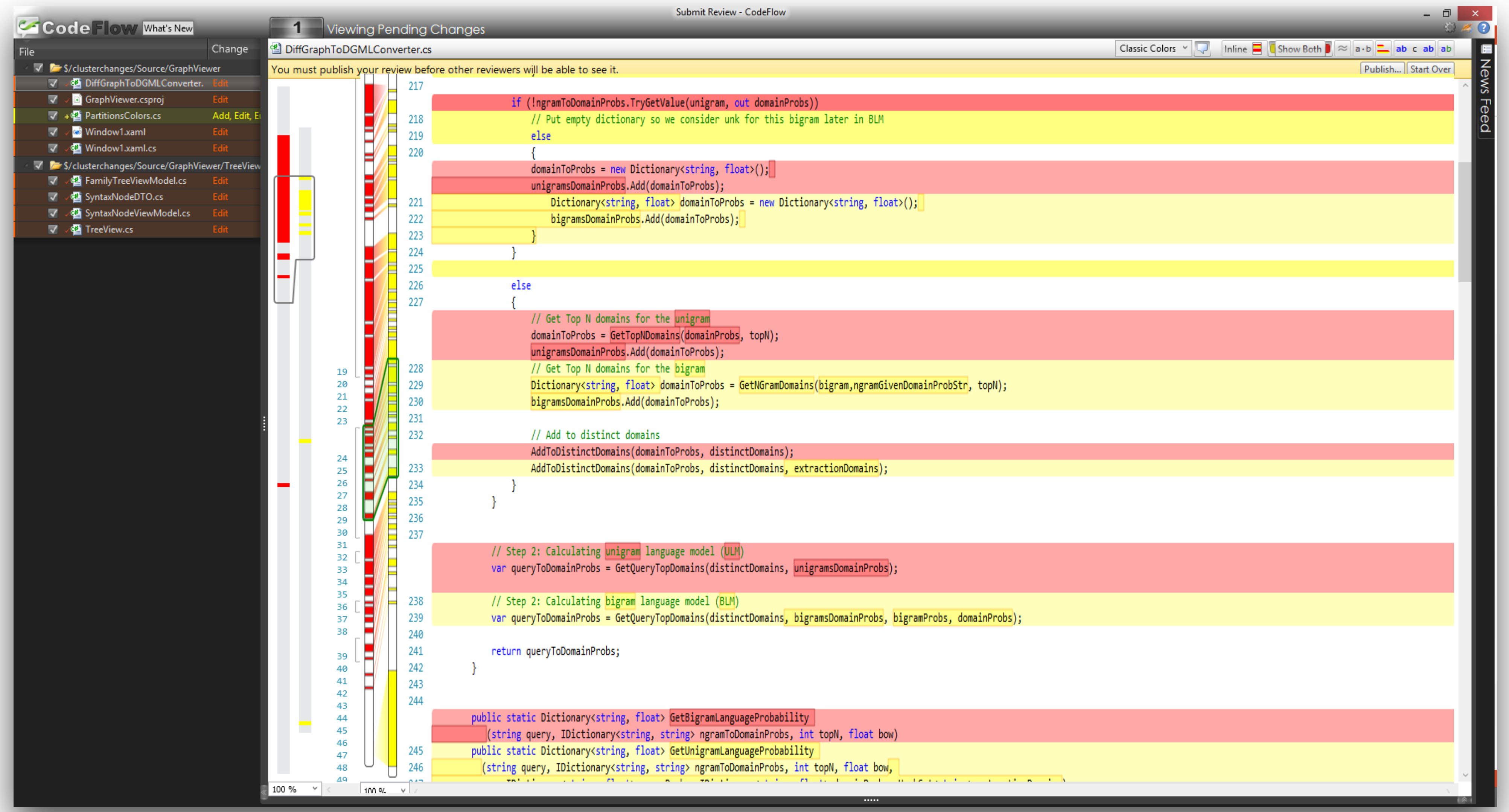
HELPING DEVELOPERS HELP THEMSELVES: AUTOMATIC DECOMPOSITION OF CODE REVIEW CHANGES

MIKE BARNETT, CHRISTIAN BIRD, AND SHUVENDU K. LAHIRI
MICROSOFT RESEARCH

JOÃO BRUNET
FEDERAL UNIVERSITY OF CAMPINA GRANDE (UFCG), BRAZIL

The code review process



File	Change
$/clusterchanges/Source/GraphViewer	
DiffGraphToDGMLConverter.	Edit
GraphViewer.csproj	Edit
PartitionsColors.cs	Add, Edit, E
Window1.xaml	Edit
Window1.xaml.cs	Edit
$/clusterchanges/Source/GraphViewer/TreeView	
FamilyTreeViewModel.cs	Edit
SyntaxNodeDTO.cs	Edit
SyntaxNodeViewModel.cs	Edit
TreeView.cs	Edit

DiffGraphToDGMLConverter.cs

You must publish your review before other reviewers will be able to see it.

```
            if (!ngramToDomainProbs.TryGetValue(unigram, out domainProbs))
                // Put empty dictionary so we consider unk for this bigram later in BLM
                else
                {
                domainToProbs = new Dictionary<string, float>();
                unigramsDomainProbs.Add(domainToProbs);
                    Dictionary<string, float> domainToProbs = new Dictionary<string, float>();
                    bigramsDomainProbs.Add(domainToProbs);
                }
            }

            else
            {
                // Get Top N domains for the unigram
                domainToProbs = GetTopNDomains(domainProbs, topN);
                unigramsDomainProbs.Add(domainToProbs);
                // Get Top N domains for the bigram
                Dictionary<string, float> domainToProbs = GetNGramDomains(bigram,ngramGivenDomainProbStr, topN);
                bigramsDomainProbs.Add(domainToProbs);

                // Add to distinct domains
                AddToDistinctDomains(domainToProbs, distinctDomains);
                AddToDistinctDomains(domainToProbs, distinctDomains, extractionDomains);
            }
        }


        // Step 2: Calculating unigram language model (ULM)
        var queryToDomainProbs = GetQueryTopDomains(distinctDomains, unigramsDomainProbs);

        // Step 2: Calculating bigram language model (BLM)
        var queryToDomainProbs = GetQueryTopDomains(distinctDomains, bigramsDomainProbs, bigramProbs, domainProbs);

        return queryToDomainProbs;
    }


    public static Dictionary<string, float> GetBigramLanguageProbability
            (string query, IDictionary<string, string> ngramToDomainProbs, int topN, float bow)
    public static Dictionary<string, float> GetUnigramLanguageProbability
        (string query, IDictionary<string, string> ngramToDomainProbs, int topN, float bow,
```

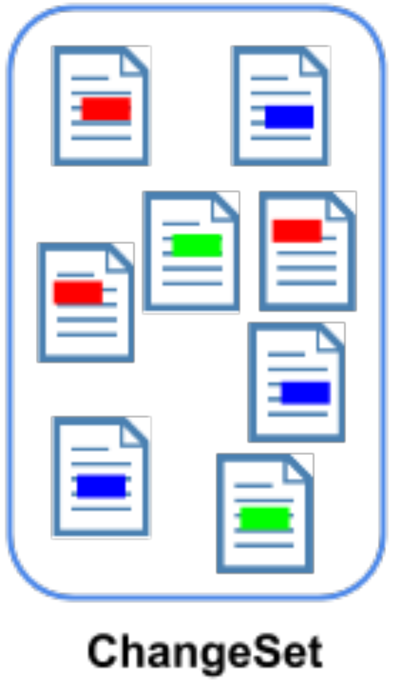
100 %

CodeFlow

25,000 developers
100,000 reviews per month

The Problem

Developers commit loosely related changes.



“Fixed #244 by adding inmethod binders in between properties and indexers. Also refactors expression-bodied member semantic model tests into their own file and adds some extra tests.”

It is difficult to review unrelated changes at once.

