Static and Runtime API Usage Analysis on .NET

Rufus Linke

18. August 2011

Outline

Introduction

Introduction

Overview

Analysis of

- the shape of existing APIs.
- their usage by "projects in the wild."

In terms of

- reusability of APIs.
- OO-specific usage.
- general acceptance and coverage.

Introduction

Overview

Analysis of

- the shape of existing APIs.
- their usage by "projects in the wild."

In terms of

- reusability of APIs.
- OO-specific usage.
- general acceptance and coverage.

Introduction

Overview

Analysis of

- the shape of existing APIs.
- their usage by "projects in the wild."

In terms of

- reusability of APIs.
- OO-specific usage.
- general acceptance and coverage.

Introduction

Overview

Analysis of

- the shape of existing APIs.
- their usage by "projects in the wild."

In terms of

- reusability of APIs.
- OO-specific usage.
- general acceptance and coverage.

Introduction

Overview

Analysis of

- the shape of existing APIs.
- their usage by "projects in the wild."

In terms of

- reusability of APIs.
- OO-specific usage.
- general acceptance and coverage.

- provide understanding of existing API design
- support research on API migration
- facilitate development of new APIs
- show up refactoring options

- provide understanding of existing API design
- support research on API migration
- facilitate development of new APIs
- show up refactoring options

- provide understanding of existing API design
- support research on API migration
- facilitate development of new APIs
- show up refactoring options

- provide understanding of existing API design
- support research on API migration
- facilitate development of new APIs
- show up refactoring options

Introduction

Contributions

- combined static and dynamic analysis of API design and usage
- application of analysis to a corpus of selected open-source software
- composition of metrics that are suitable to measure API related software usage

Introduction

Contributions

- combined static and dynamic analysis of API design and usage
- application of analysis to a corpus of selected open-source software
- composition of metrics that are suitable to measure API related software usage

- combined static and dynamic analysis of API design and usage
- application of analysis to a corpus of selected open-source software
- composition of metrics that are suitable to measure API related software usage

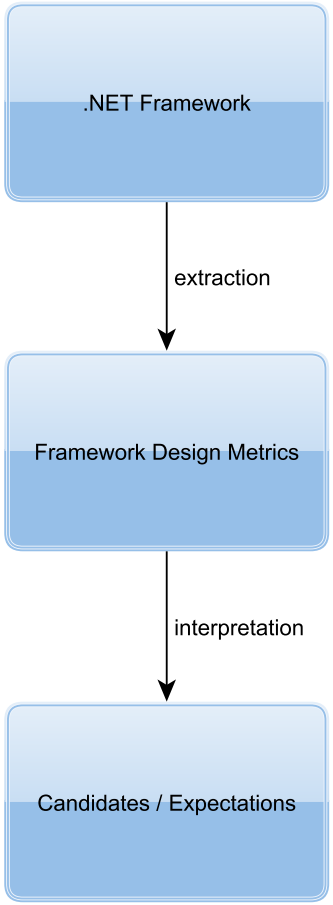
Methodology

Methodology

Overview

Explorative analysis approach:

