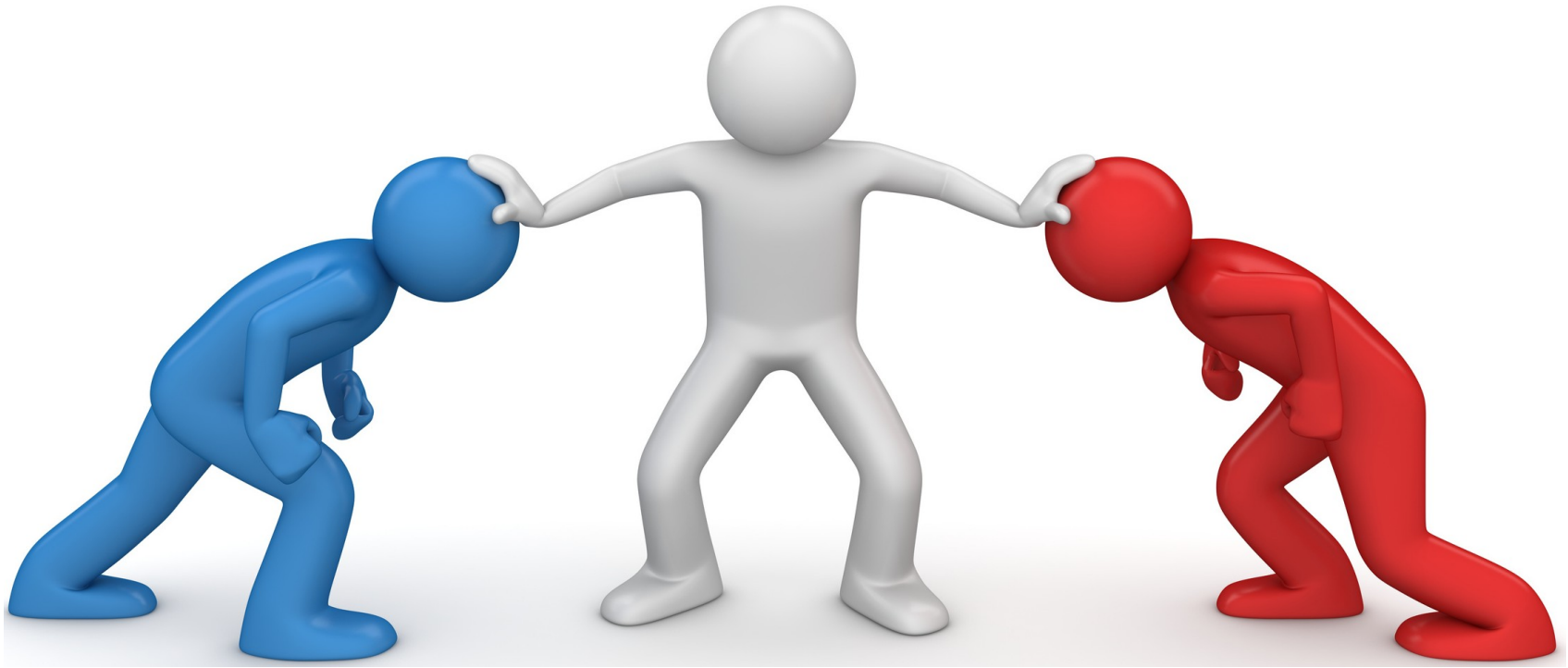
Emma Callery

University of Minnesota, Morris

UMN Morris Senior Seminar Conference Fall 2014

Why is this important?

- Static Types or Dynamic Types? The BIG debate



Outline

Static Types

Static types: types known to the compiler at compile-time; Programmers required to include necessary information. Most common languages are Java and C++.
An example in Java.

```
1 public class Language{
2  public String setLanguage(String lang){
3  return "Primary language is: " + lang;
4  }
5 }
```

Dynamic Types

Dynamic types are checked by the interpreter at run-time.
Dynamically typed languages include JavaScript, Ruby, and Groovy.
An example in JavaScript.

```
1 var Language{
2 getLanguage: function(lang){
3 console.log(‘Primary language is: ’ + lang);
4 }
5 };
```

Groovy

An example in Java (Static)

```
1 public class HelloWorld{
2  private String name;
3  public void greet(){
4  return "Hello " + name;;
5  }};
```

An example in Groovy (Dynamic)

```
1 class HelloWorld{
2  def name;
3  def greet() { "Hello ${name}" }
4 };
```

Study of Programmer Preferences

In 2014, *How do Programmers Use Optional Typing? An Empirical Study*,

- Two Brazilian researchers, Carlos Souza and Eduardo Figueiredo's study.
- In which contexts Groovy programmers type or do not type their declarations.
- 6638 Open Source Groovy projects.
- Five questions.