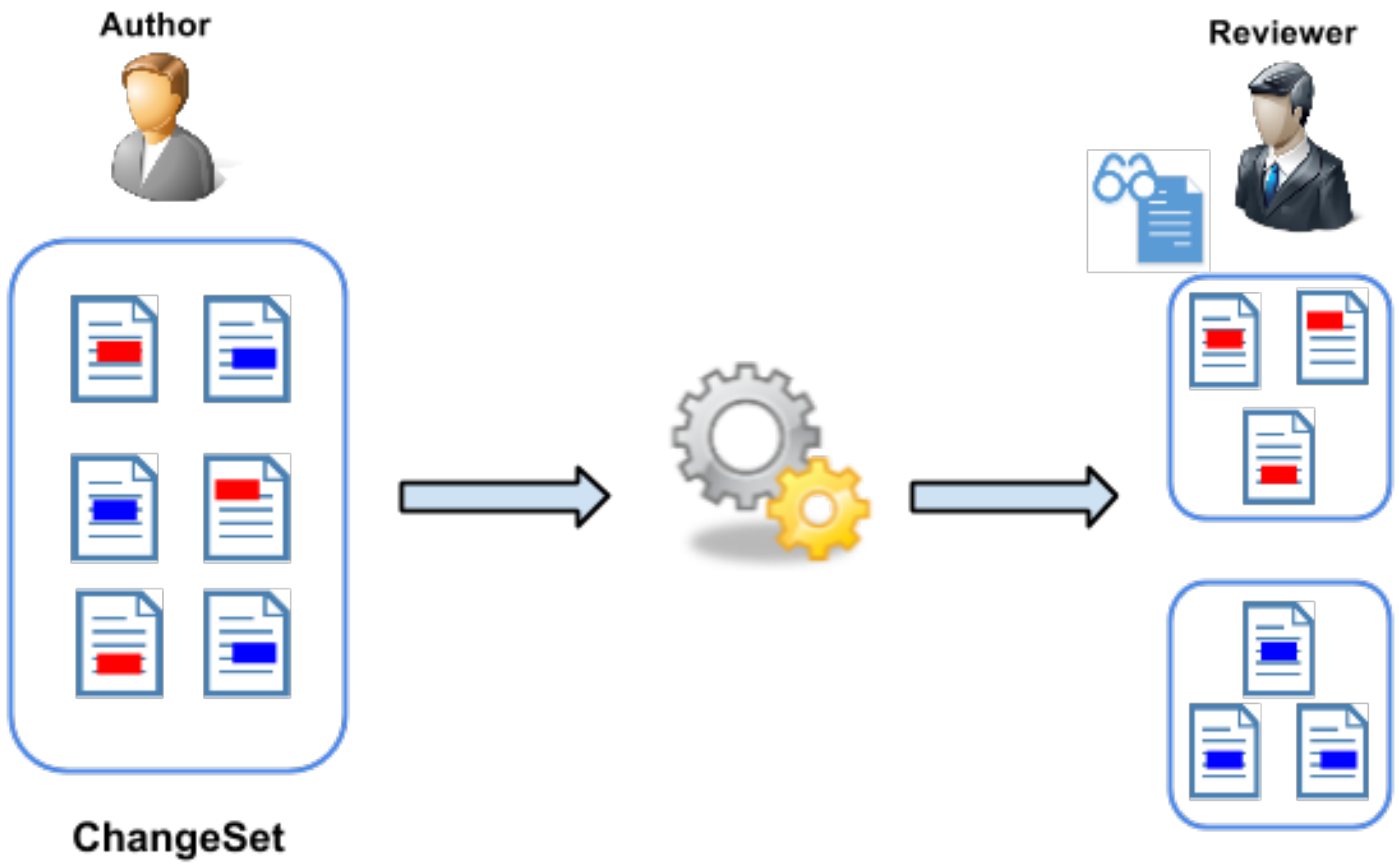
It is difficult to review unrelated changes at once.

Literature said so

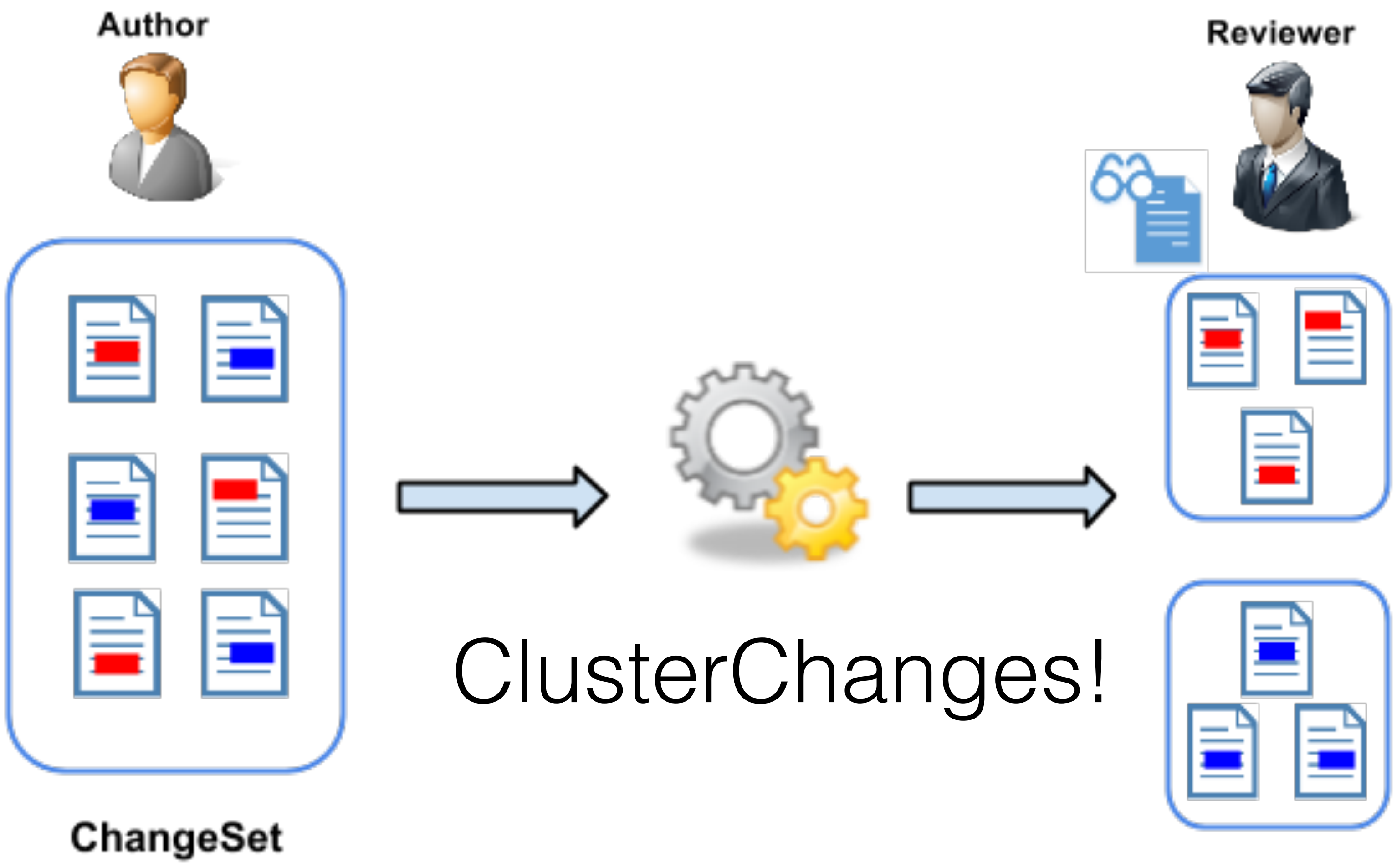
“The more files and diffs the more relatively independent changes” [1]

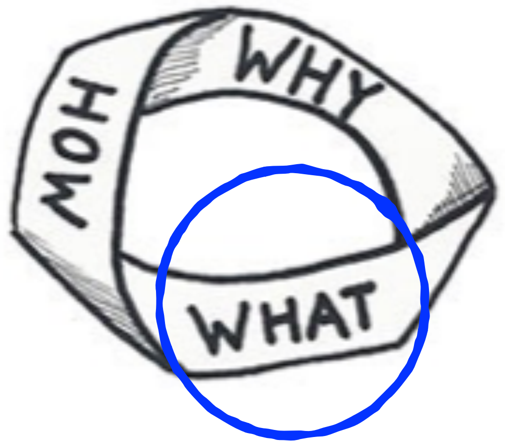
“Participants call for a tool that can automatically decompose a change into separate sub-changes“ [2]

Wouldn’t it be nice?



