Final Details

Kathleen Dollard

@kathleendollard | www.CodeRapid.com

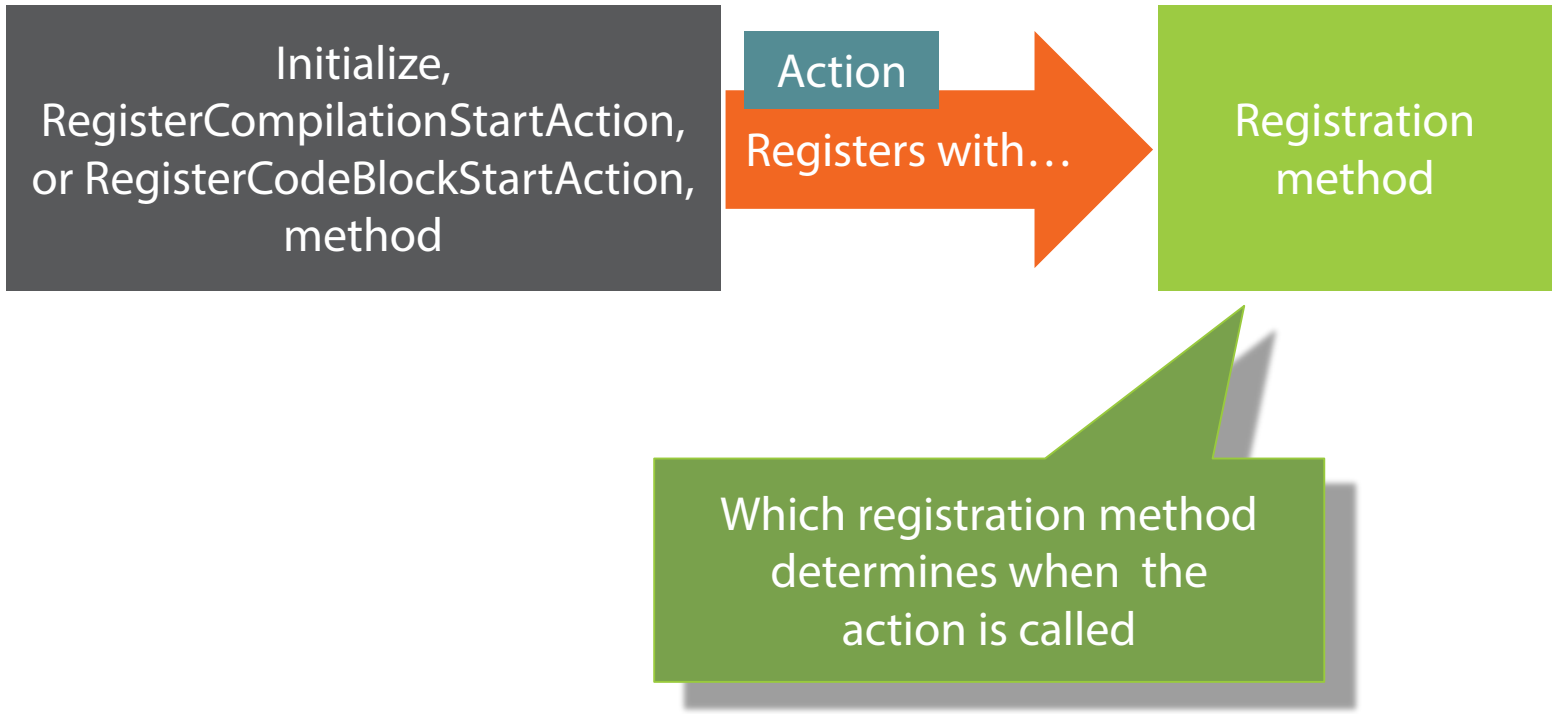
Execution Order

Localizing (or not)