1. framework design analysis

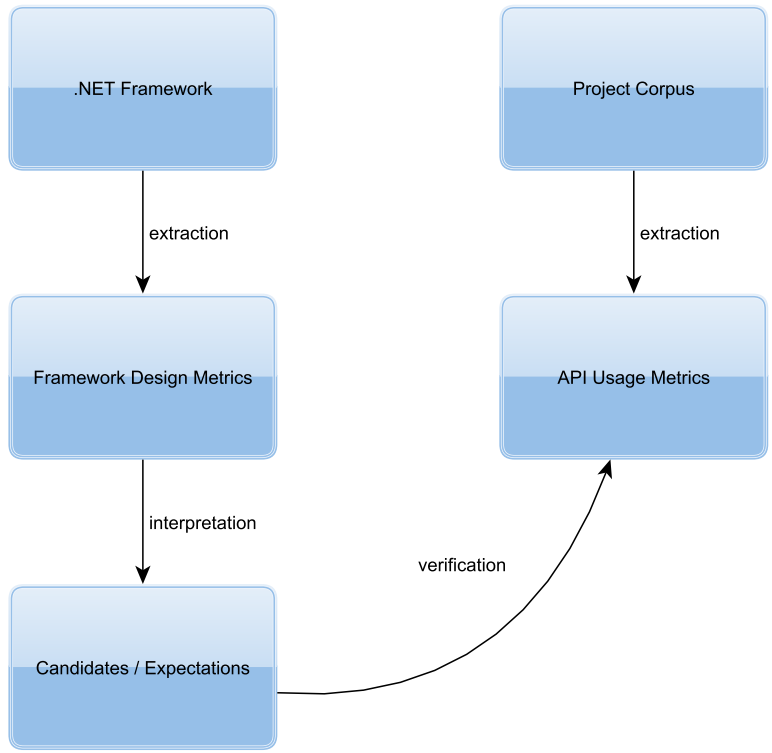


Methodology

Overview

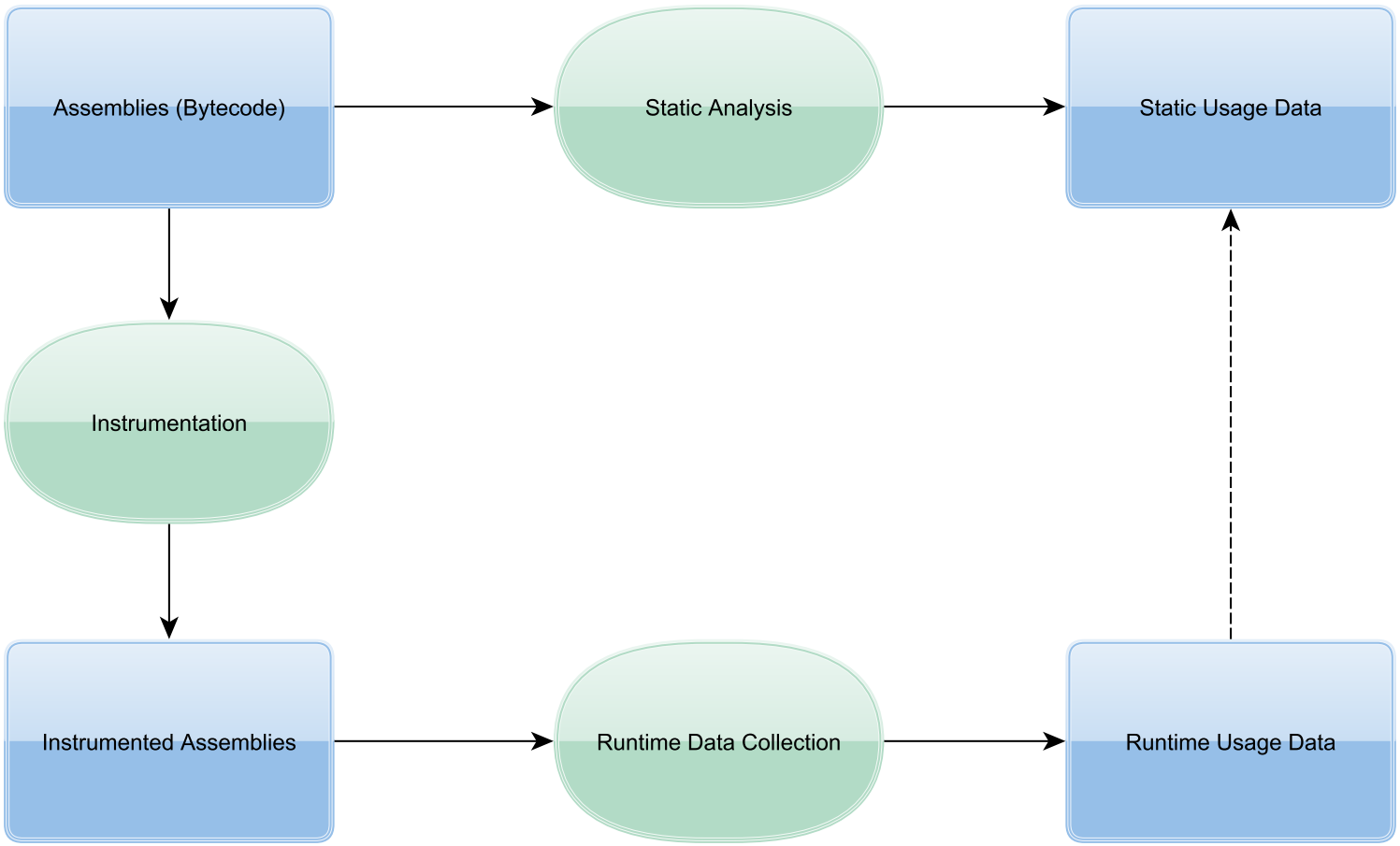
Explorative analysis approach:

1. framework design analysis
2. framework usage analysis



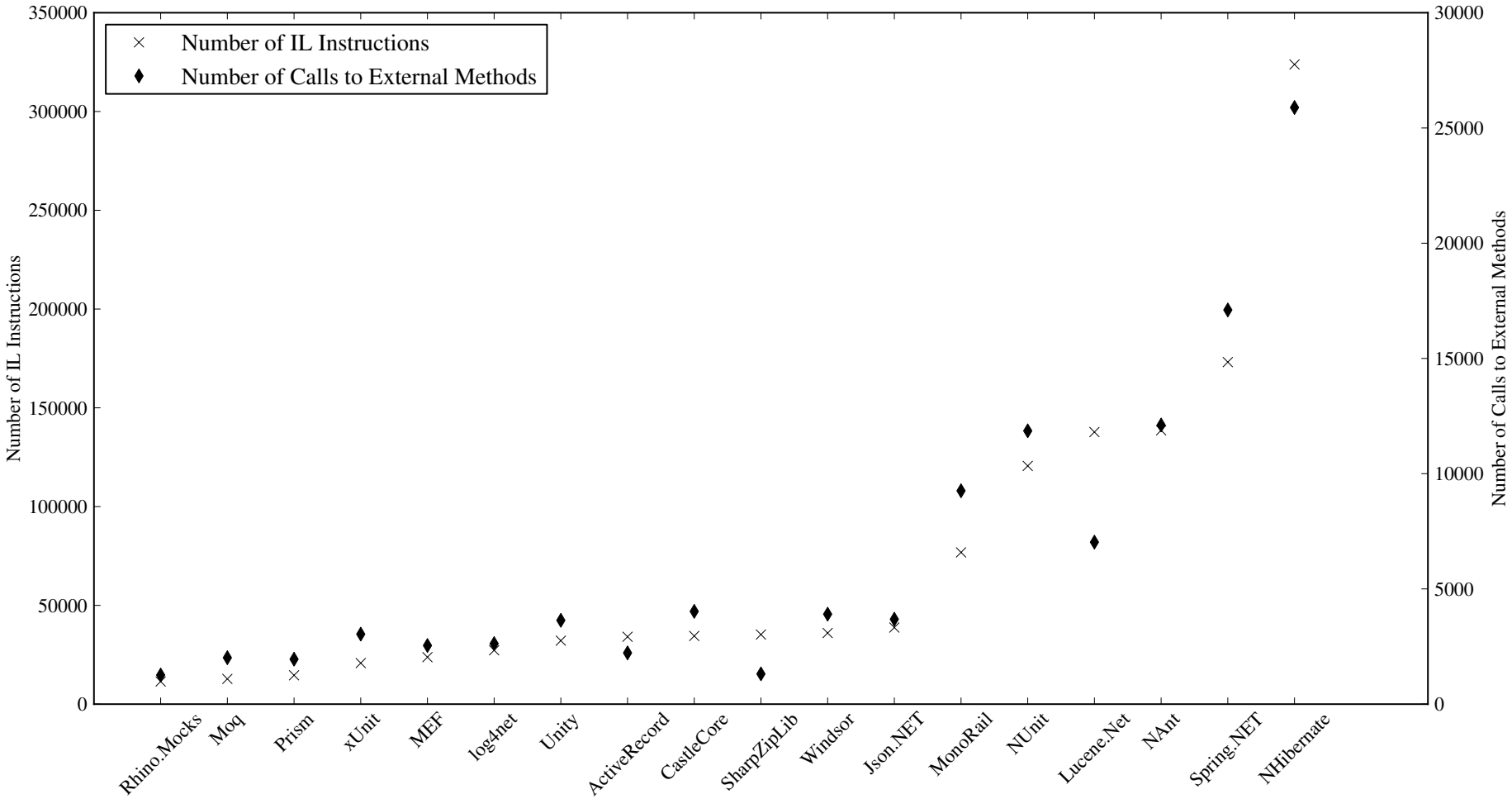
Methodology

Analysis Overview



Methodology

Corpus



Methodology

API

APIs are defined by their namespace.
Namespaces are grouped for easier handling.

Example

The XML API is covered by the namespaces

- `System.Xml`
- `System.Xml.Schema`
- `System.Xml.Serialization`
- ...

All these namespaces are grouped and represented as

- `System.Xml.*`

Methodology

API

APIs are defined by their namespace.
Namespaces are grouped for easier handling.