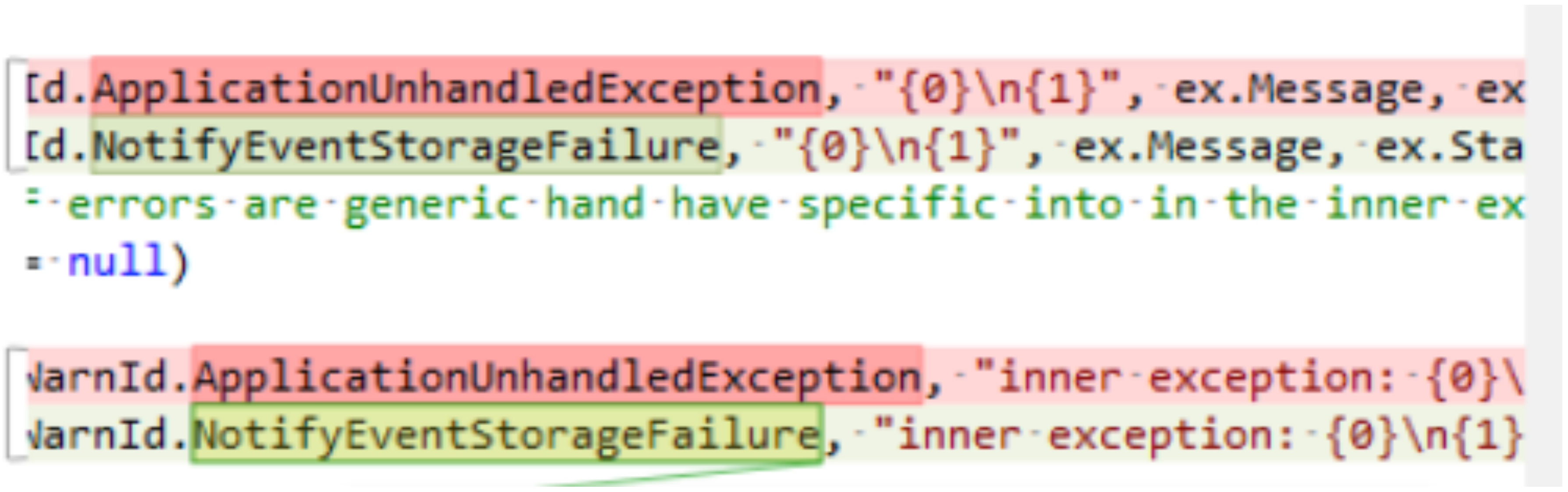
Wouldn't it be nice?



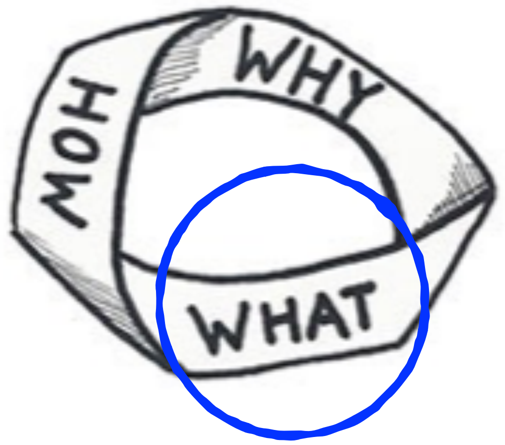


ClusterChanges

diff regions

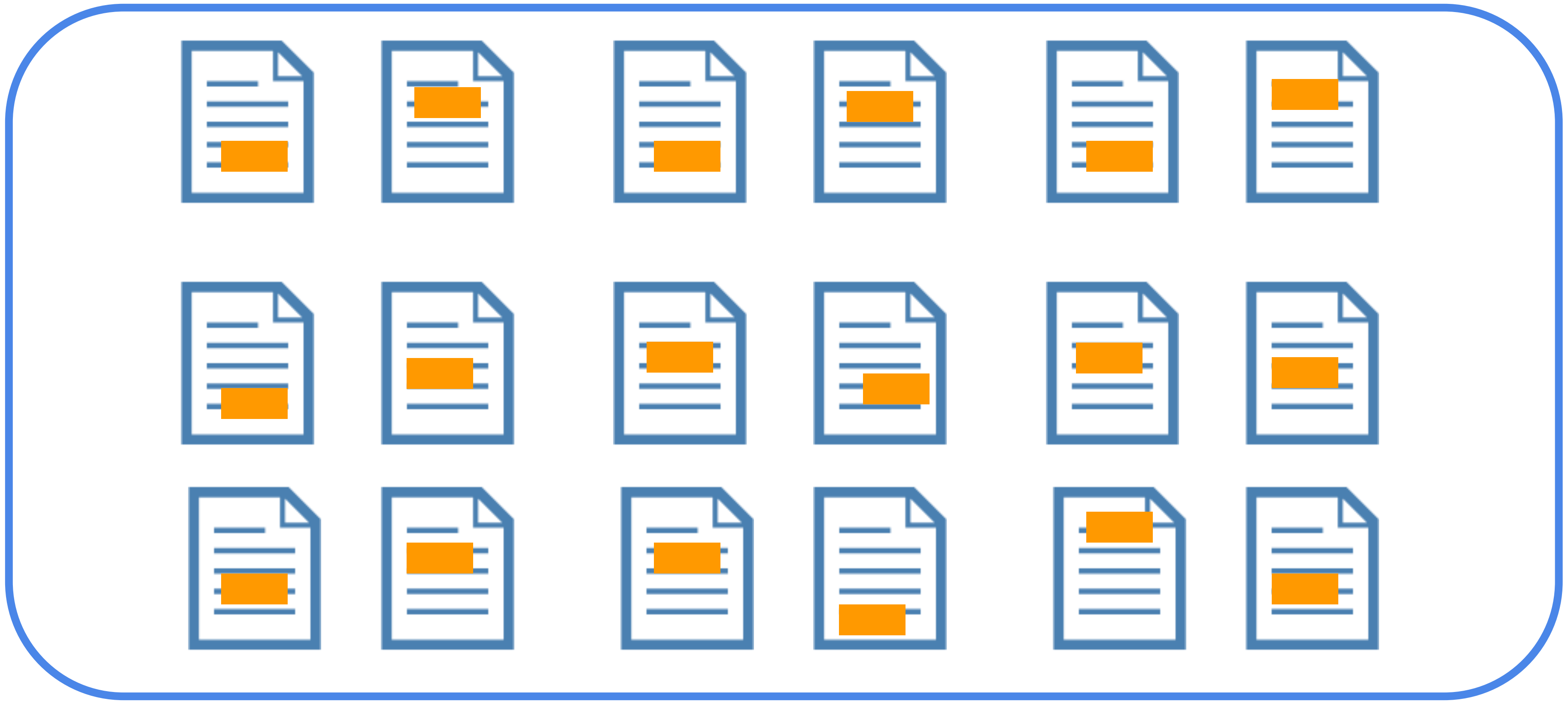


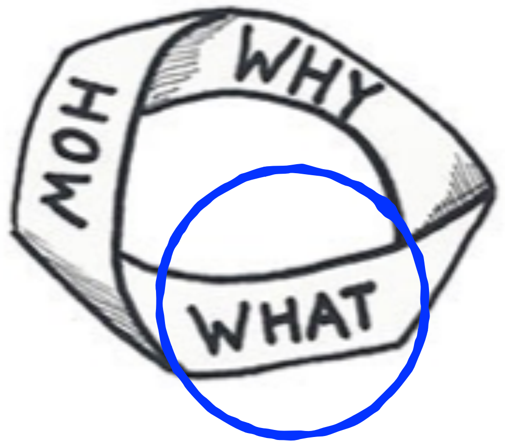
```
Id.ApplicationUnhandledException, "{0}\n{1}", ex.Message, ex
Id.NotifyEventStorageFailure, "{0}\n{1}", ex.Message, ex.Sta
= errors are generic hand have specific into in the inner ex
= null)

WarnId.ApplicationUnhandledException, "inner exception: {0}\
WarnId.NotifyEventStorageFailure, "inner exception: {0}\n{1}
```