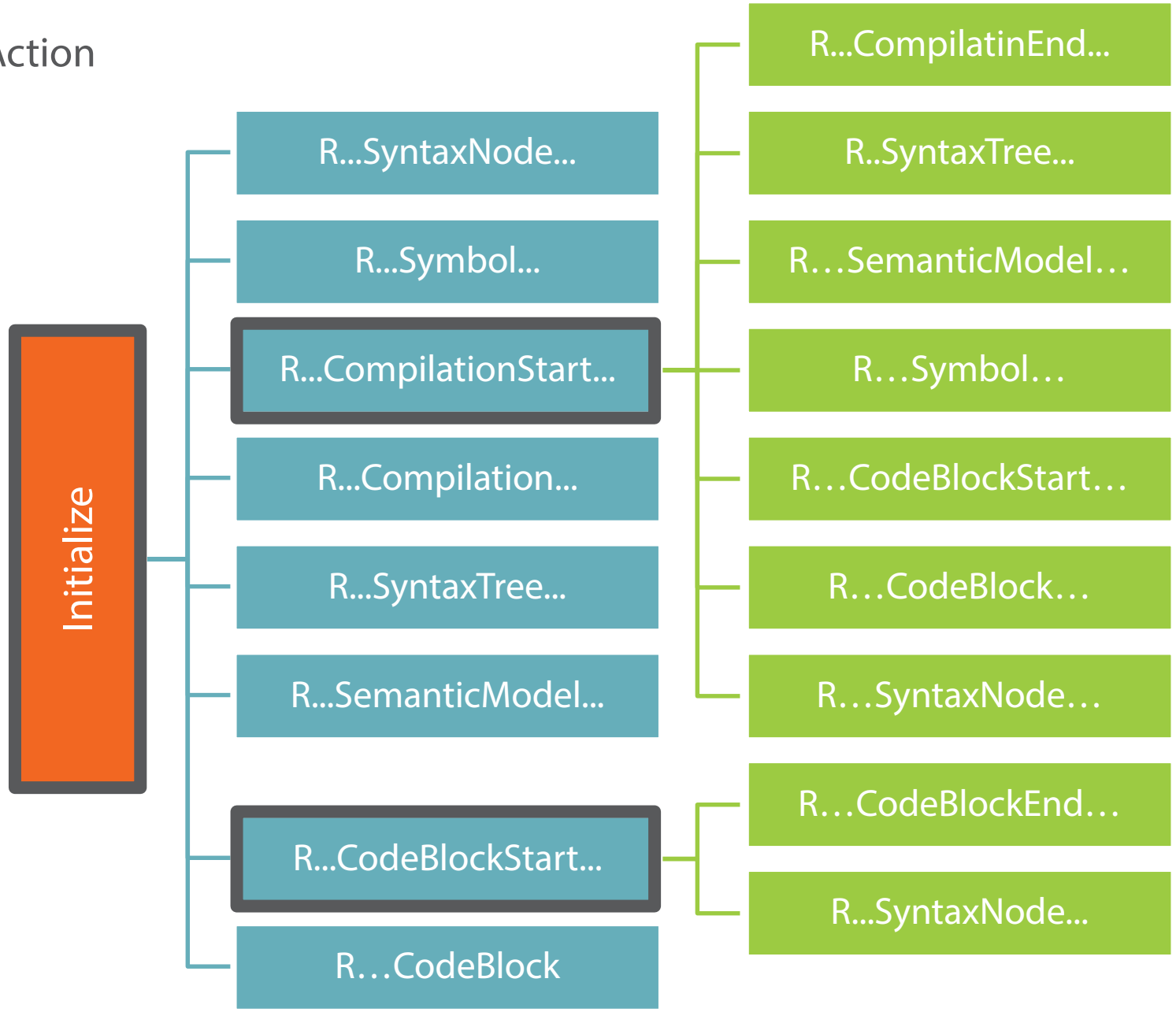
Deployment

Guidelines and Performance

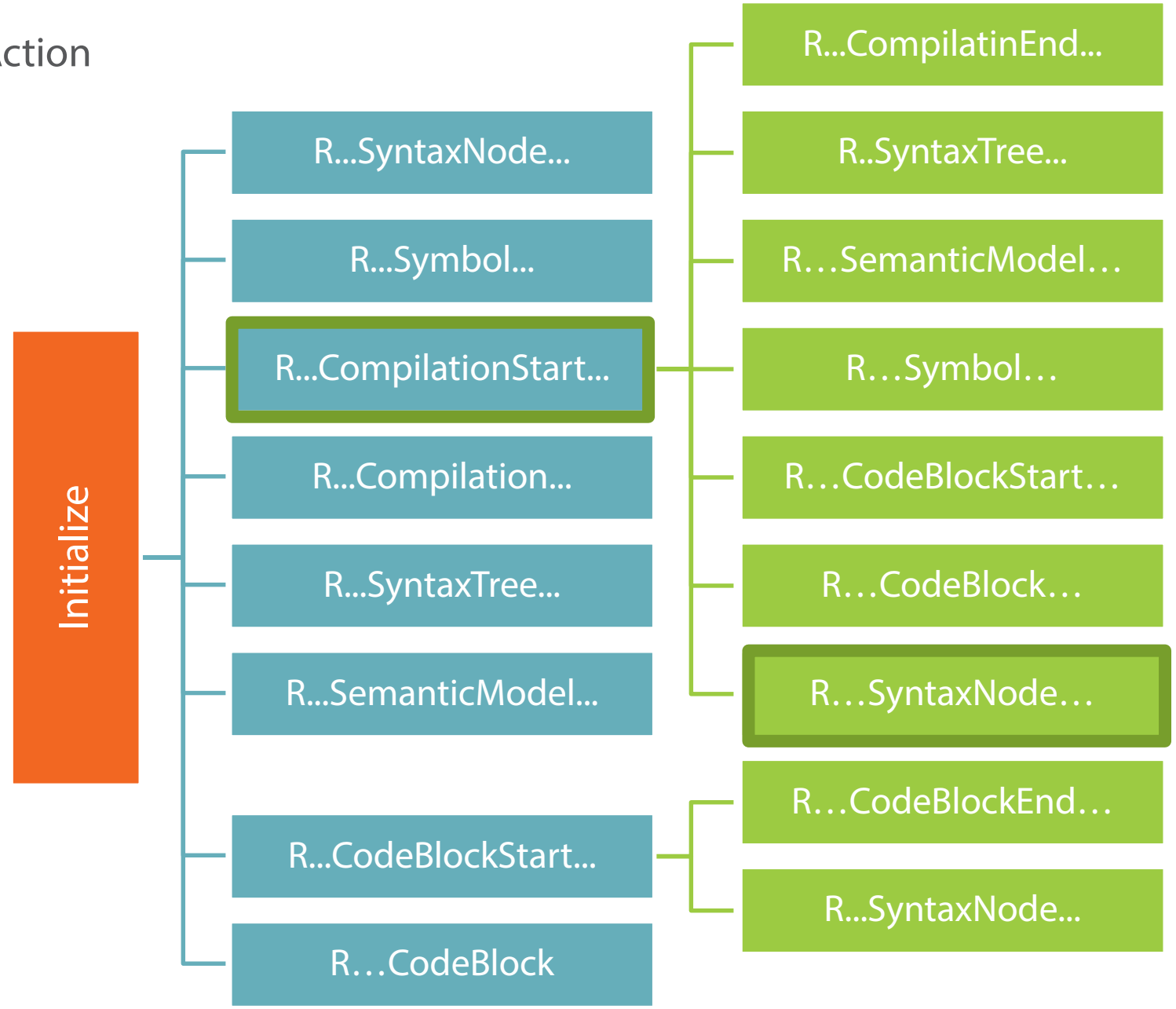
Execution Order Is Determined by Registration Method