Example

The XML API is covered by the namespaces

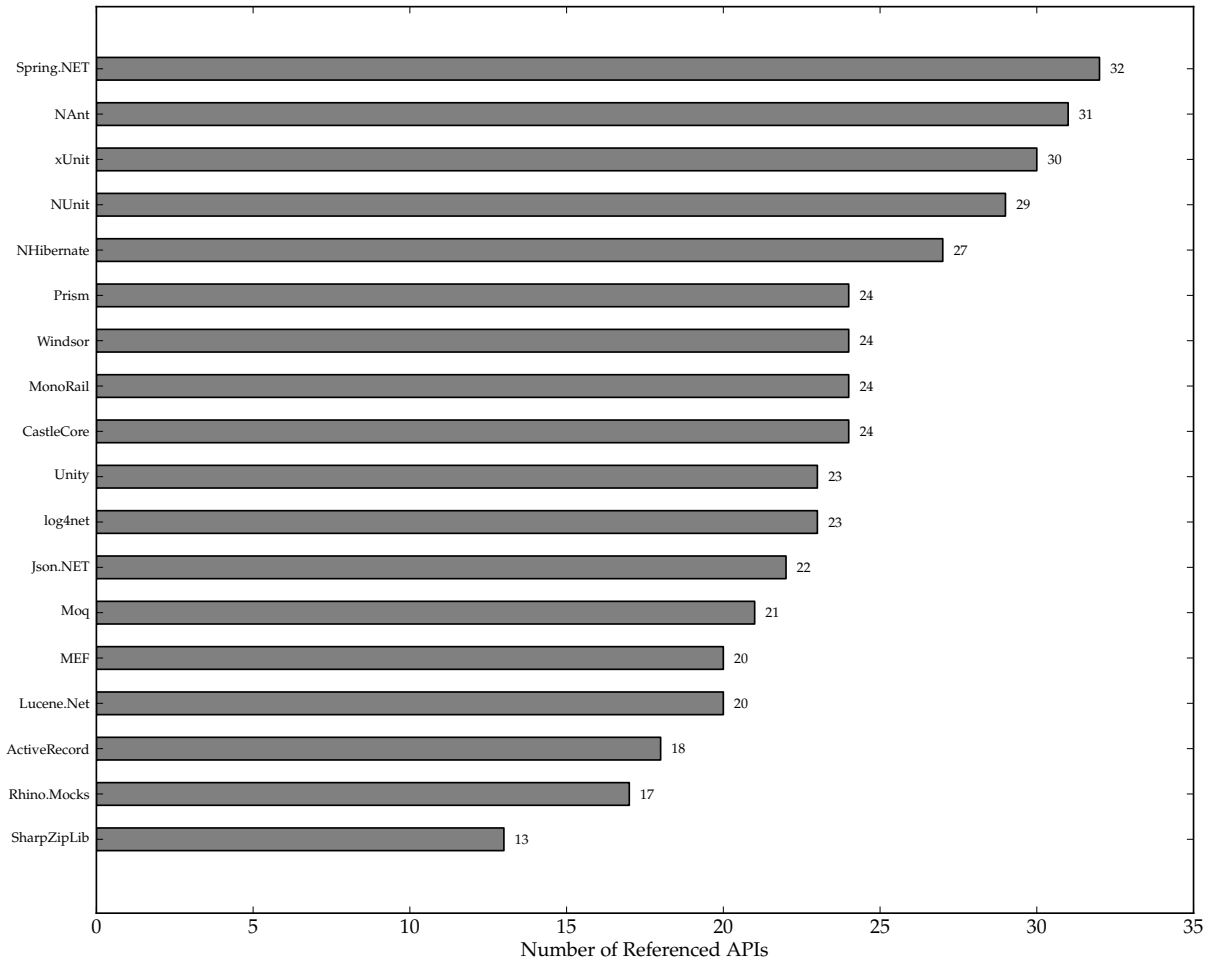
- `System.Xml`
- `System.Xml.Schema`
- `System.Xml.Serialization`
- ...

All these namespaces are grouped and represented as

- `System.Xml.*`

Methodology

Corpus



API Usage

The term “API usage” refers to “usage by the developer”.

Example

```
class SomeClass { public SomeClass() { } }
```

Implicitly calls System.Object::.ctor() ⇒ not API usage.

API Usage

The term "API usage" refers to "usage by the developer".

Example

```
class SomeClass { public SomeClass() { } }
```

Implicitly calls System.Object::.ctor() ⇒ not API usage.