ClusterChanges

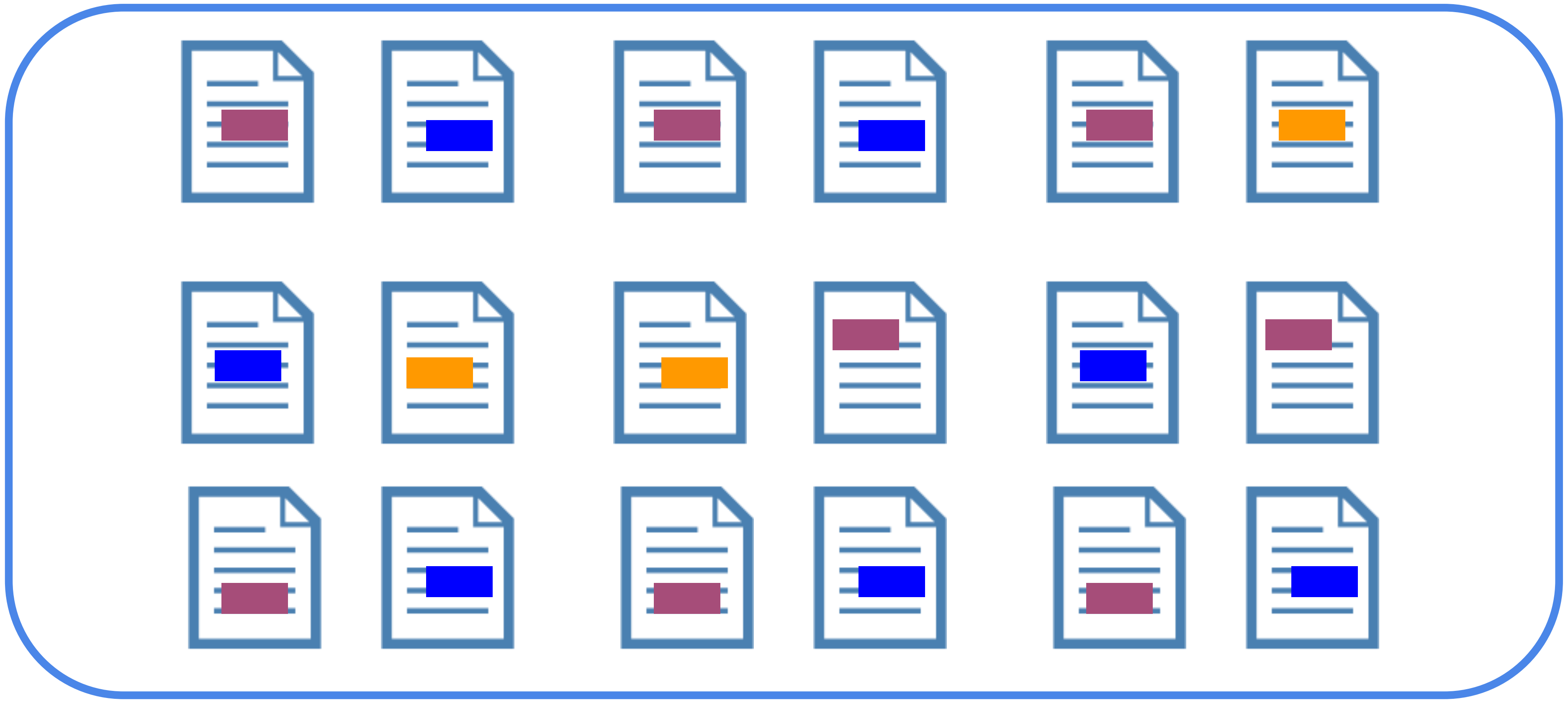
Standard set of diff regions

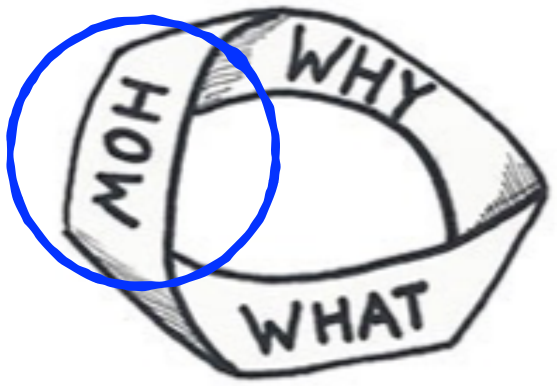




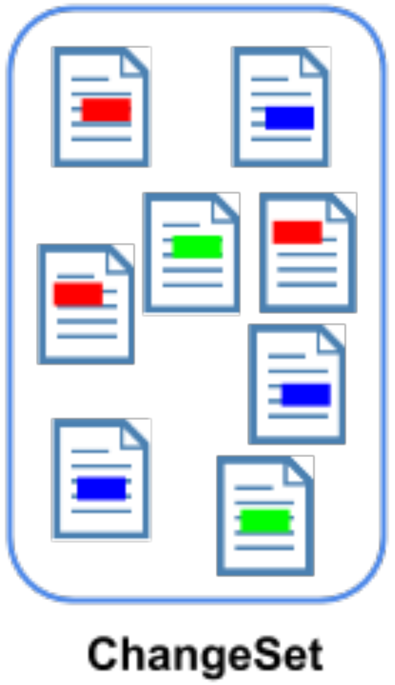
ClusterChanges

Some of them are syntactically and semantically related.

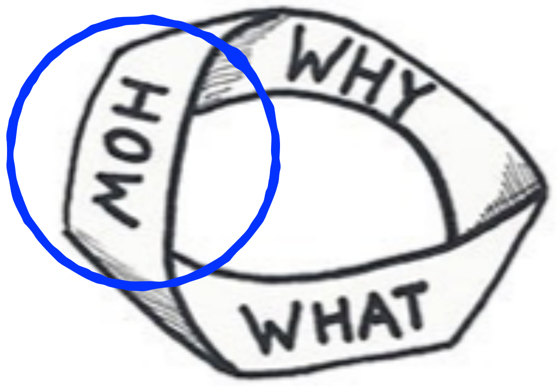




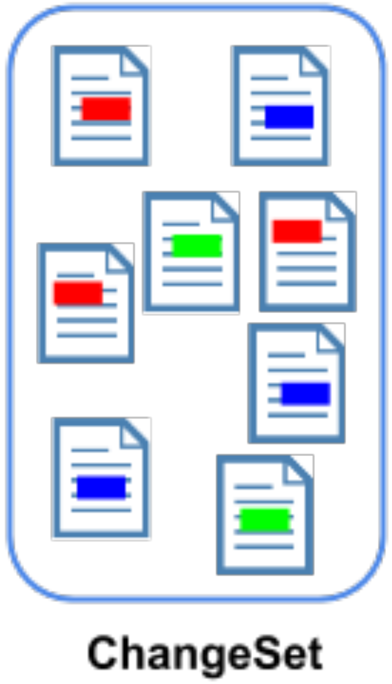
ClusterChanges



$$\xrightarrow[\text{Partial program analysis}]{\text{Roslyn}}$$



ClusterChanges



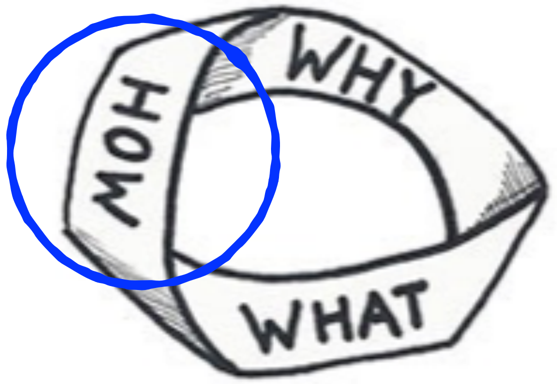
Roslyn

$\longrightarrow$

Partial program analysis

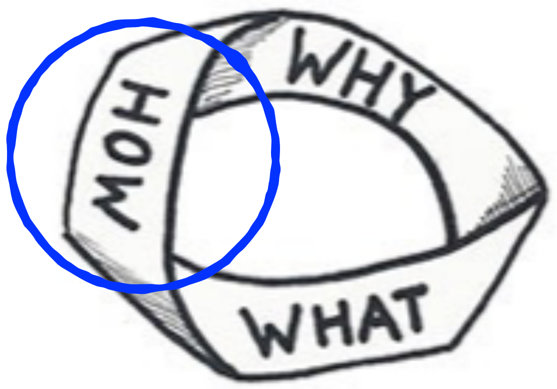
Entities (nodes)
methods, fields, properties etc