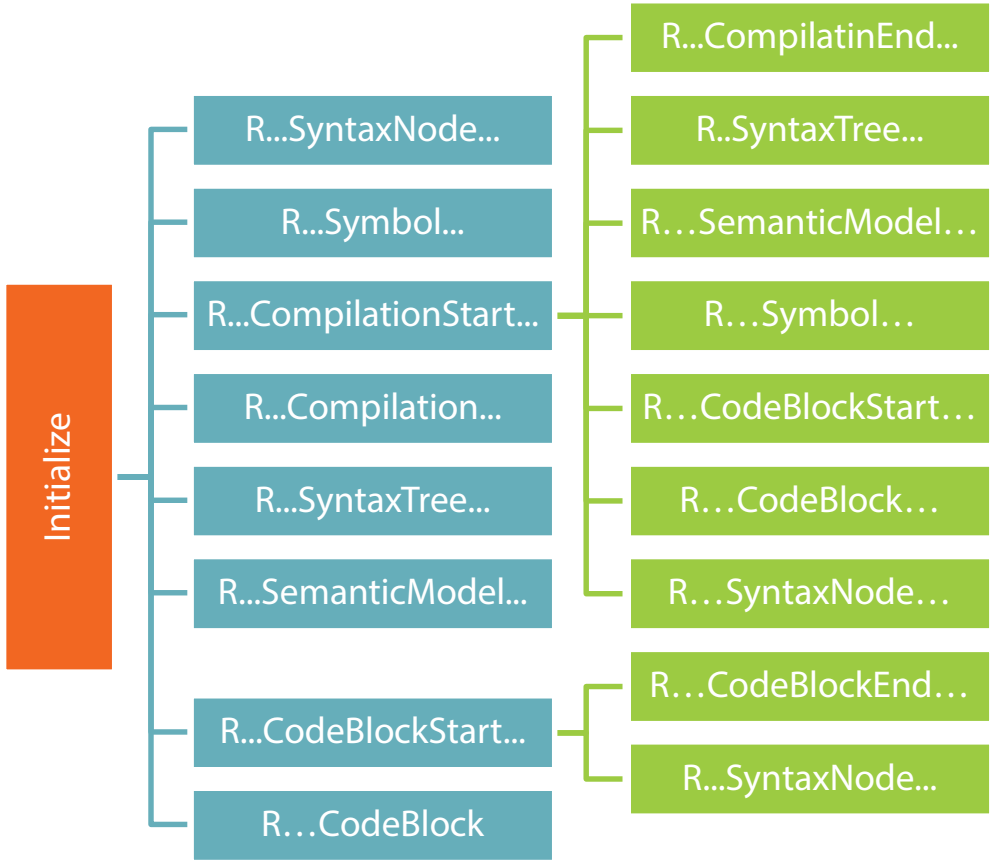


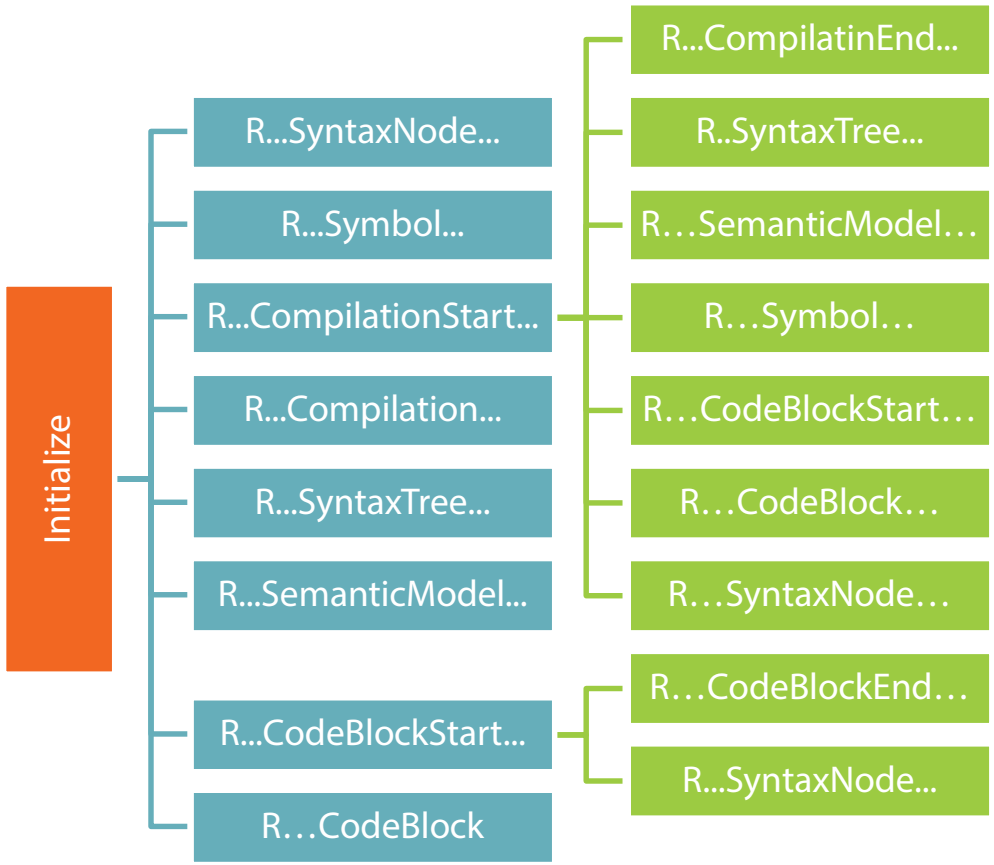
Register[ ]Action



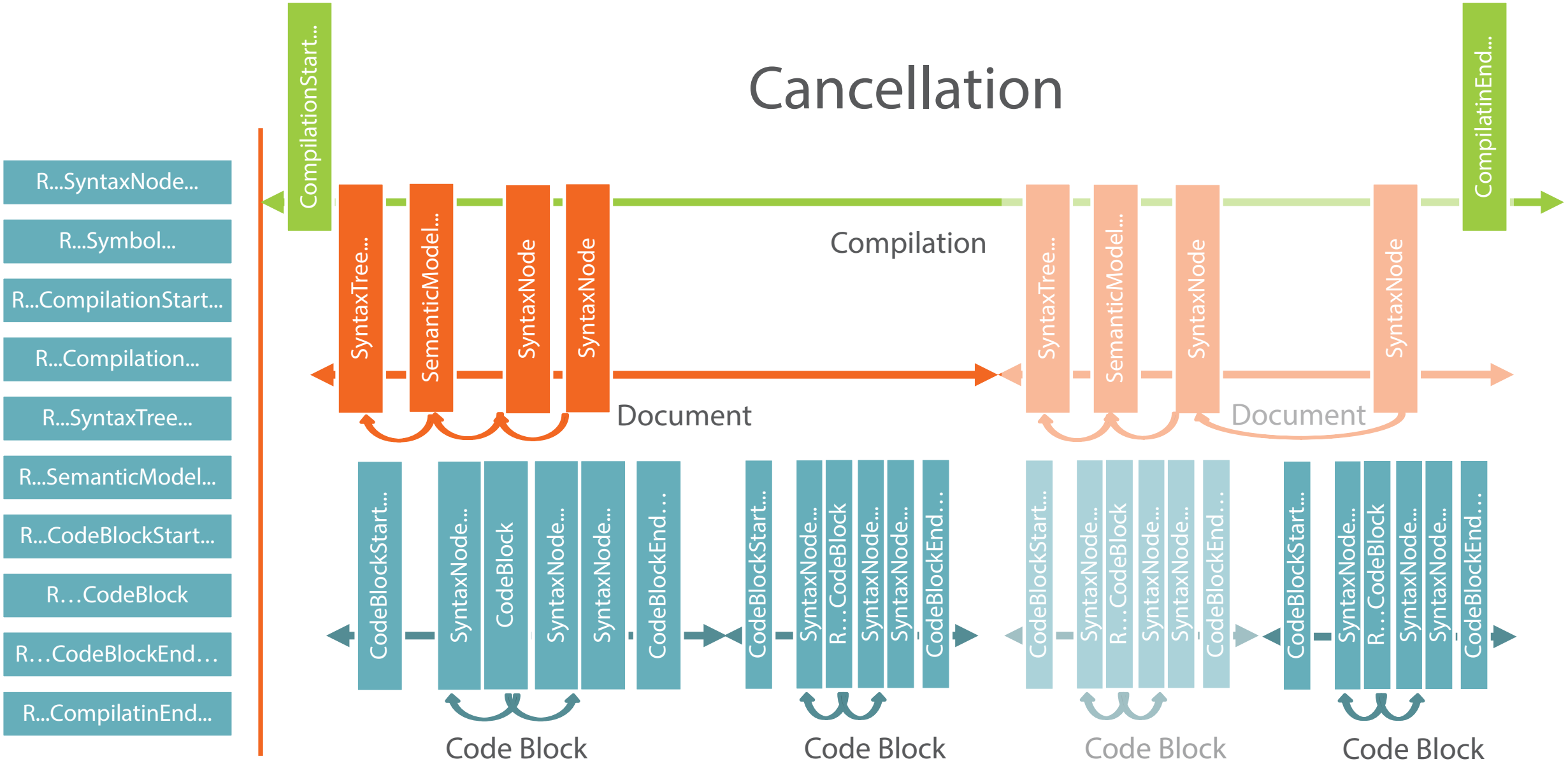
Register[ ]Action







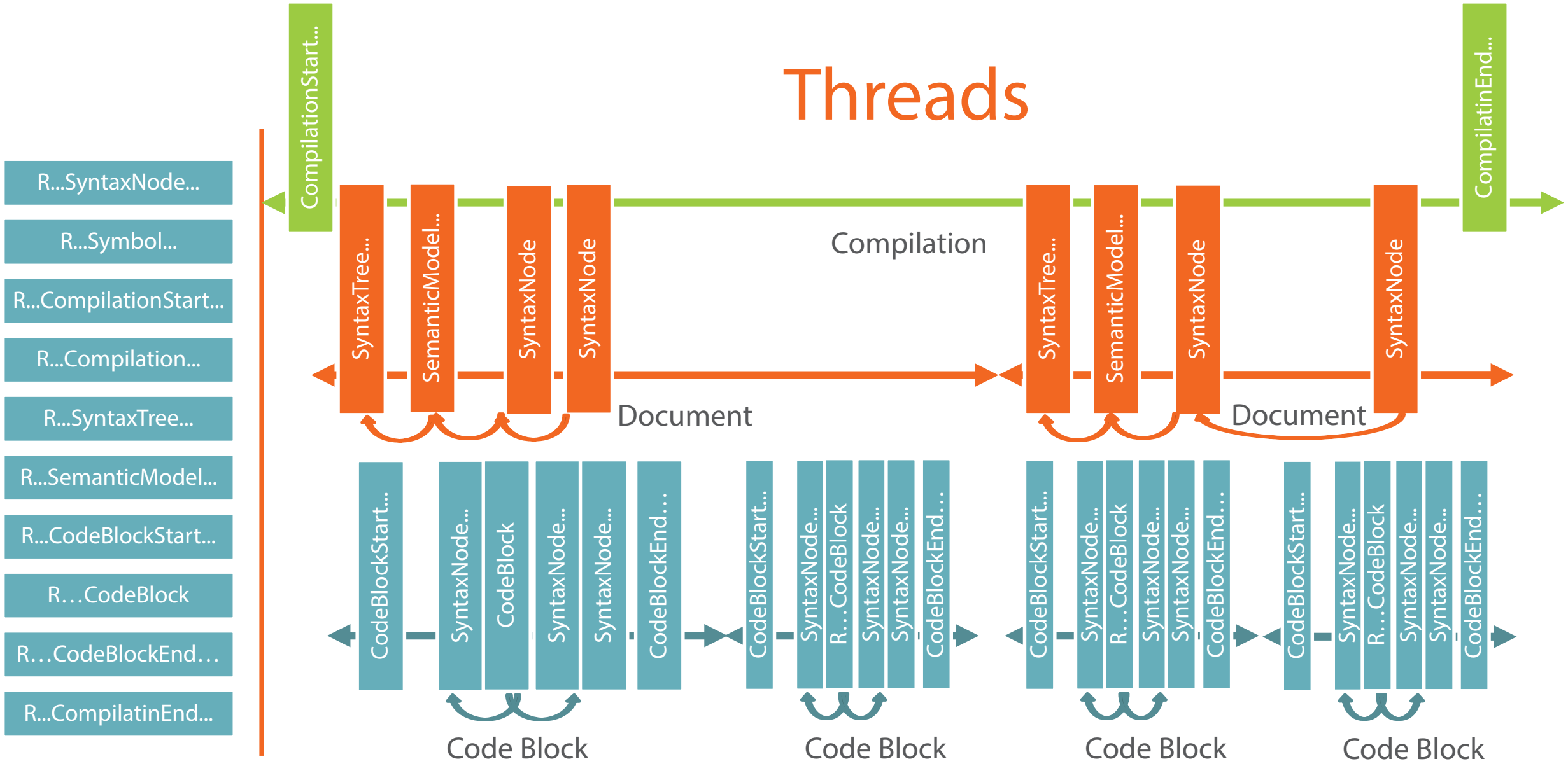
Cancellation

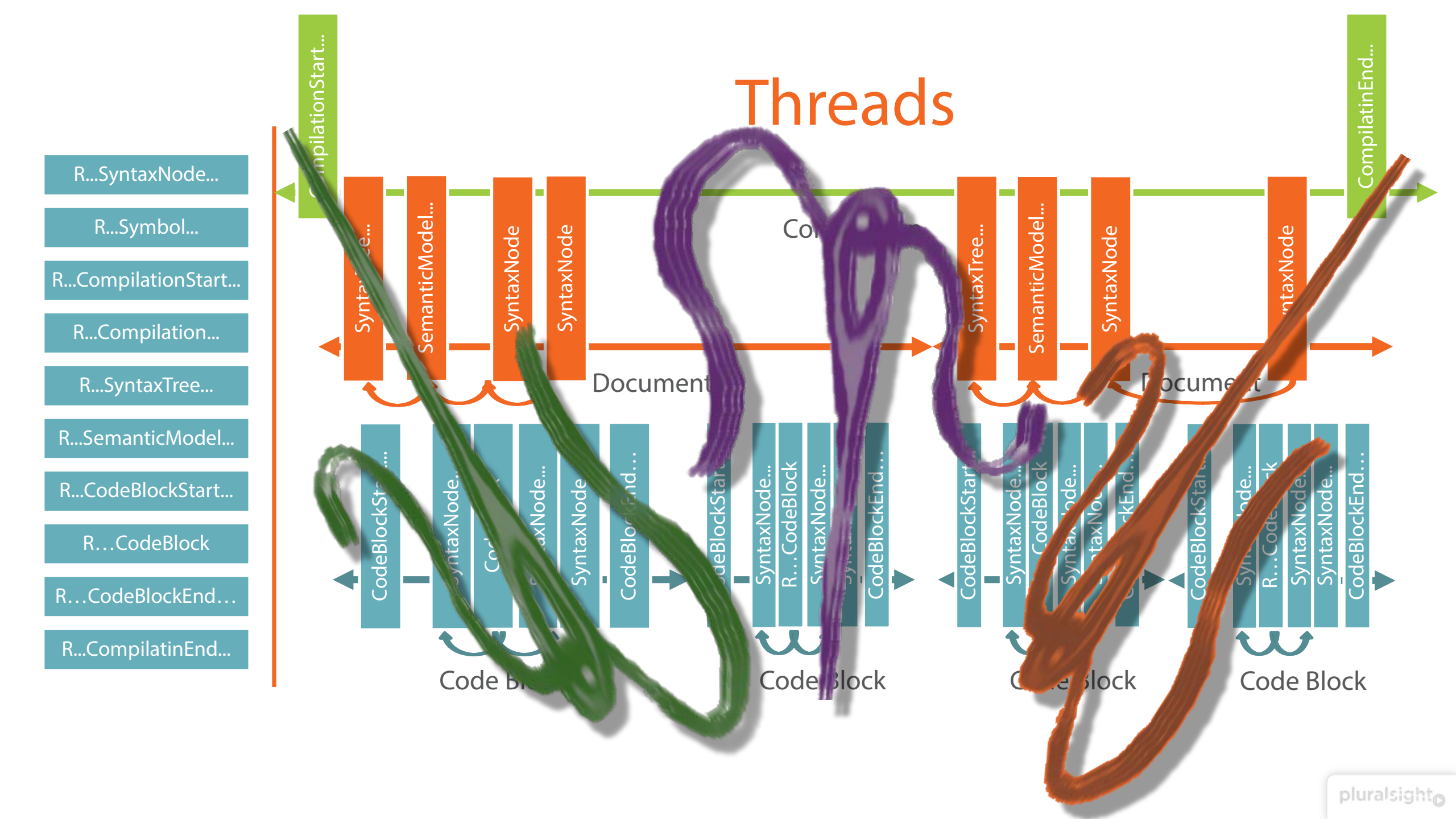


The Only Guarantees Are…

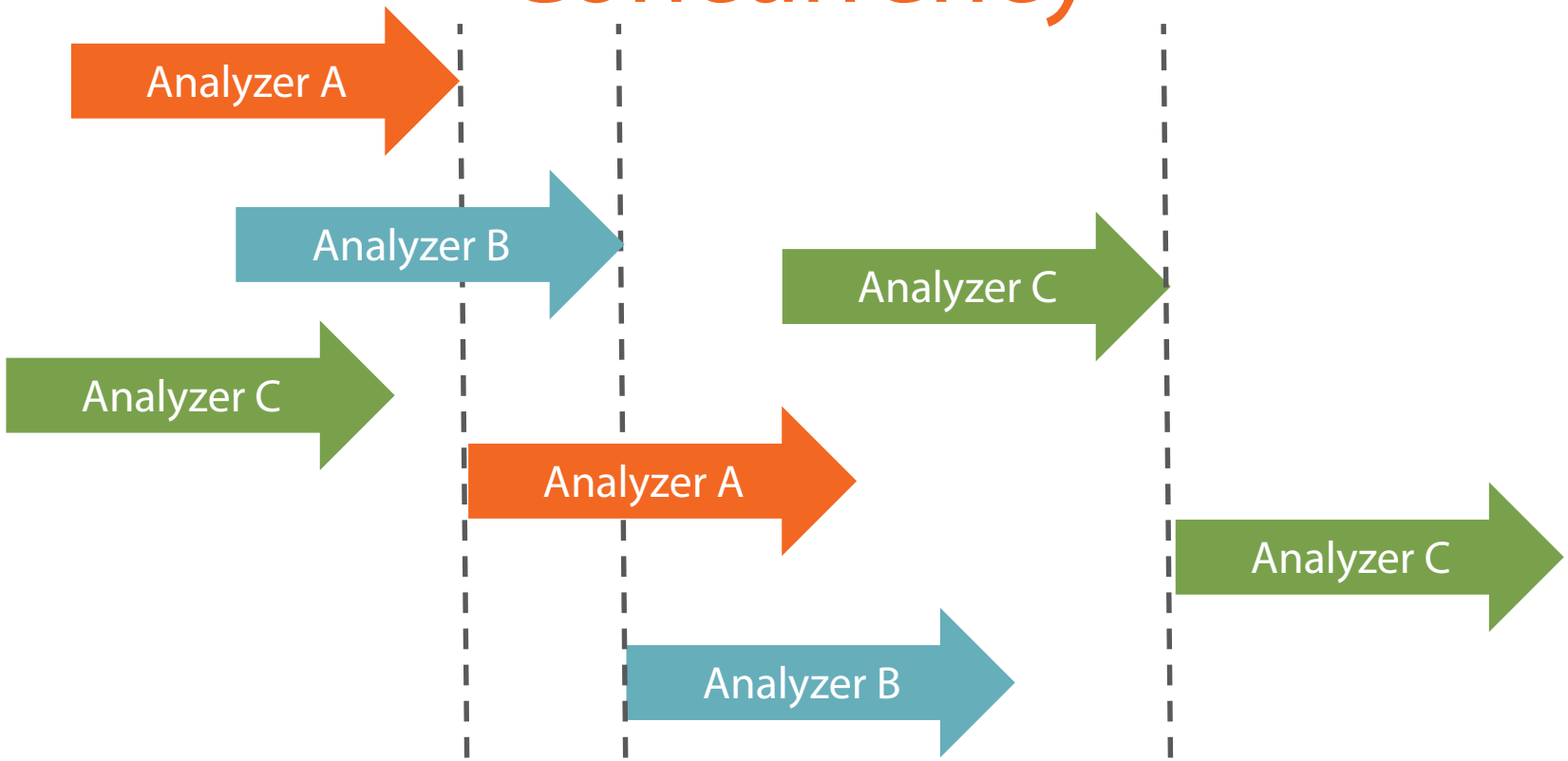
- Compilation start actions will run first
- Compilation end actions will run last
- Compilation actions will run once
- Code block start actions will run before anything in the code block
- Code block end actions will run after everything in the code block
- Code block actions run once per code block
- Within each compilation
  - Code blocks will each be processed once
  - Syntax tree and semantic model actions will run once per document