Results

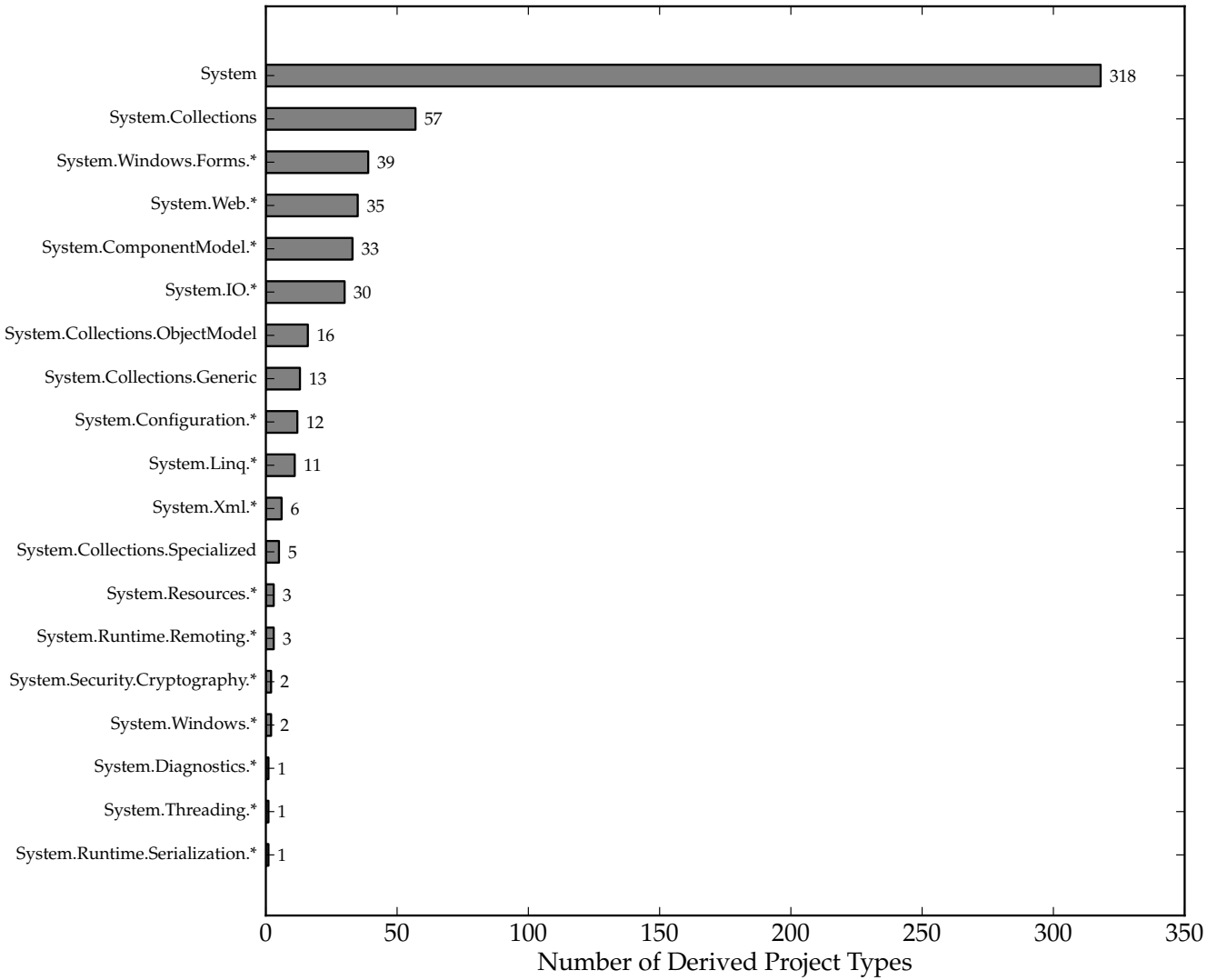
.NET Framework

Design Metrics

General statistics

- namespaces: 74 (3 with only sealed types)
- number of classes: 10457 (4295 sealed classes)
- interfaces: 957

Specialization of types from the .NET APIs



.NET Framework

Design Metrics

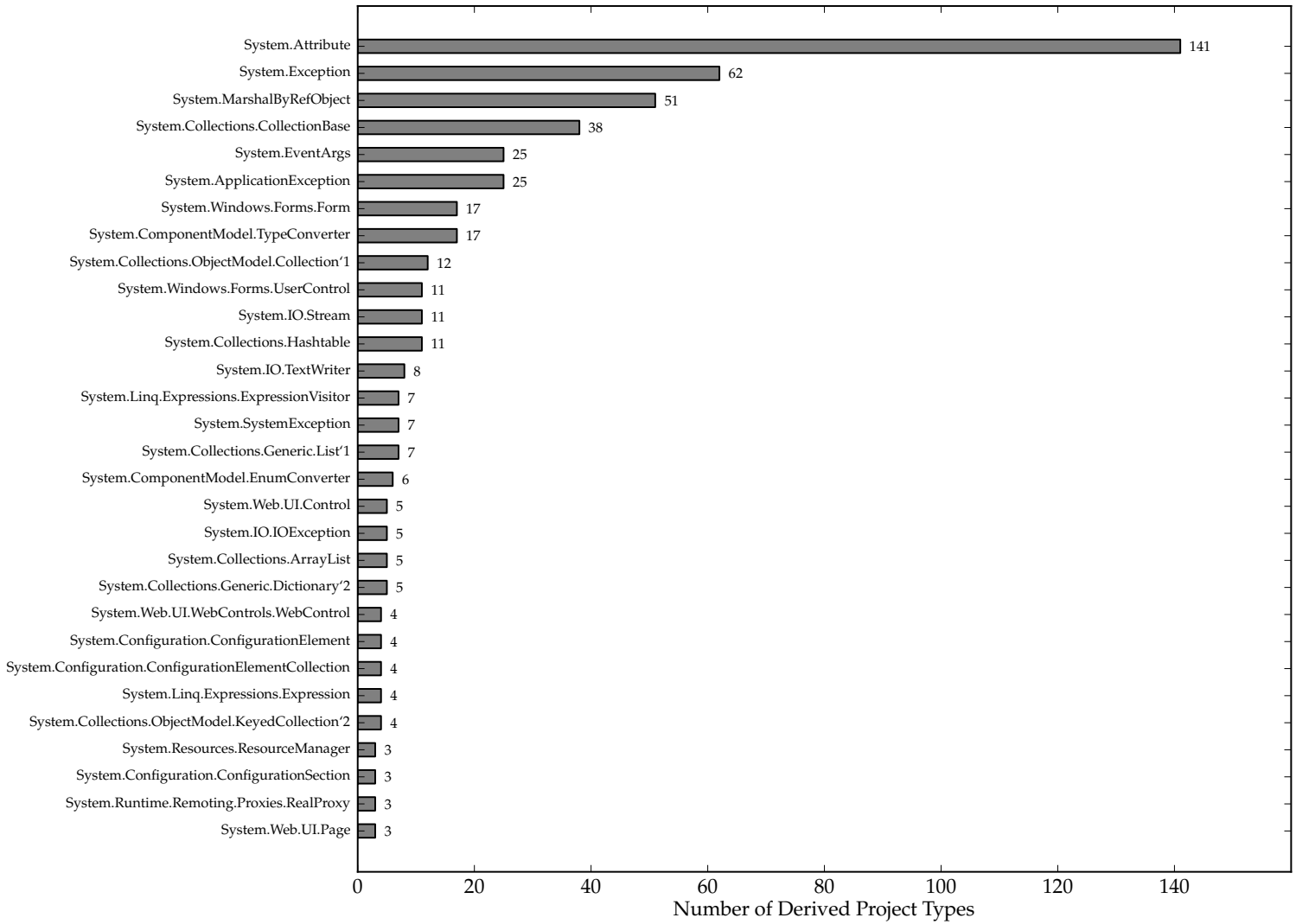
General statistics

- namespaces: 74 (3 with only sealed types)
  - 19 namespaces have types that were specialized (26,76%)
- number of classes: 10457 (4295 sealed classes)
- interfaces: 957