Relationships (edges)
method calls, field access etc



The relationships

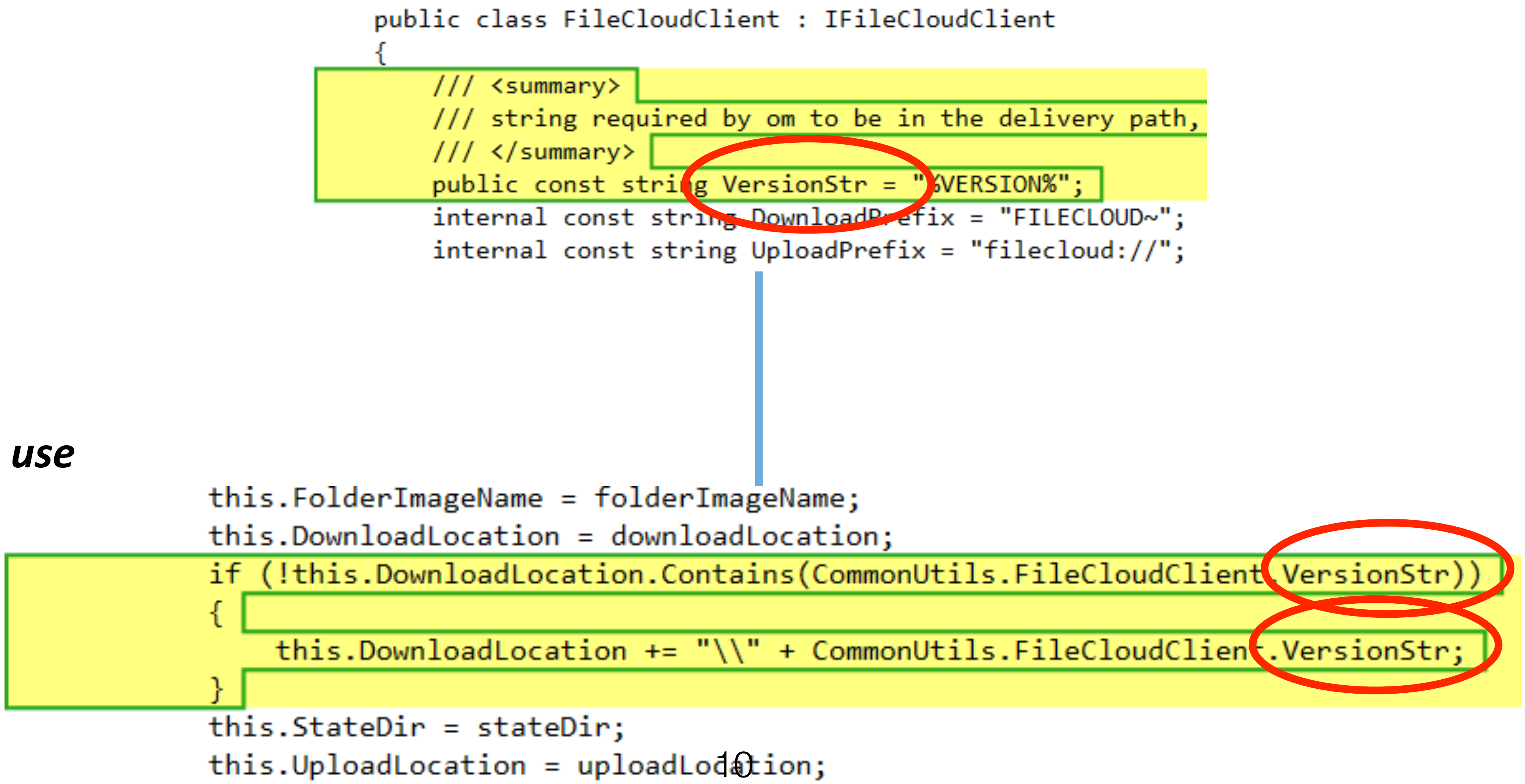
def-use relationship



The relationships

def-use relationship

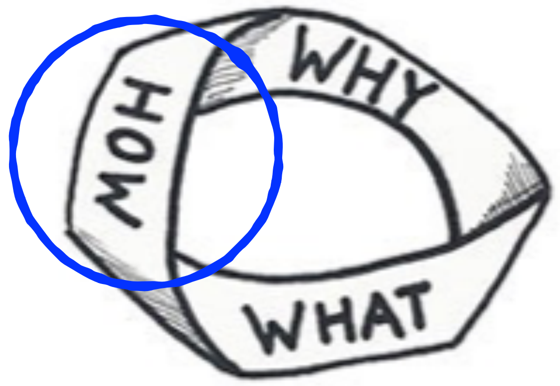
definition



```
public class FileCloudClient : IFileCloudClient
{
    /// <summary>
    /// string required by om to be in the delivery path,
    /// </summary>
    public const string VersionStr = "%VERSION%";
    internal const string DownloadPrefix = "FILECLOUD~";
    internal const string UploadPrefix = "filecloud://";
```

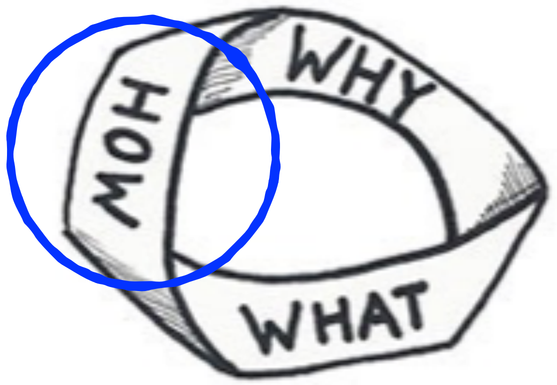
use

```
this.FolderImageName = folderImageName;
this.DownloadLocation = downloadLocation;
if (!this.DownloadLocation.Contains(CommonUtils.FileCloudClient.VersionStr))
{
    this.DownloadLocation += "\\" + CommonUtils.FileCloudClient.VersionStr;
}
this.StateDir = stateDir;
this.UploadLocation = uploadLocation;
```



The relationships

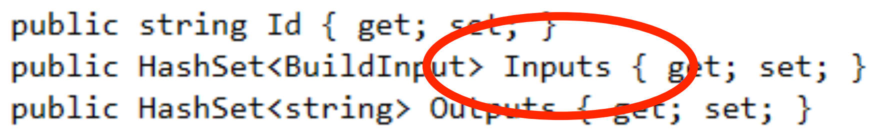
use-use relationship



The relationships

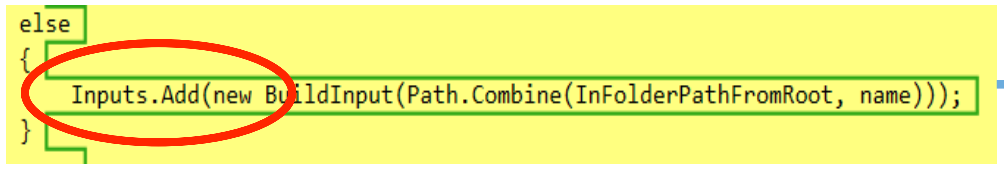
use-use relationship

definition



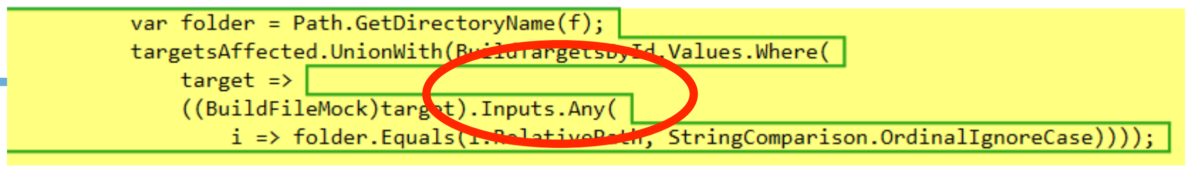
```
public string Id { get; set; }
public HashSet<BuildInput> Inputs { get; set; }
public HashSet<string> Outputs { get; set; }
```

use

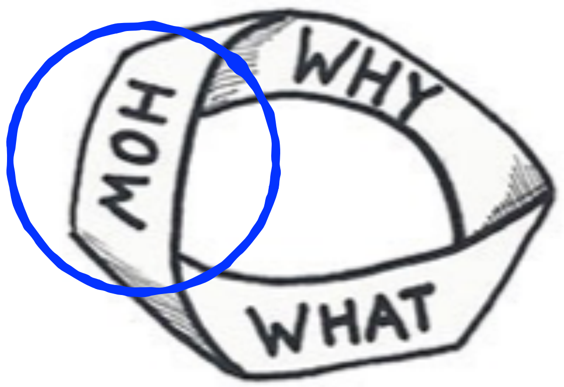


```
else
{
    Inputs.Add(new BuildInput(Path.Combine(InFolderPathFromRoot, name)));
}
```