  - Syntax node actions will run once for each node of the specified type
  - Symbol actions will run once for each symbol of the specified type

Threads



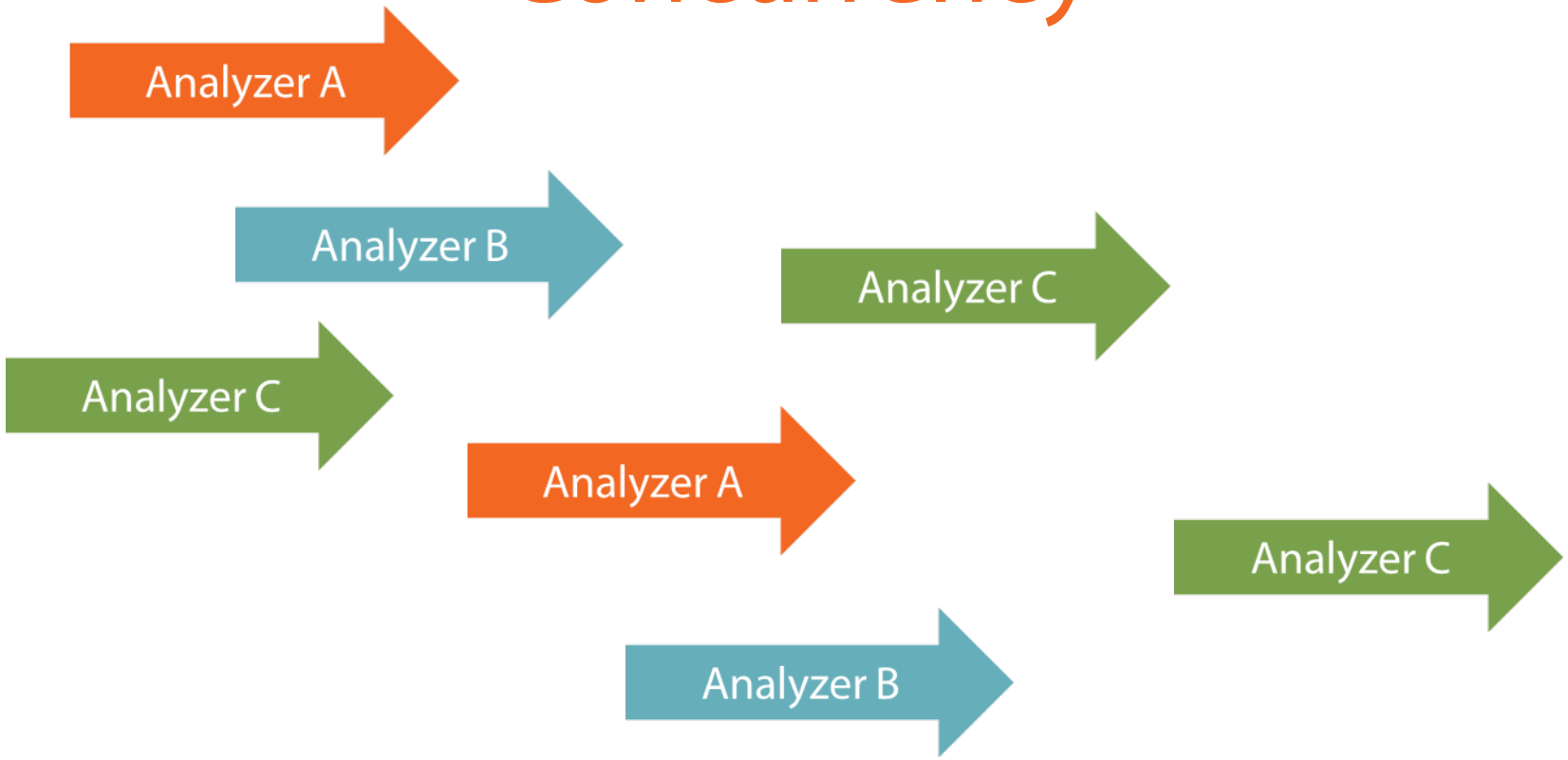

Threads
R...SyntaxNode...
R...Symbol...
R...CompilationStart...
R...Compilation...
R...SyntaxTree...
R...SemanticModel...
R...CodeBlockStart...
R…CodeBlock
R…CodeBlockEnd…
R...CompilatinEnd...
CompilationStart...
SemanticModel...
SyntaxNode
SyntaxNode
SyntaxTree...
SemanticModel...
SyntaxNode
CompilatinEnd...
Document
CodeBlockStart...
SyntaxNode...
SyntaxNode...
SyntaxNode
CodeBlockEnd...
CodeBlockStart...
SyntaxNode...
R...CodeBlock
SyntaxNode...
CodeBlockEnd...
CodeBlockStart...
SyntaxNode...
CodeBlock
CodeBlockStart...
SyntaxNode...
SyntaxNode...
CodeBlockEnd...
Code Block
Code Block
Code Block