Study Questions

- Q 1 Do programmers use types more often in the interface of their modules?
- Q 2 Do programmers use types more often in test classes and scripts?
- Q 3 Does the experience of programmers with other languages influence their choice for typing their code?
- Q 4 Does the size, age or level of activity of a project have any influence on the usage of types?
- Q 5 In frequently changed code, do programmers prefer statically typed over untyped dynamic declarations?

Analysis

The researchers gathered different kinds of meta-data, mainly

- The kind and number of different type declarations.
- The visibility of type declarations.
- The frequency and total number of commits to the projects.
- The background experience of the programmers.

Results

Q 1 Do programmers use types more often in the interface of their modules?

Yes, interface variables and fields that are public, such as constructors and method parameters are frequently typed.

Q 2 Do programmers use types more often in test classes and scripts?

No, test classes, usually, have one sole purpose. Scripts are for the most part small, easy to understand, and can not be accessed by other modules.

Results cont.

Q 3 Does the experience of programmers with other languages influence their choice for typing their code?
Yes, how programmers uses types is greatly impacted by what the programmers are use to. This is believed to be the largest factor on how programmers use types.

Results cont.

Q 4 Does the size, age or level of activity of a project have any influence on the usage of types?
No, the usage of types in 'mature' projects was similar to that of other, 'non-mature', projects.

Q 5 In frequently changed code, do programmers prefer typed over untyped declarations?
No, programmers appear to prefer untyped declarations.

Types and Development Time Study

In 2011, *Static vs. Dynamic Type Systems: an Empirical Study About the Relationship between Type Casts and Development Time*

- Two researchers, Andreas Stuchlik and Stefan Hanenberg.
- The Argument: The existence of type declarations should lead to a reduction in the development time of simple programming tasks.
- 21 students.
- 2 sets of 5 tasks; one set in Java the other using a restricted form of Groovy.

The Experiment

All participants completed both sets of tasks.
Tasks differed in:

- number of declarations.
- Lines of Code.

Results

Lowest Time Results	Sum	Task 1	Task 2	Task 3	Task 4	Task 5
Group A	—	Groovy	—	—	—	—
Group B	Groovy	Groovy	—	Groovy	—	—

Dashes: No significant benefit

Influence of Static Types Study

In 2012, *An Empirical Study of the Influence of Static Type Systems on the Usability of Undocumented Software*

- By Clemens Mayer, Stefan Hanenberg, Romain Robbes, Eric Tanter, and Andreas Stefik.
- The Argument:Types act as a form of documentation.
- 27 students.
- 2 sets of 5 tasks; one set in Java the other using a restricted form of Groovy.

The Tasks

Task 1: (Easy, 1 class to identify) Return an instance of a Tree class.

Task 2: (Easy, 3 classes) Initialize a Tree object.

Task 3: (Medium, 3 classes) Transform the Tree.

Task 4: (Hard, 3 classes) Add to a node to the Tree.

Task 5: (Easy, 6 classes) Create a 'menu' for the Tree.

The Experiment

The researchers had two null hypotheses:

1. The development time for completing a programming task in an undocumented API is equivalent when using either a static type system of a dynamic type system.
2. There is no difference in respect to development time between static and dynamic type systems, despite the number and complexity of type declarations in an undocumented API.

Influence Study Results

Aspect	Task 1	Task 2	Task 3	Task 4	Task 5
Less Development Time	Java	Groovy	Groovy	Java	Java
Fewer Builds/Runs	——	Groovy	Java	Java	——
Fewer Files Looked At	Java	Groovy	Java	Java	Java
Fewer File Switches	Java	Groovy	Groovy	Java	Java

Conclusion

There are too many trade offs in both systems for there to be one conclusive answer.

- Study 1: largest effect comes from familiarity.
- Study 2: dynamic is better in some situations
- Study 3: static maybe better for larger projects.

Any Questions?