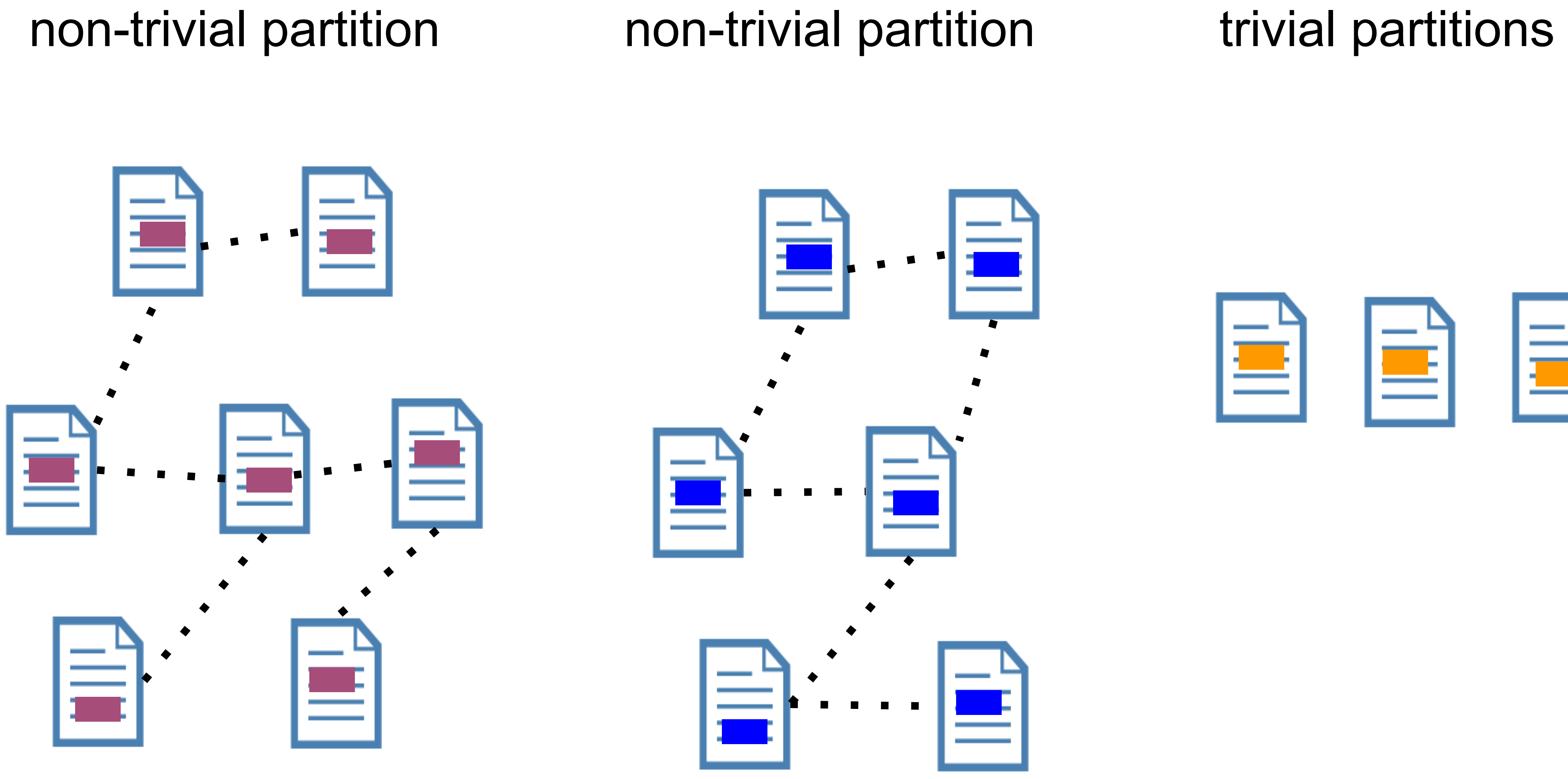
use

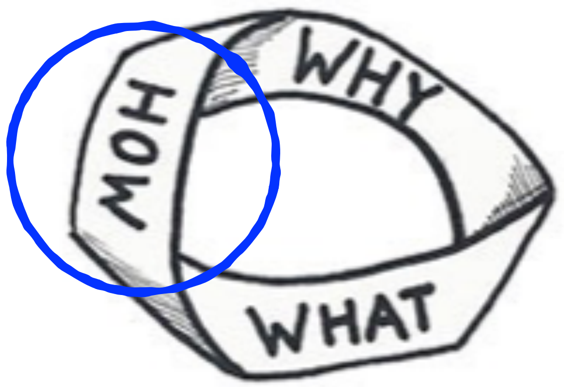


```
        var folder = Path.GetDirectoryName(f);
        targetsAffected.UnionWith(BuildTargetsById.Values.Where(
            target =>
            ((BuildFileMock)target).Inputs.Any(
                i => folder.Equals(i.RelativePath, StringComparison.OrdinalIgnoreCase))));
```

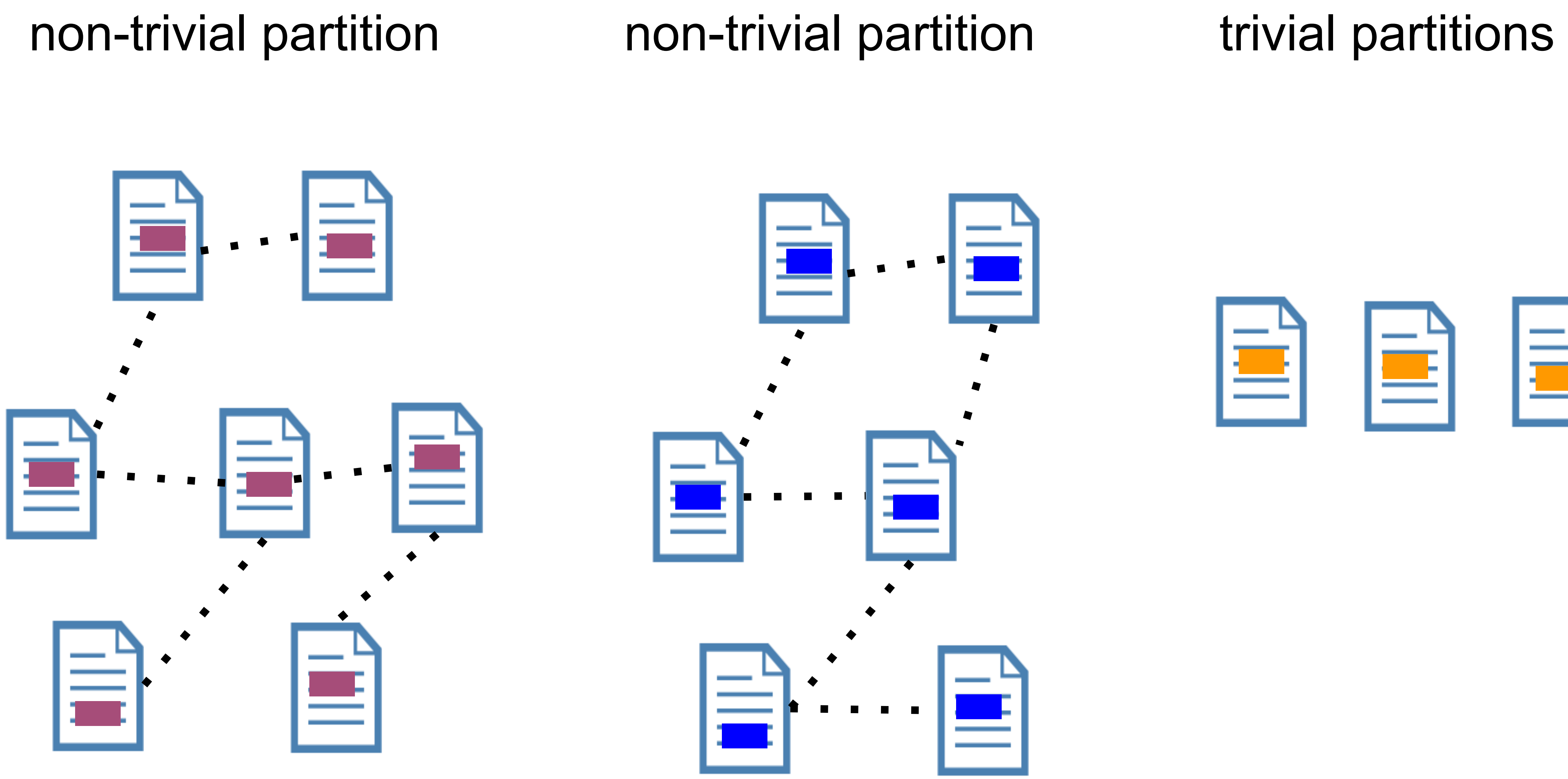


ClusterChanges





ClusterChanges



Diffs within the same method are linked together.
We don't split method across partitions.