Corpus

Usage Metrics

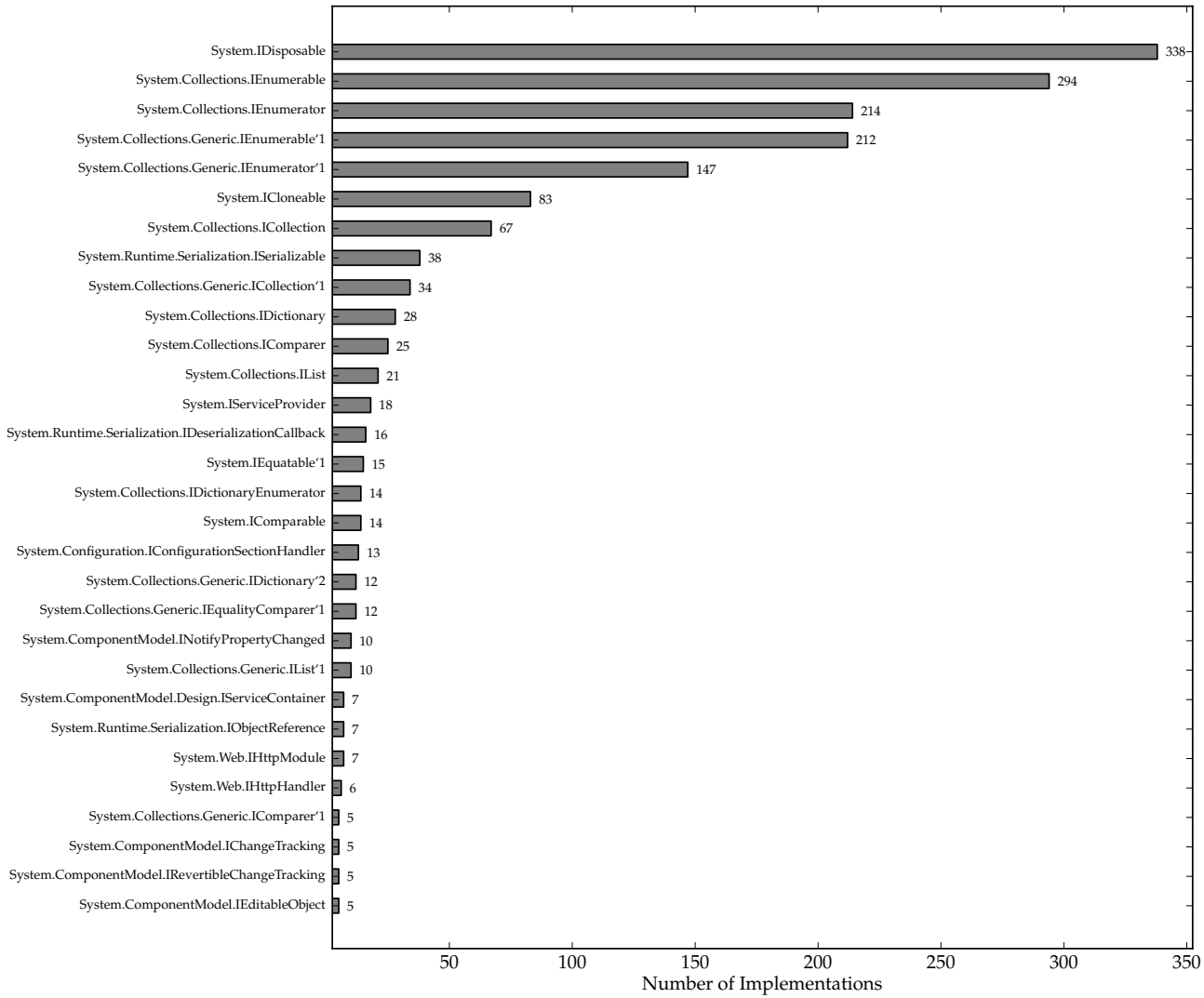
.NET types specialized by corpus



General statistics

- namespaces: 74 (3 with only sealed types)
  - 19 namespaces have types that were specialized (26.76%)
- number of classes: 10457 (4295 sealed classes)
  - 100 types are specialized (1.62%)
- interfaces: 957

Implemented interfaces



General statistics

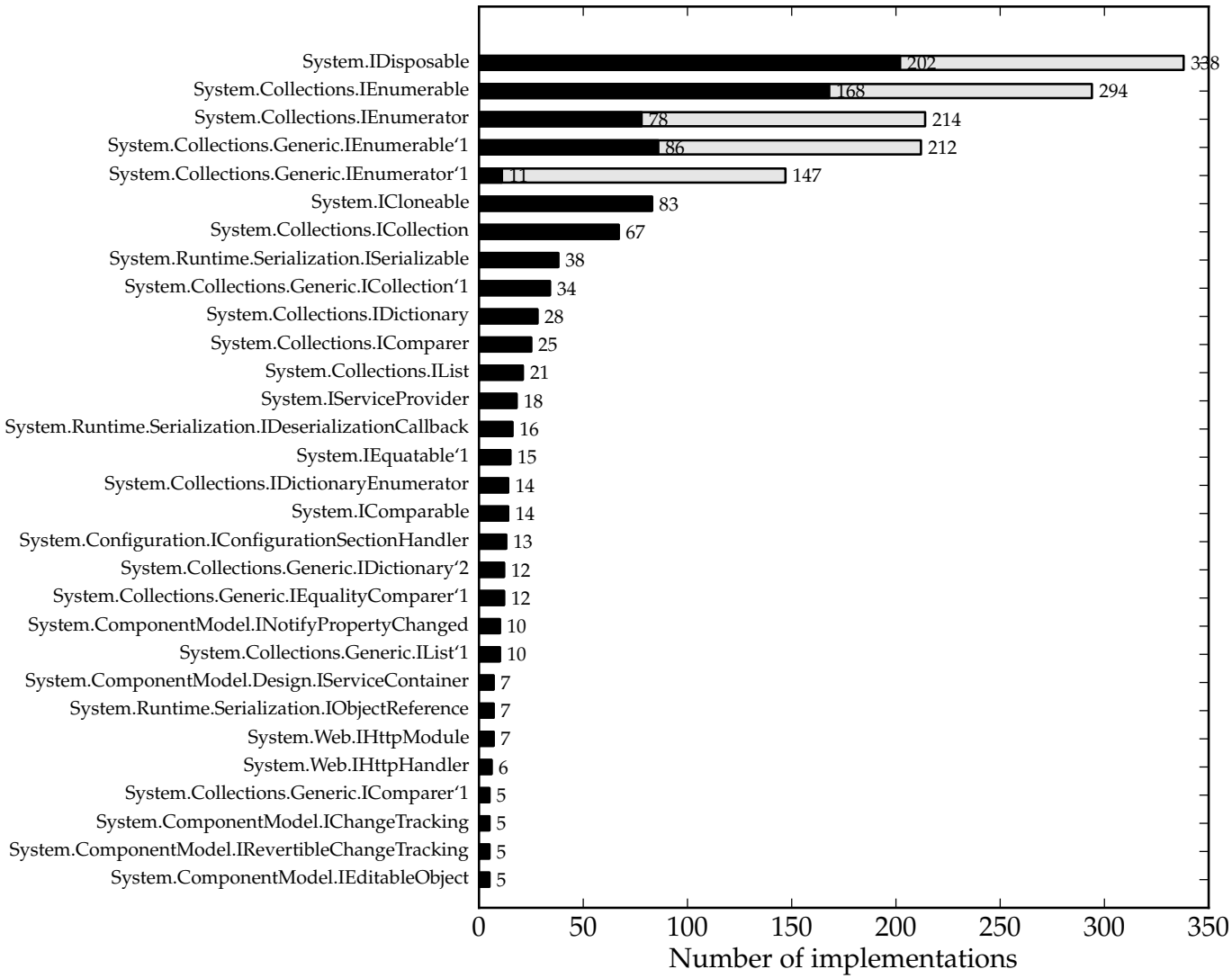
- namespaces: 74 (3 with only sealed types)
  - 19 namespaces have types that were specialized (26.76%)
- number of classes: 10457 (4295 sealed classes)
  - 100 types are specialized (1.62%)
- interfaces: 957
  - 70 interfaces are implemented (7.31%)