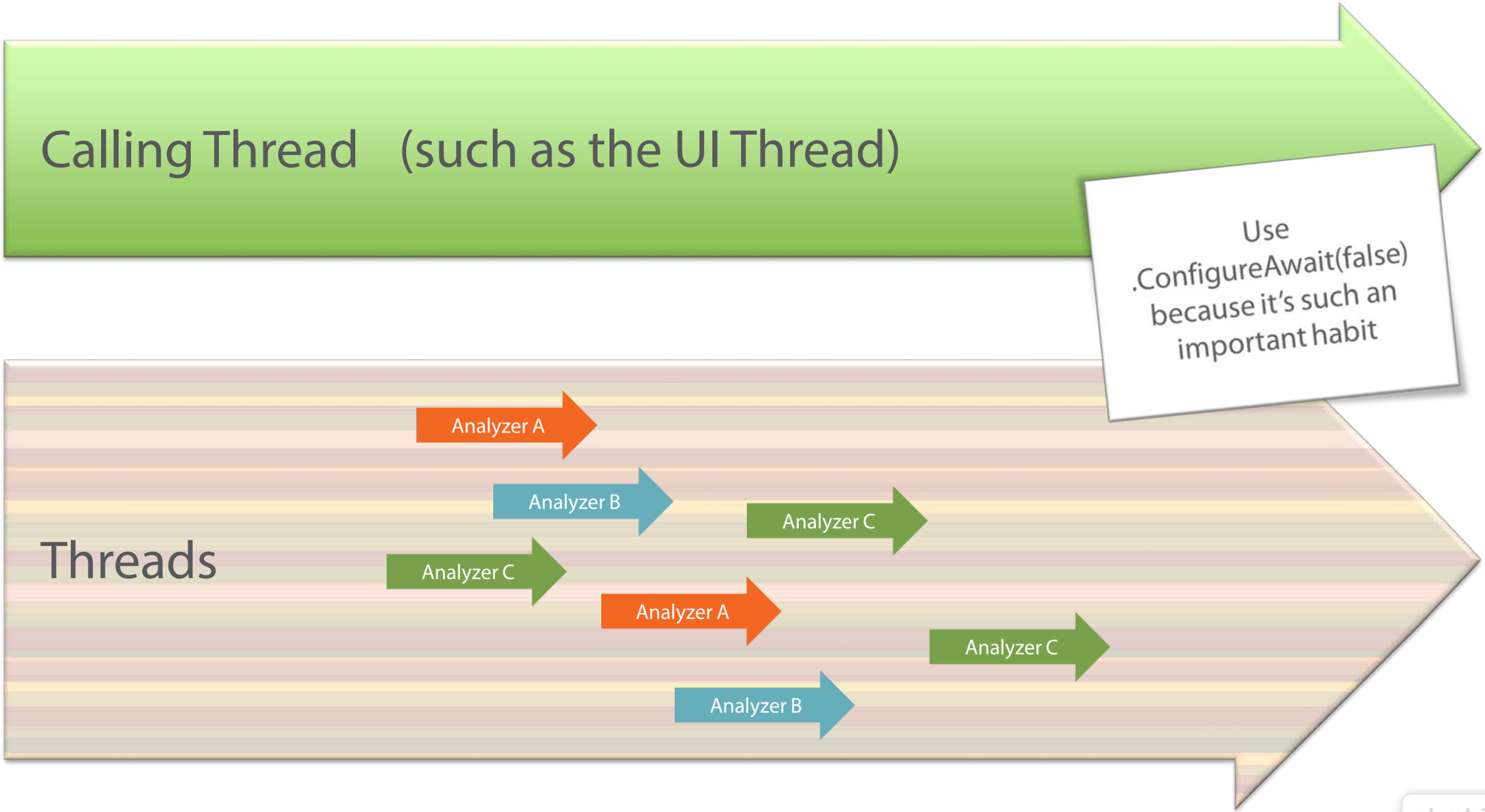
Concurrency



Actions of a single analyzer will not run concurrently
Separate analyzers should not share state
Locks are not generally required

Concurrency




Calling Thread (such as the UI Thread)
Use .ConfigureAwait(false) because it's such an important habit
Threads
Analyzer A
Analyzer B
Analyzer C
Analyzer C
Analyzer A
Analyzer C
Analyzer B

Localization

Don't actually localize

- English Google/Bing trumps comfort
- Most translations are bad

- Allow teams to localize
- Non-native OSS analyzers can get help translating into English

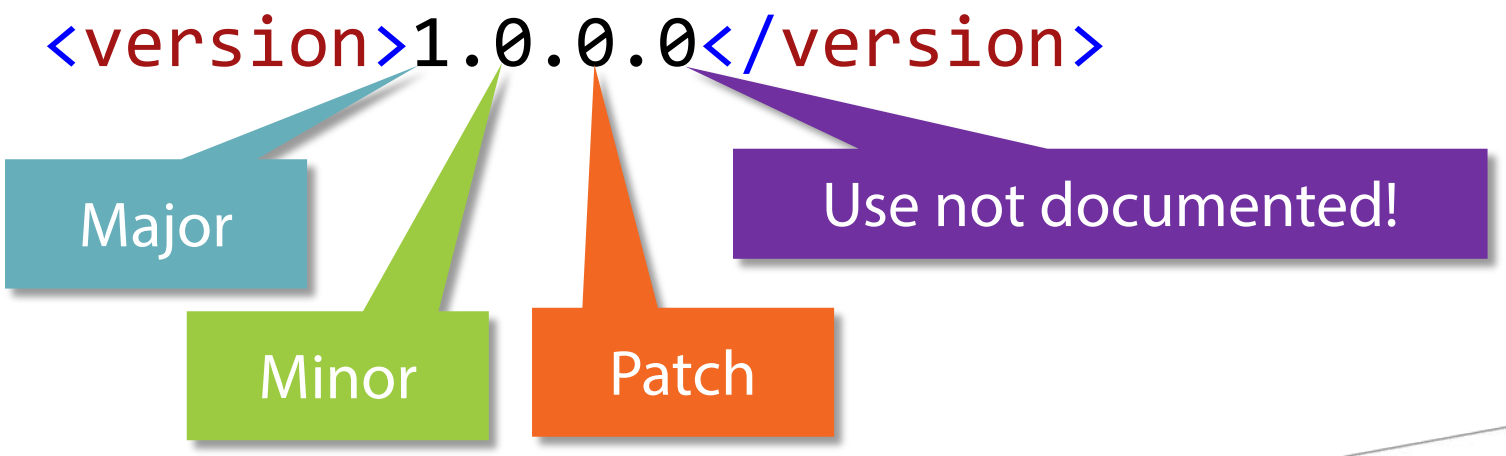
- English Google/Bing trumps comfort
- Most translations are bad

Prepare for Localization

Put Your Strings in Resource Files

```
internal static readonly LocalizableString Title =
        new LocalizableResourceString(
                    nameof(Resources.EmptyCatchClauseTitle),
                    Resources.ResourceManager,
                    typeof(Resources));
```

NuGet Semantic Versioning



- **Major**: Breaking changes.
- **Minor**: New features, but backwards compatible.
- **Patch**: Backwards compatible bug fixes only.

NuGet Pre-Release

-alpha

-beta

```
<version>1.0.0-alpha</version>
```

Absolutely anything!

-rc

-iamnotdoneyet

Visual Studio®

C#


```
foreach (var item in list)
{
    ThingYouShouldOnlyDoOnce();
```

```
var list = new List<string>();
// Where previous thing should be called


PreviousThingMustBeCalled();
// BTW, this is tricky, & I doubt guaranteed
```

```
var myDate = new DateTime();
```