2-step Evaluation

Quantitative

Distribution of partitions

1000 Bing and Office reviews

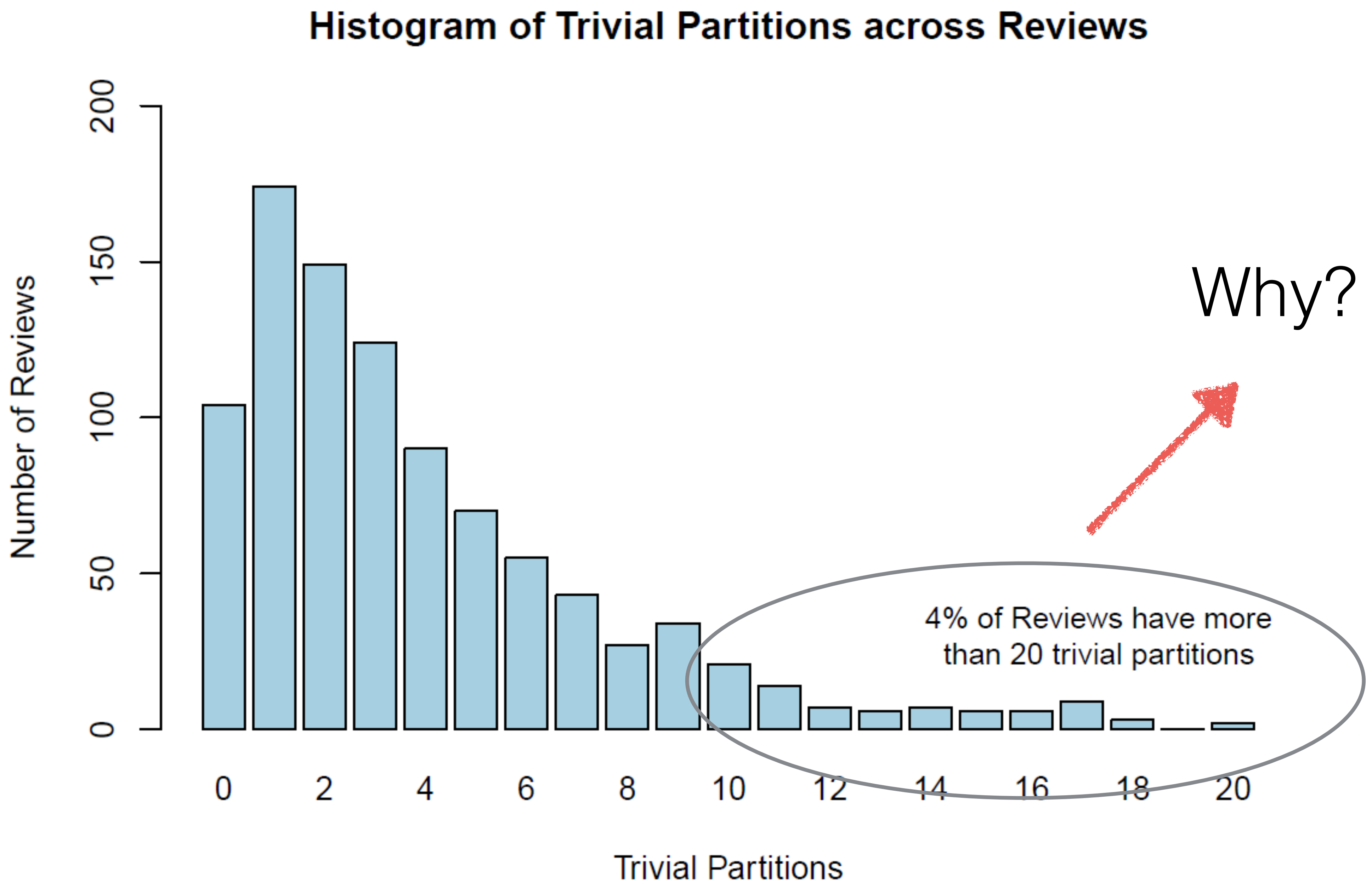
Manual inspection

100 reviews

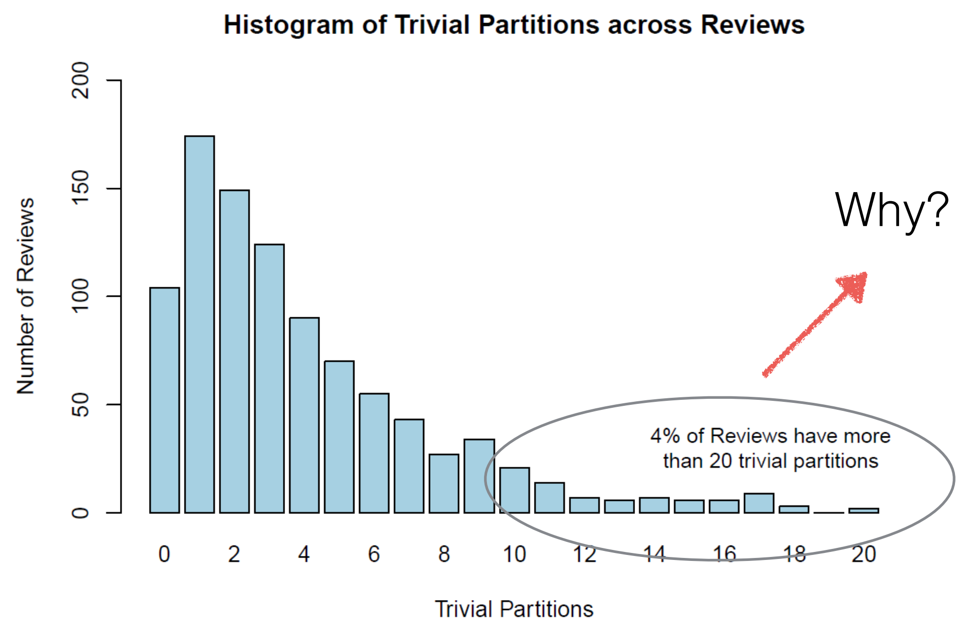
Qualitative

Interviews: 20 developers from 13 projects

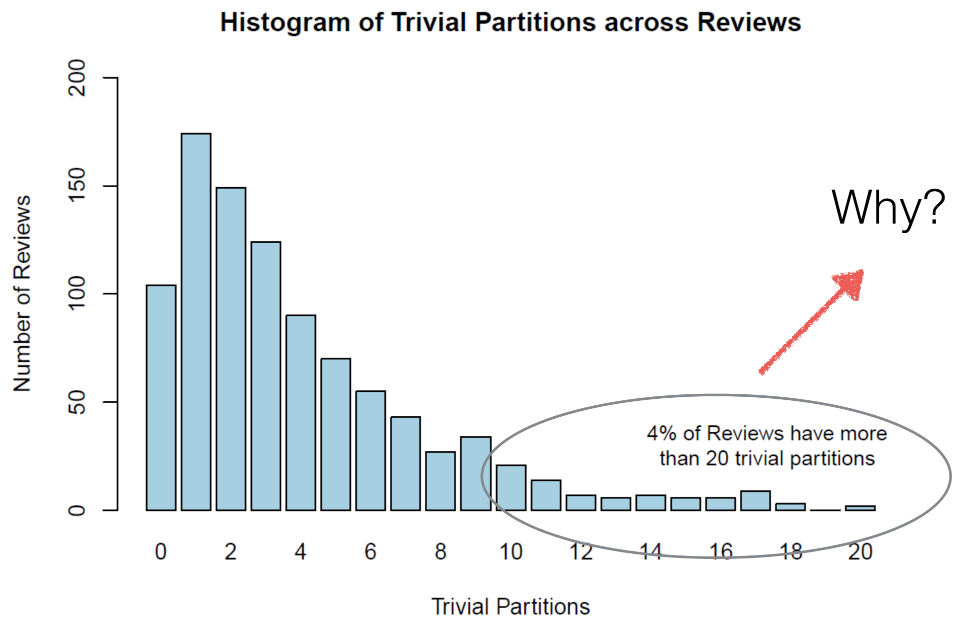