Compiler generated interface implementations

A simple number generator in the form of an `IEnumerable`1` can be generated with the following code:

```
IEnumerable<int> GetSomeNumbers(int num)
{
    Random r = new Random();
    for (int i = 0; i < num; i++)
    {
        yield return r.Next();
    }
}
```

Implemented interfaces (without compiler generated)



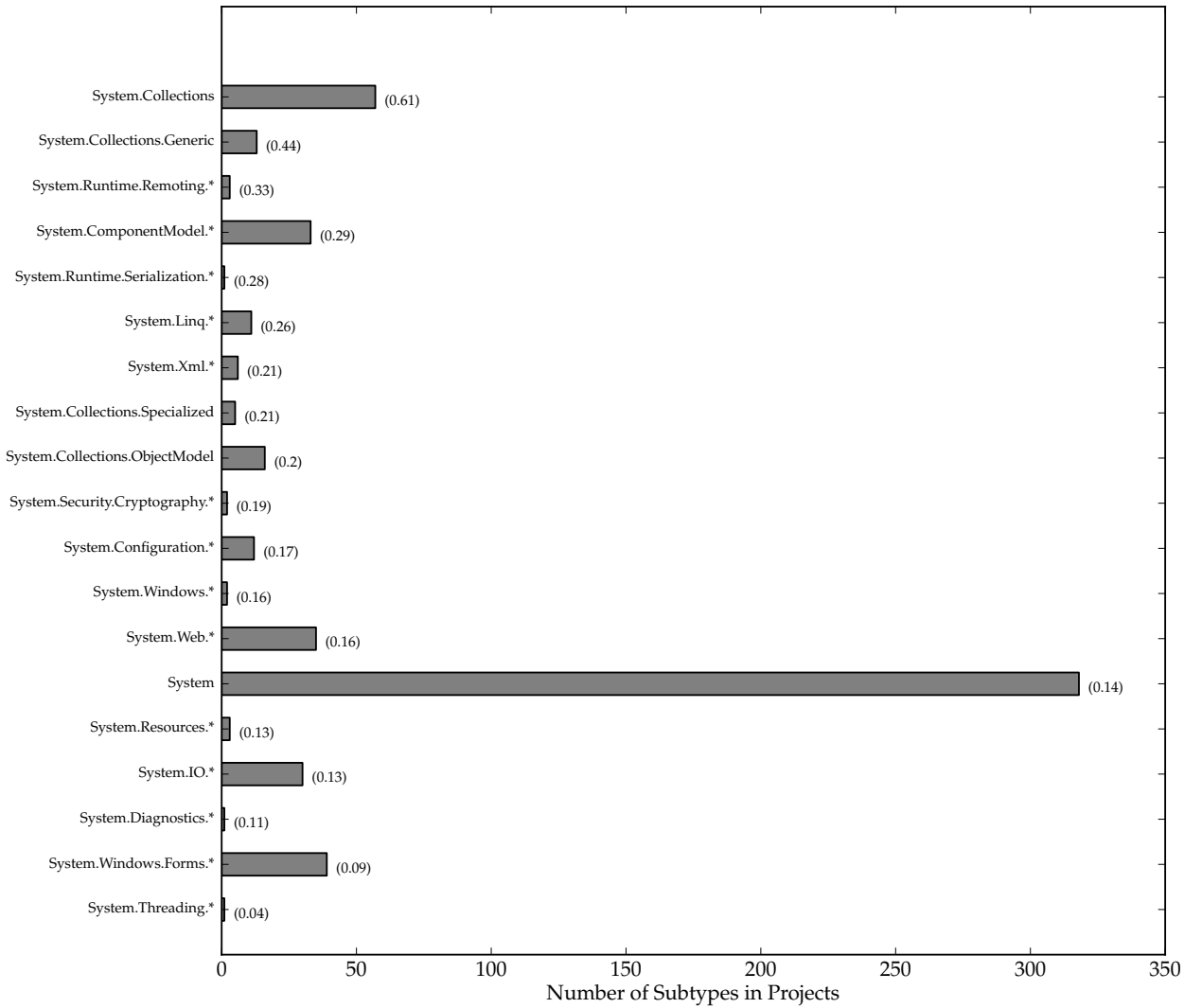
Abstractness of .NET namespaces

Namespace	ConcreteTypes	AbstractTypes	Abstractness
Accessibility	1	6	0.86
System.Dynamic	4	16	0.80
System.AddIn.*	14	27	0.66
System.Collections	9	14	0.61
System.Runtime.InteropServices.*	68	80	0.54
Microsoft.VisualC.*	21	17	0.45
System.Collections.Generic	14	11	0.44
System.Collections.Concurrent	5	3	0.38
System.Security.AccessControl	33	17	0.34
System.Runtime.Remoting.*	108	54	0.33
System.Runtime.Caching.*	14	7	0.33
System.Runtime.ConstrainedExecution	2	1	0.33
System.Xaml.*	37	16	0.30
System.Text	17	7	0.29
System.ComponentModel.*	280	114	0.29
System.Runtime.Serialization.*	46	18	0.28
System.Security	19	7	0.27
Microsoft.JScript.*	140	51	0.27
System.Linq.*	39	14	0.26
System.IdentityModel.*	67	23	0.26