Guidelines

Creating Great Analyzers

Guidelines

Flexibility

Performance

DiagnosticIDs

Managing Opinions

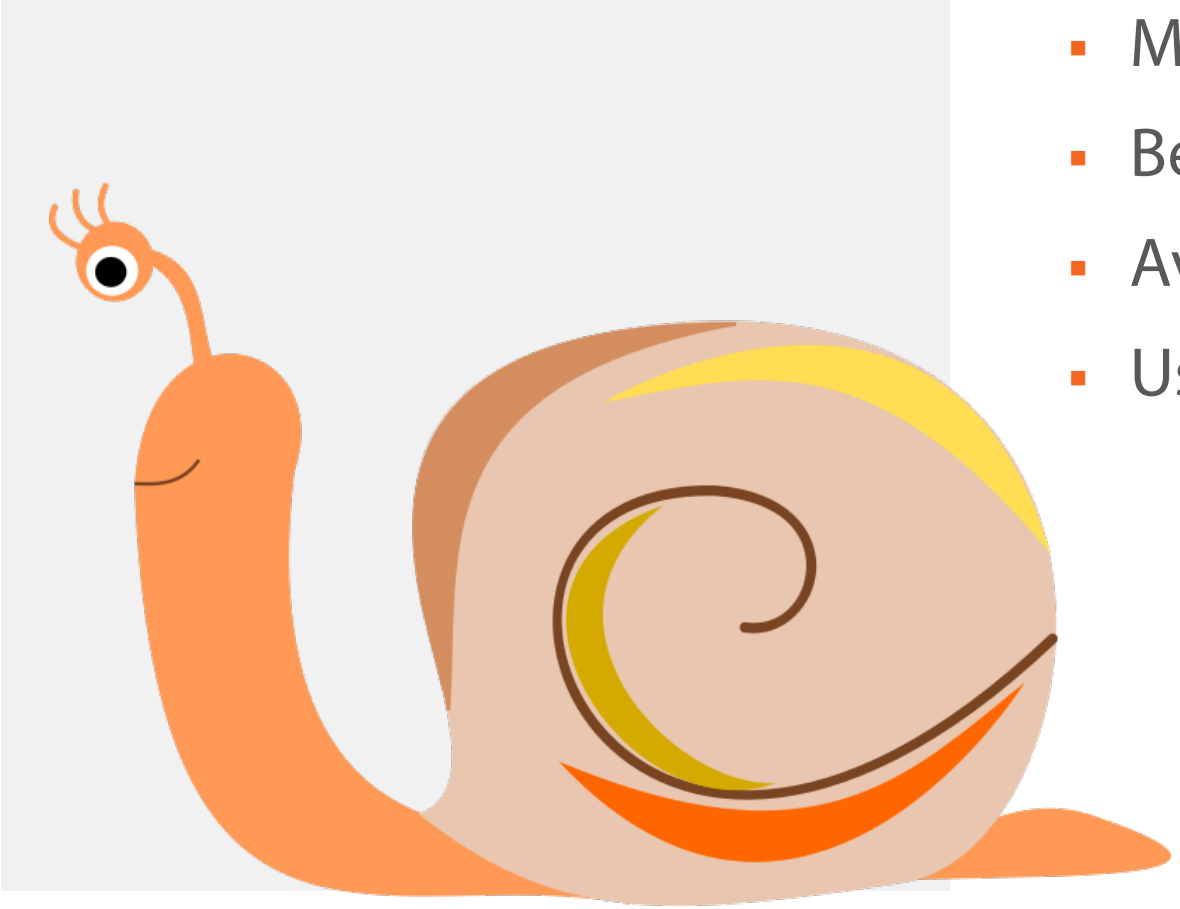
Severity

Organization and Categories

Use Portable Class Libraries for Analyzers

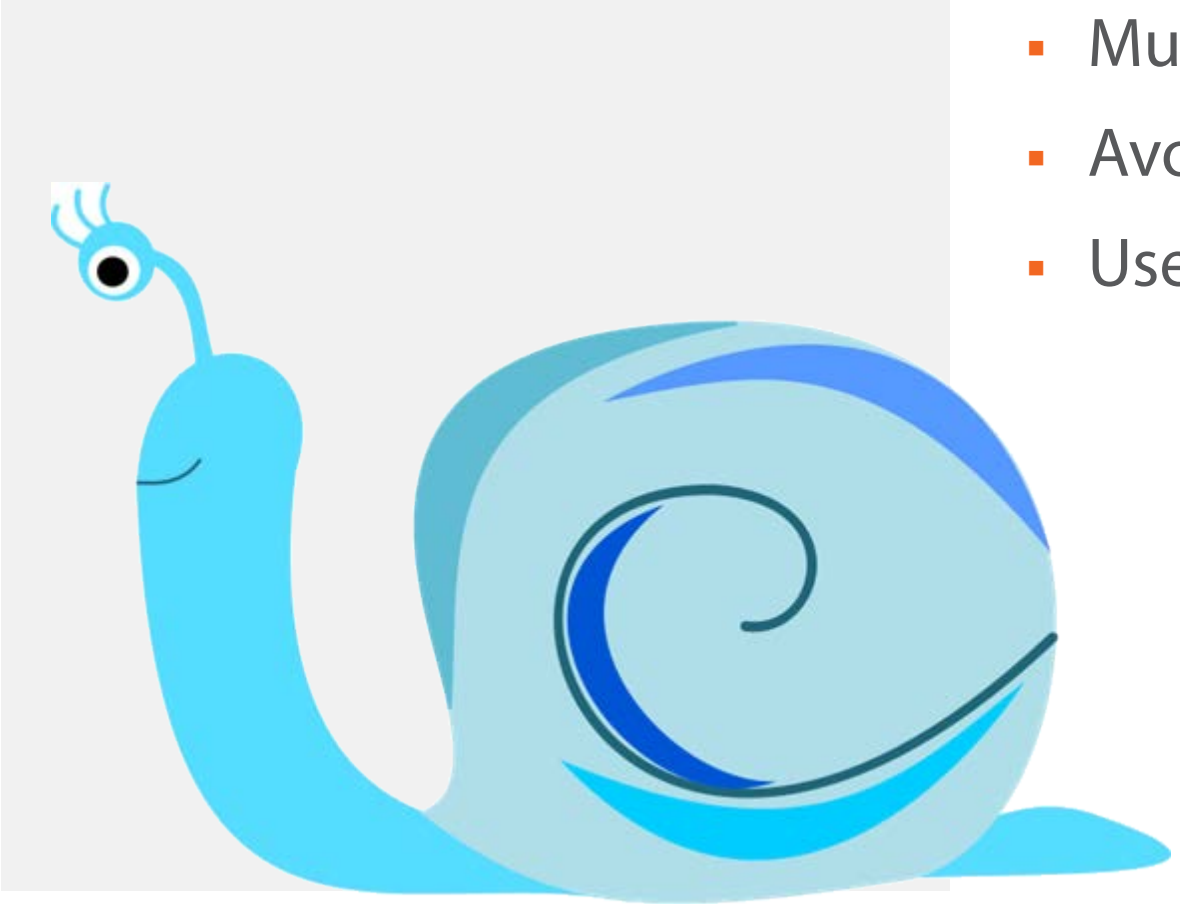
- Run during build process
- Run on any machine
- Forces isolation of dependencies
  - Example: WPF dialog for extra information

Avoid Slow Things in Analyzer Actions



- Most registered analyzer actions run often
- Be proactive avoiding exception
- Avoid using system resources!
- Use refactoring if it will be slow

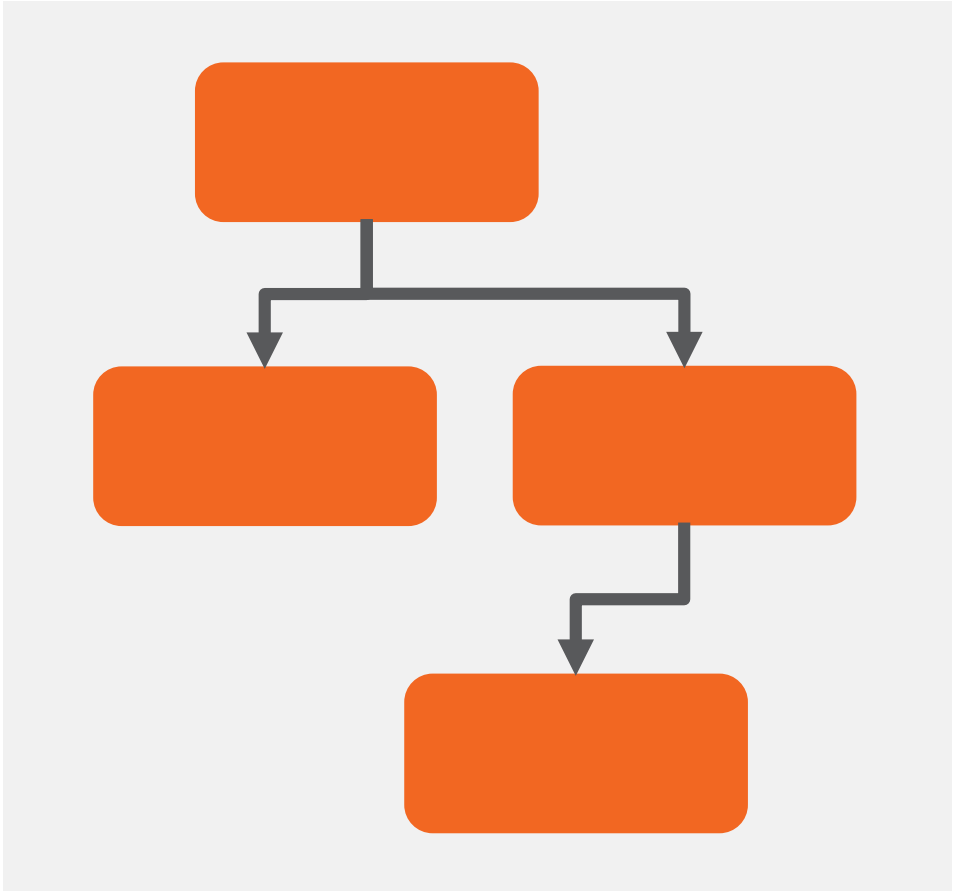
Also, Avoid Slow Things in RegisterCodeFixesAsync



- Must run before displaying code-fix options
- Avoid using system resources
- Use refactoring if it will be slow

Retrieve the Semantic Model

- If Semantic is offered without async
  - It's fine to use it
- In some cases, async retrieval may have a slight performance impact

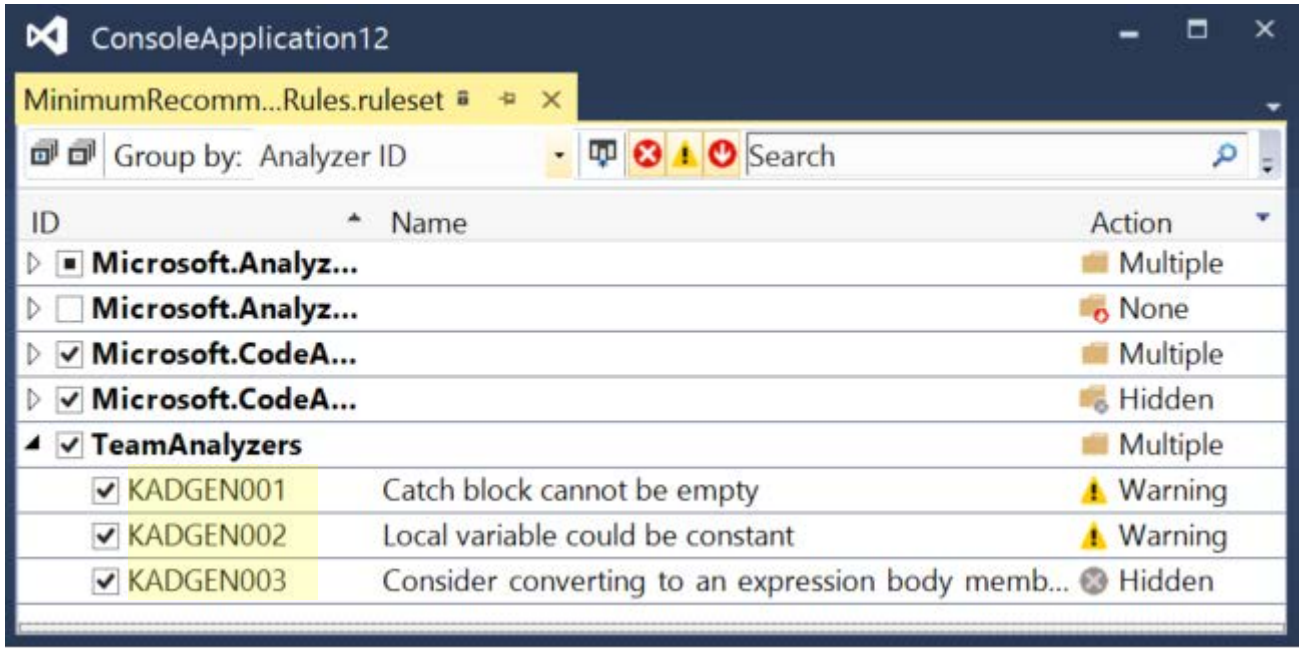


Diagnostic IDs Must Be Unique in All the Universe

KADGEN001

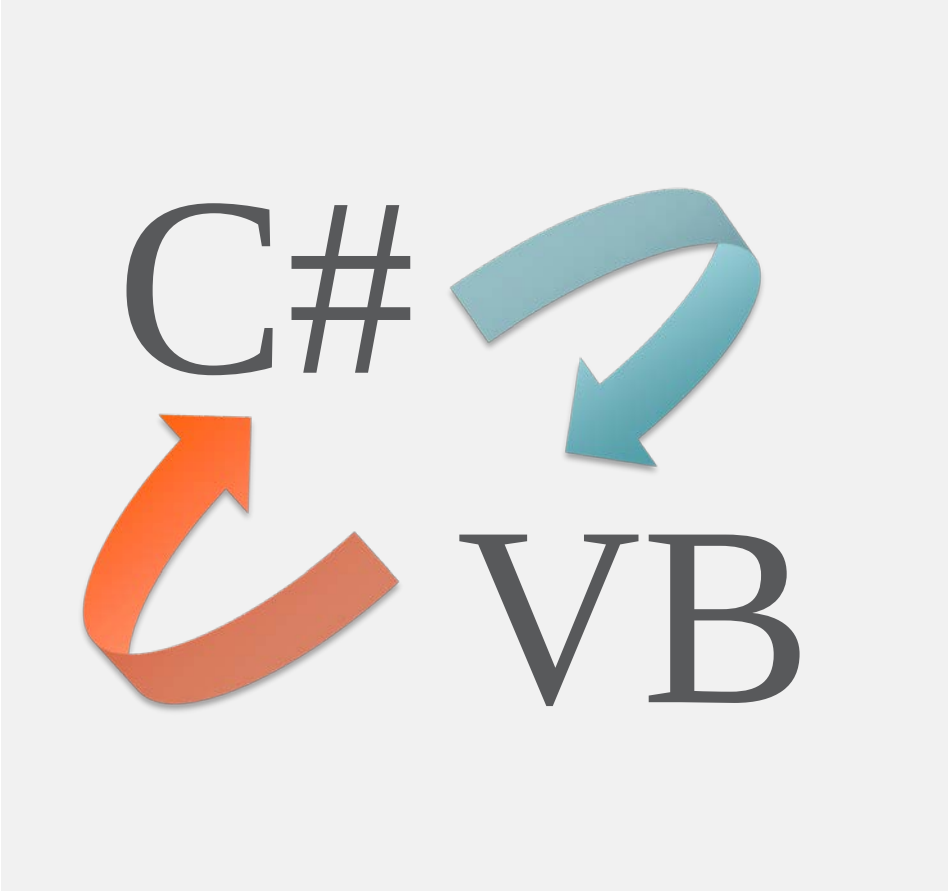
- Plan a prefix you can rely on being unique
  - More than two characters
- Ensure unique numeric suffixes
- If you copy and tweak someone's analyzer, ensure you also change the ID
- Keep them as short as possible
  - And no shorter
  - Display in suppression dialog

Do Not Change Your DiagnosticId's



- Your DiagnosticId's will be used by rulesets, searches, URLs and human notes

Careful with VB/C# Cross Language



- Don't create dependency on both assemblies
- Consider that people work a bit differently
- Test in both languages
  - You might find unexpected differences in the semantic model

Use tag:analyzers in Your NuSpec File

- Include the tag "analyzers" in .nuspec file
  - This is the default
  - This helps people file your analyzer
- If your tool focuses on a particularly tool
  - Also include a tag for that library or tool

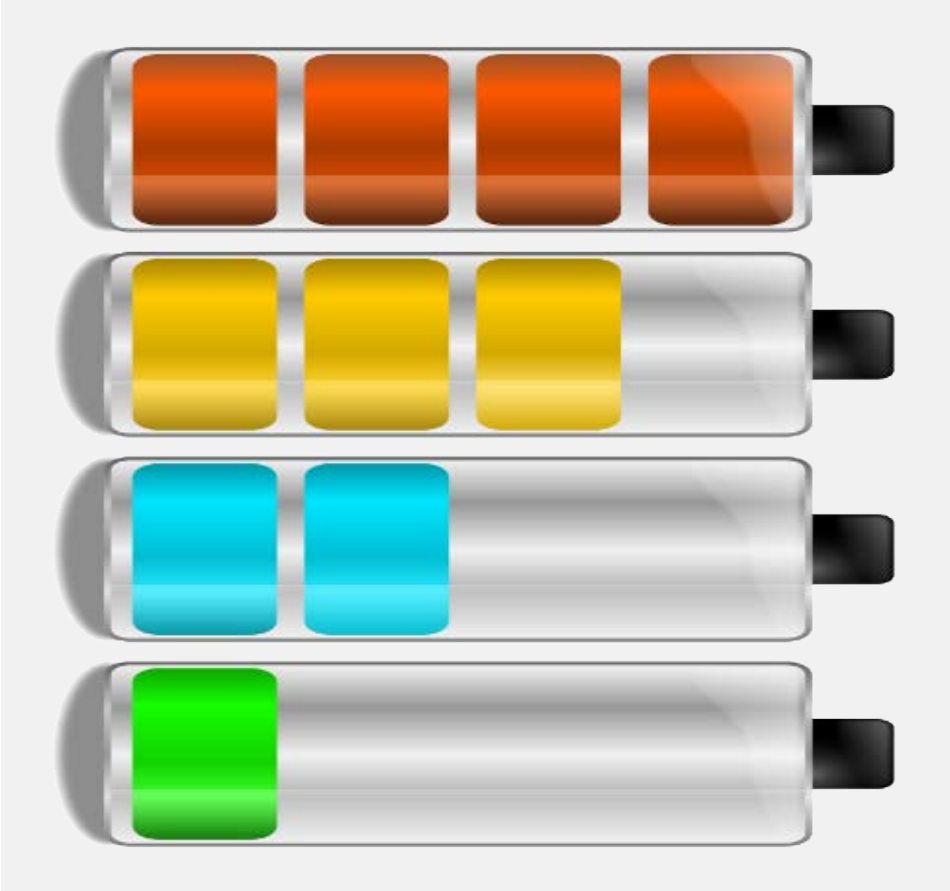
Recognize Opinion

- Opinionated analyzers are great
  - If it is the team opinion
  - If it's demonstrably better
- In order to be less opinionated
  - Hidden (lightbulb, no squiggle)
  - Not enabled by default
  - Use refactorings for alternate options

Recognize Opinion

- Opinionated analyzers are great
  - If it is the team opinion
  - If it's demonstrably better
- In order to be less opinionated
  - Hidden (lightbulb, no squiggle)
  - Not enabled by default
  - Use refactorings for alternate options
- This is not an opportunity for style wars