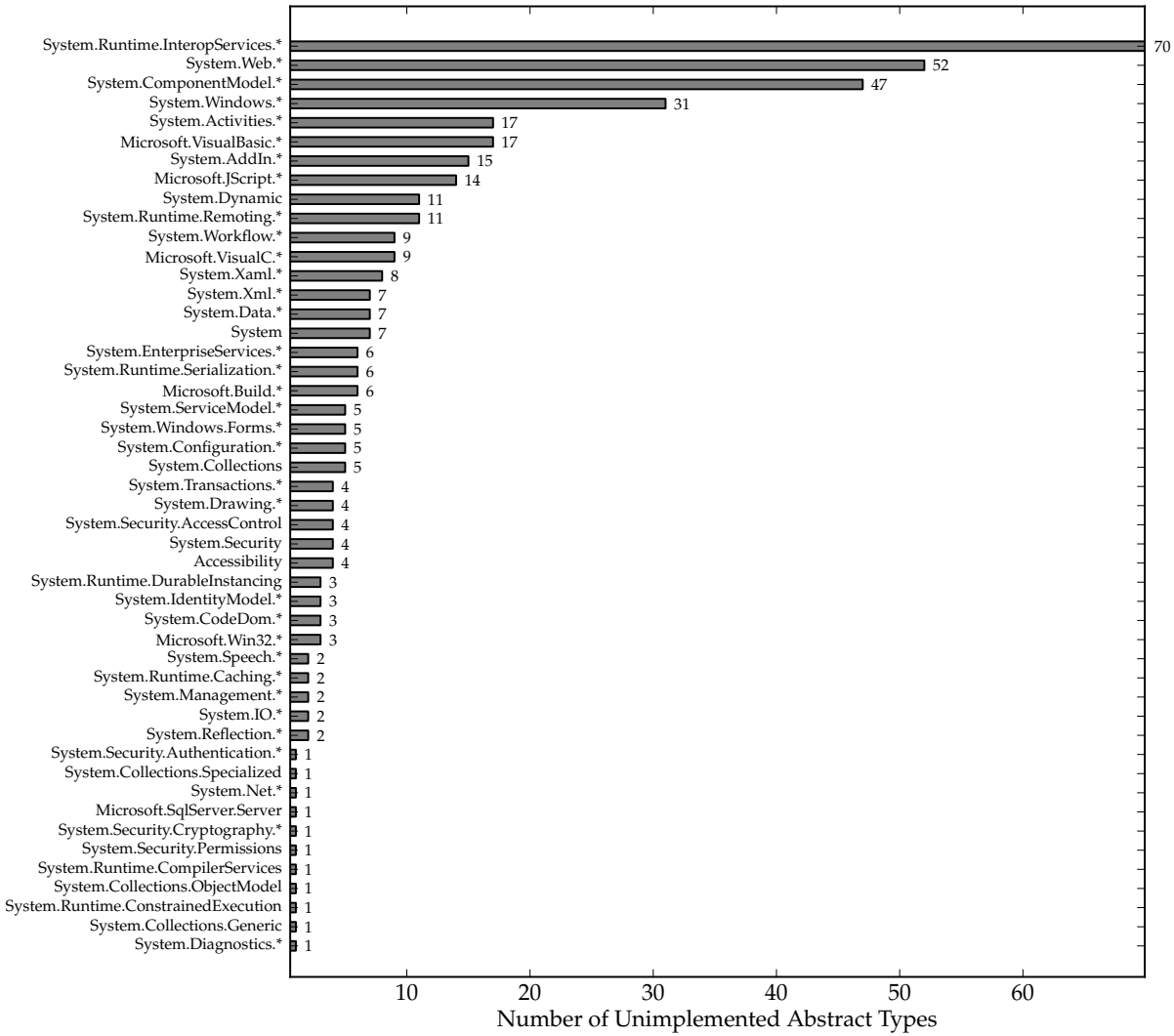
Corpus

Usage Metrics

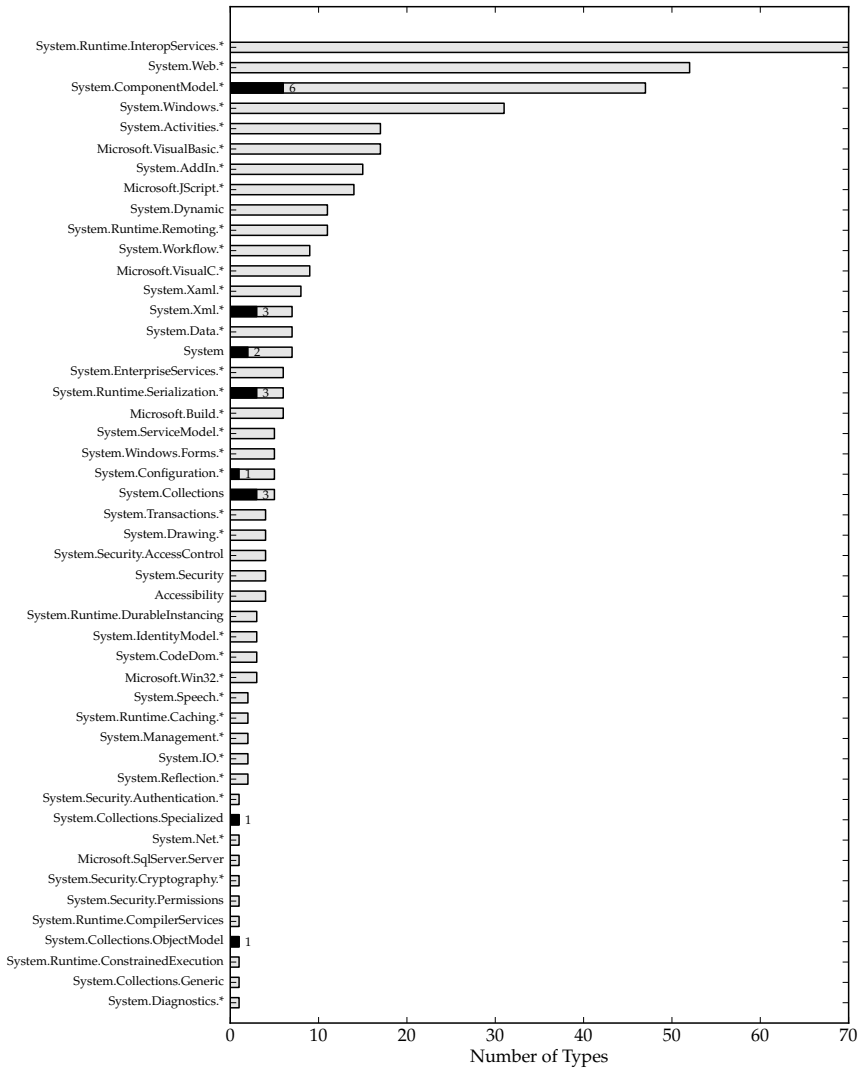
Comparison of API abstractness to number of specializations



Unimplemented abstract types



Previously unimplemented abstract types



Receiver Types

- static receiver type
- runtime receiver type

Example

```
Exception ex = new InvalidOperationException();
string s = ex.ToString();
```

static receiver type: `Exception`
runtime receiver type: `InvalidOperationException`

Receiver Types

- static receiver type
- runtime receiver type

Example

```
Exception ex = new InvalidOperationException();
string s = ex.ToString();
```

static receiver type: `Exception`
runtime receiver type: `InvalidOperationException`

Receiver Types

- static receiver type
- runtime receiver type

Example

```
Exception ex = new InvalidOperationException();
string s = ex.ToString();
```

static receiver type: `Exception`
runtime receiver type: `InvalidOperationException`

Corpus

Usage Metrics

Late-bound methods

	Static Data	Additional Runtime Data			
Namespace	Methods	Methods	% Public API	% Internal API	% Project
System	493	506	24.90	18.38	56.72
System.Collections	130	488	22.75	32.17	45.08
System.Collections.Generic	105	327	10.70	18.65	70.64
System.Reflection.*	233	111	0.90	99.10	0.00
System.Data.*	123	97	77.32	0.00	22.68
System.Xml.*	354	82	45.12	54.88	0.00
System.IO.*	203	51	45.10	17.65	37.25
System.ComponentModel.*	79	25	8.00	8.00	84.00
System.Linq.*	55	16	37.50	50.00	12.50
System.Runtime.Remoting.*	18	12	0.00	100.00	0.00
System.CodeDom.*	47	9	66.67	33.33	0.00
System.Text	44	9	77.78	22.22	0.00
System.Configuration.*	31	8	25.00	12.50	62.50
System.Net.*	71	7	100.00	0.00	0.00
System.Windows.Forms.*	446	3	0.00	100.00	0.00
System.Web.*	382	3	66.67	33.33	0.00
System.Security.Principal	8	3	100.00	0.00	0.00
System.Windows.*	34	2	0.00	0.00	100.00
System.Runtime.Serialization.*	25	2	50.00	0.00	50.00

Examples for late-bound receiver methods

Public API

```
IDisposable file = new FileStream(...);
file.Close();
```

Internal API

```
// Object.GetType() returns an object
// of type RuntimeType, which is internal.
Type type = file.GetType();
```

Examples for late-bound receiver methods

Public API

```
IDisposable file = new FileStream(...);
file.Close();
```

Internal API

```
// Object.GetType() returns an object
// of type RuntimeType, which is internal.
Type type = file.GetType();
```

Examples for late-bound receiver methods

Project

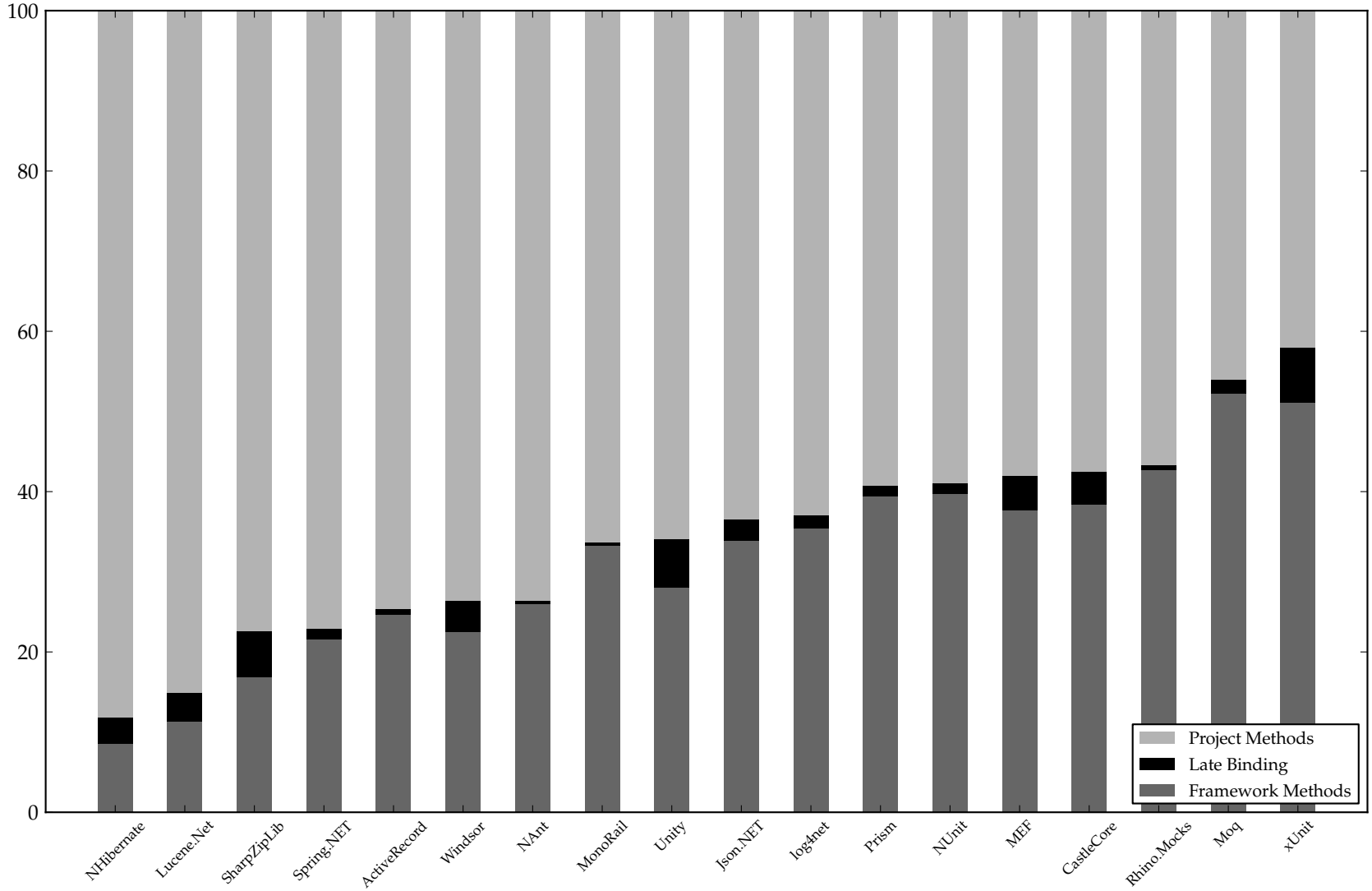
```
class ProjectClass : ICloneable { ... }

ICloneable original = new ProjectClass();
object clone = c.Clone();
```

Corpus

Usage Metrics

Distribution of method calls in the corpus



Conclusion and Further Work

Conclusion

- not much framework type specialization
  - often limited to special types
- tendency towards interfaces over type specialization
  - late-binding used with interfaces
- corpus mostly does not meet usage expectations from framework design
- many calls into API code, but primarily “library-like” usage

Conclusion

- not much framework type specialization
  - often limited to special types
- tendency towards interfaces over type specialization
  - late-binding used with interfaces
- corpus mostly does not meet usage expectations from framework design
- many calls into API code, but primarily “library-like” usage

Conclusion

- not much framework type specialization
  - often limited to special types
- tendency towards interfaces over type specialization
  - late-binding used with interfaces
- corpus mostly does not meet usage expectations from framework design
- many calls into API code, but primarily "library-like" usage

Conclusion

- not much framework type specialization
  - often limited to special types
- tendency towards interfaces over type specialization
  - late-binding used with interfaces
- corpus mostly does not meet usage expectations from framework design
- many calls into API code, but primarily “library-like” usage

Further Work

- automated test suites for runtime analysis
- specialized analyses for smaller sets of APIs
- measure other API usage forms
- comparison to Java APIs
- acceptance of new framework features

Further Work

- automated test suites for runtime analysis
- specialized analyses for smaller sets of APIs
- measure other API usage forms
- comparison to Java APIs
- acceptance of new framework features

Further Work

- automated test suites for runtime analysis
- specialized analyses for smaller sets of APIs
- measure other API usage forms
- comparison to Java APIs
- acceptance of new framework features

Further Work

- automated test suites for runtime analysis
- specialized analyses for smaller sets of APIs
- measure other API usage forms
- comparison to Java APIs
- acceptance of new framework features

Further Work

- automated test suites for runtime analysis
- specialized analyses for smaller sets of APIs
- measure other API usage forms
- comparison to Java APIs
- acceptance of new framework features

Thanks for your attention!