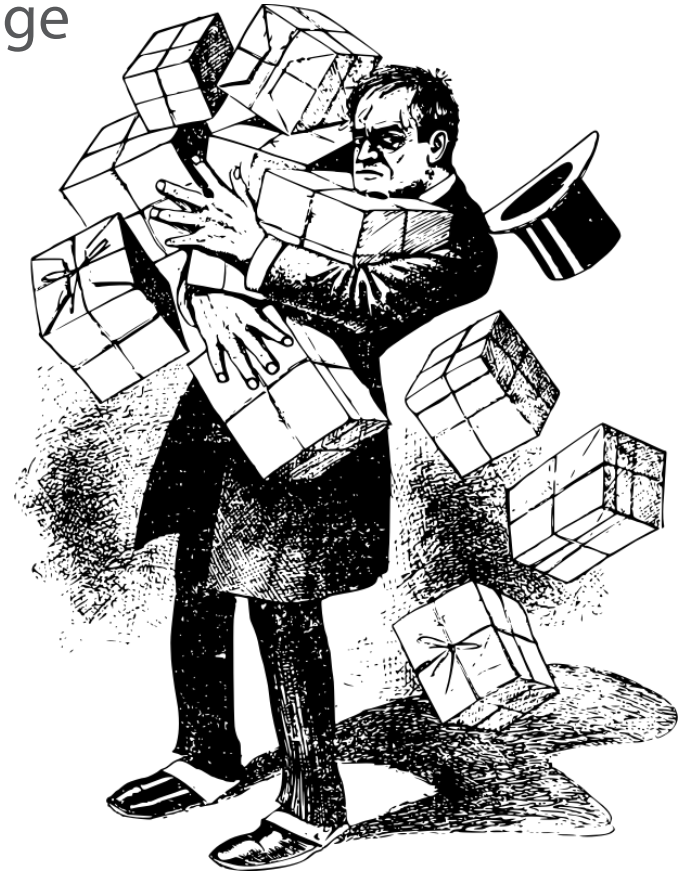
Use Severity and Refactoring Well

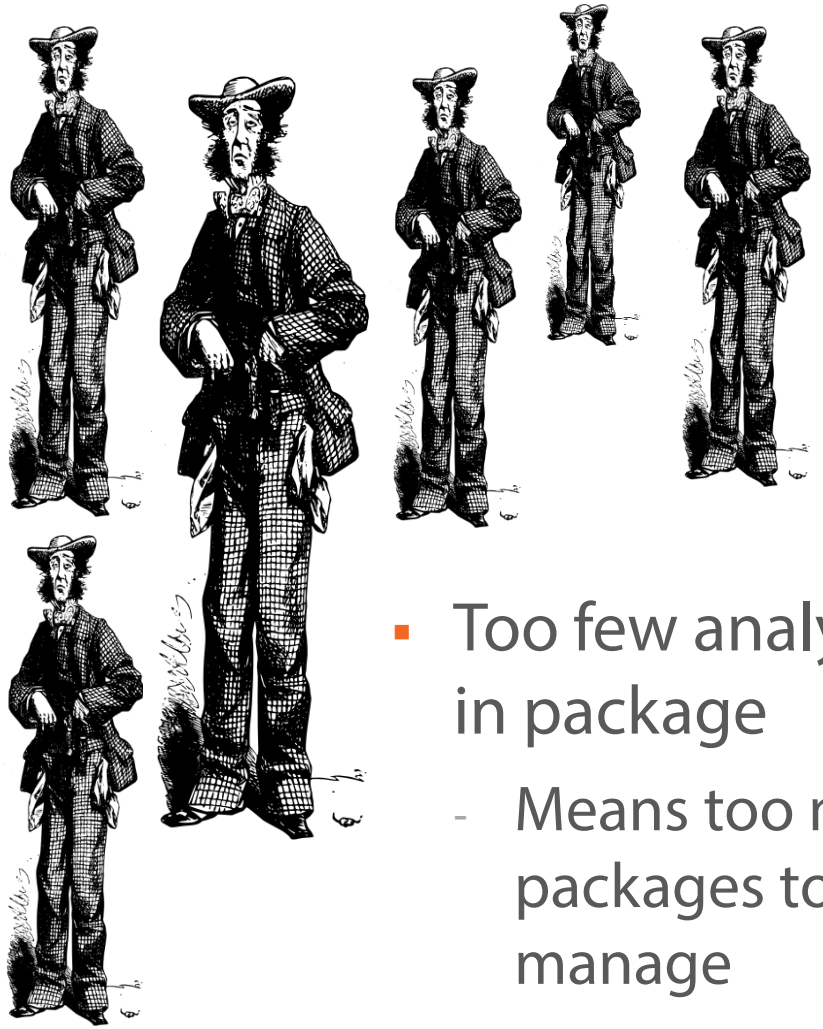


- Analyzer `Severity.Error` — often
  - Build error, red squiggle
- Analyzer `Severity.Warning` — often
  - Build warning, green squiggle
- Analyzer `Severity.Hidden` — often
  - No build message, no squiggle
- Refactoring
  - No action until user request

Package Grouping and Organization

- Too many analyzers in package
  - Hard to manage rulesets
  - Unwieldy





- Too few analyzers in package
  - Means too many packages to manage

Consistent Categories

- Use the categories from FxCop
  - Also called CodeAnalysis
- Rarely use other categories

Module Summary

Execution Order

Deployment

Localizing
(or not)

Guidelines

Due to multiple threads don't depend on execution order

- Except a handful of assurances, like start first and end last

The Readme.txt file contains deployment instructions

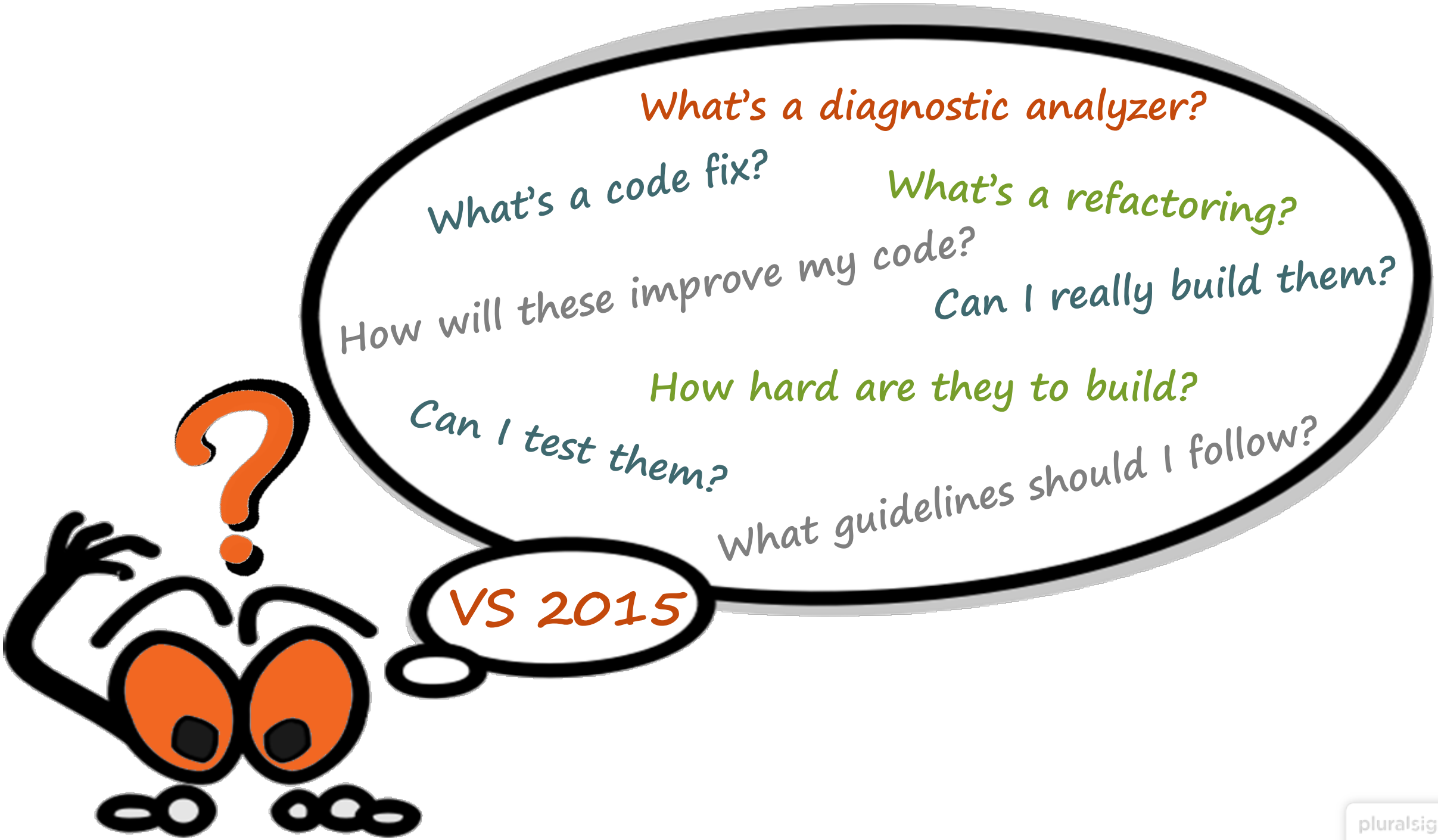
- Local NuGet feed for testing or team analyzers

Don't localize

- Use resource files to prepare for localization

A handful of guidelines will help you build better analyzers

- Don't let analyzers be an excuse for style wars


What's a diagnostic analyzer?
What's a code fix?
What's a refactoring?
How will these improve my code?
Can I really build them?
How hard are they to build?
Can I test them?
What guidelines should I follow?
VS 2015