Quantitative: trivial partitions



Quantitative: trivial partitions

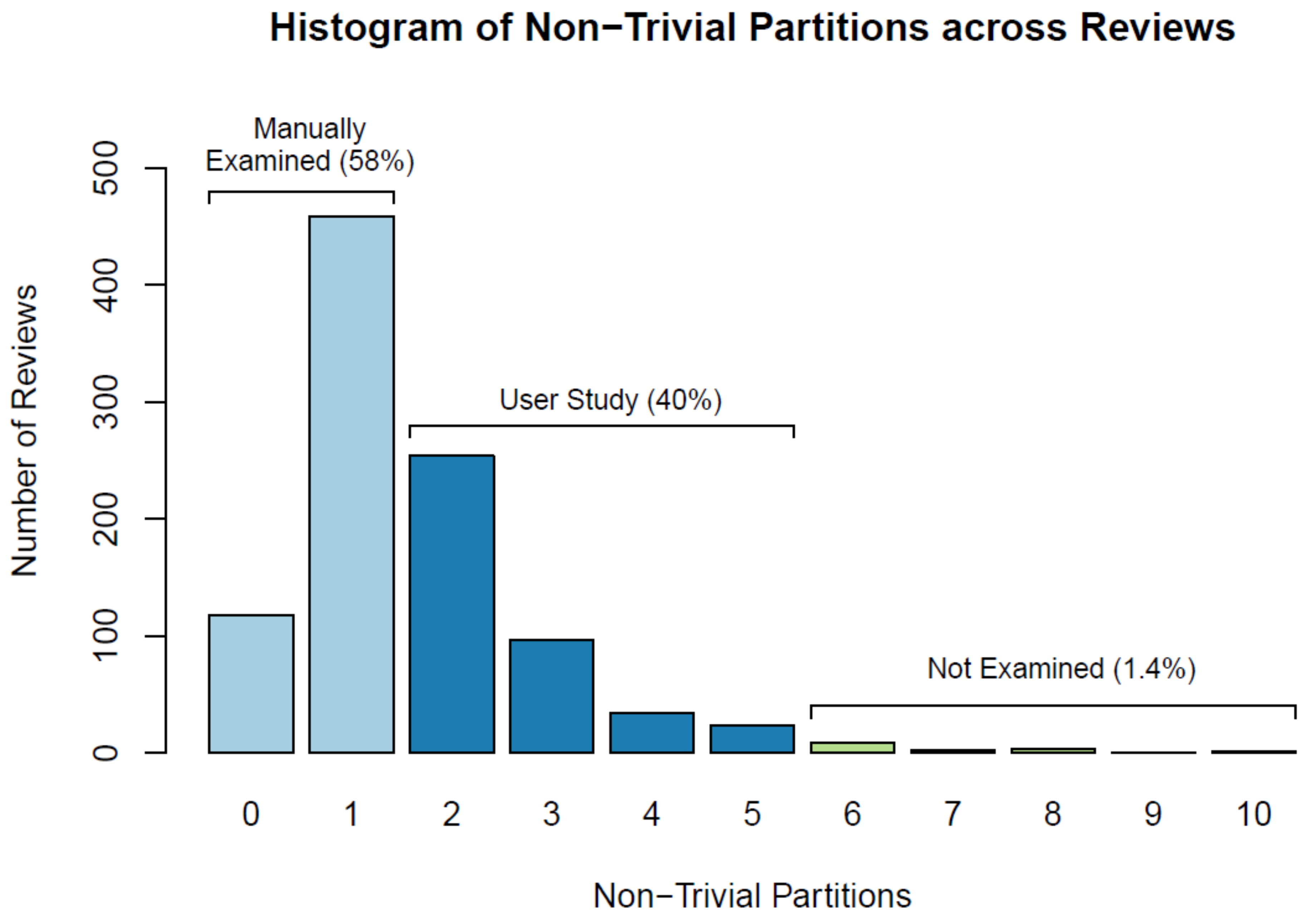


Quantitative: trivial partitions

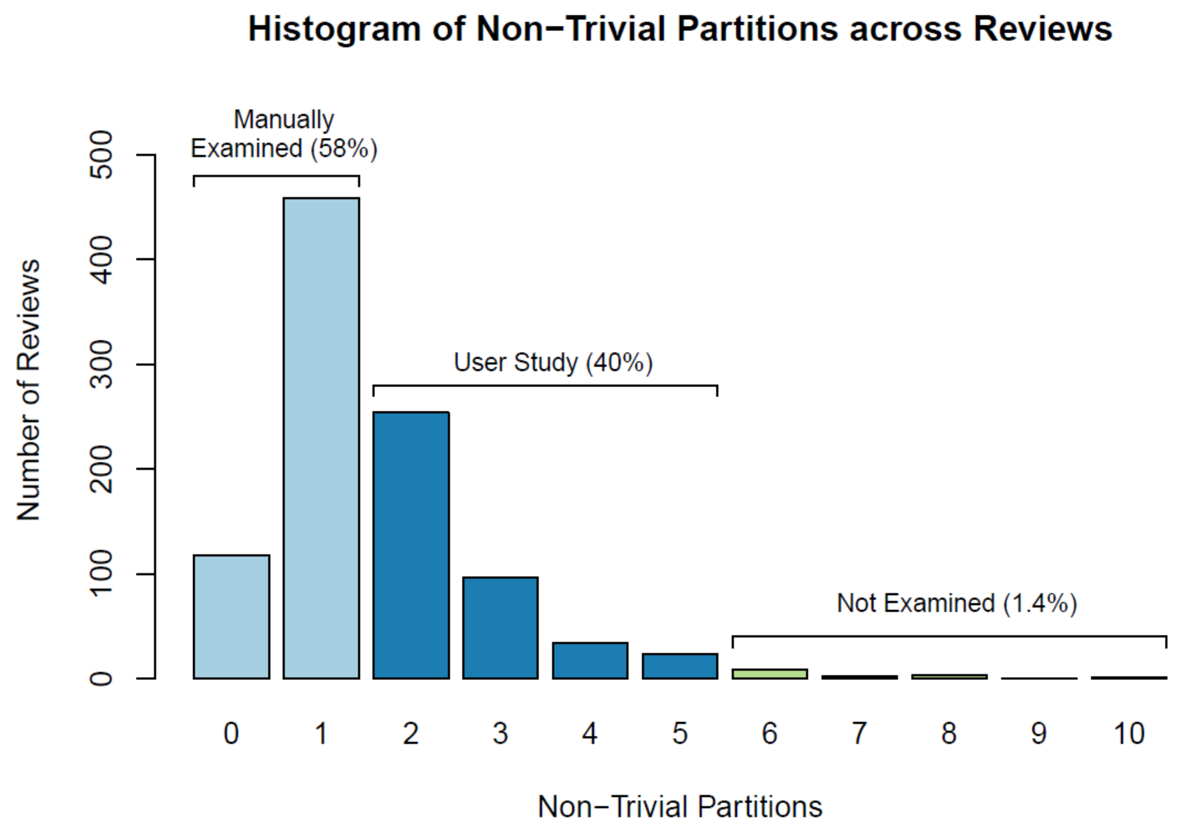


e.g. Log Messages

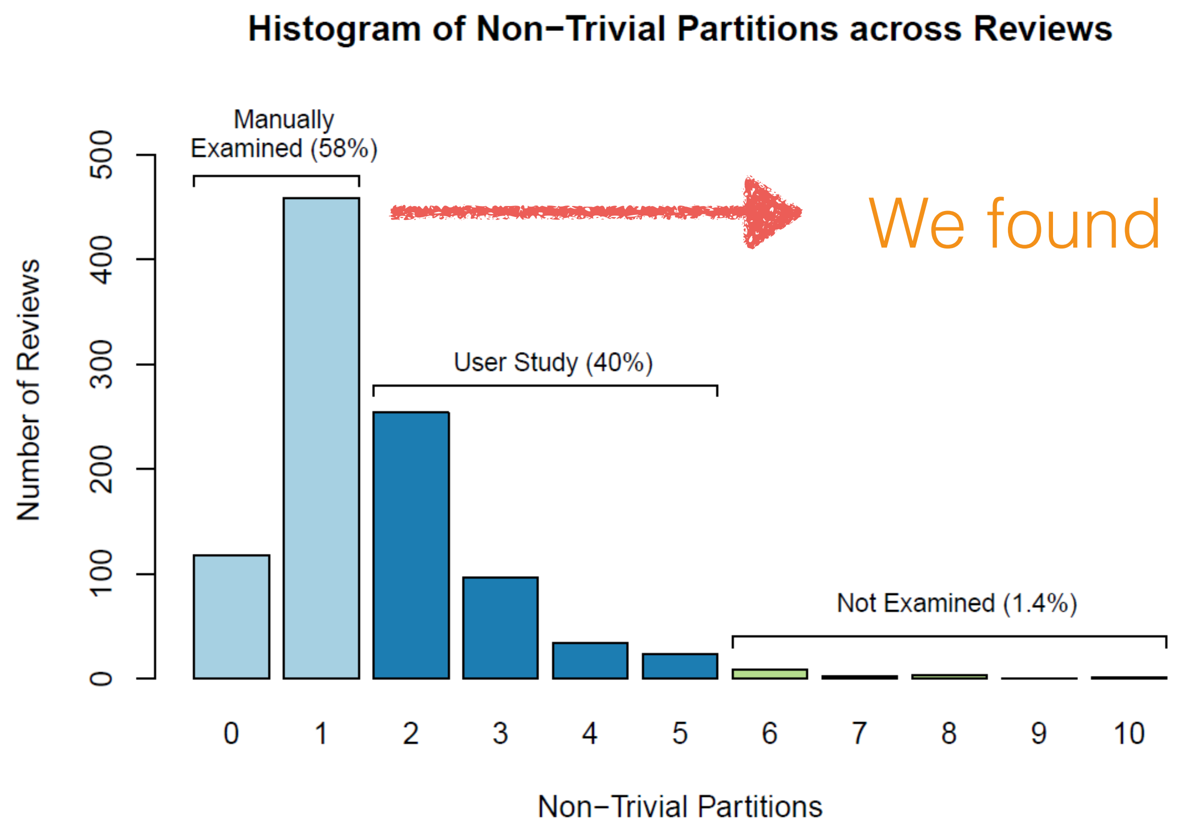
Quantitative: non-trivial partitions



Quantitative: non-trivial partitions

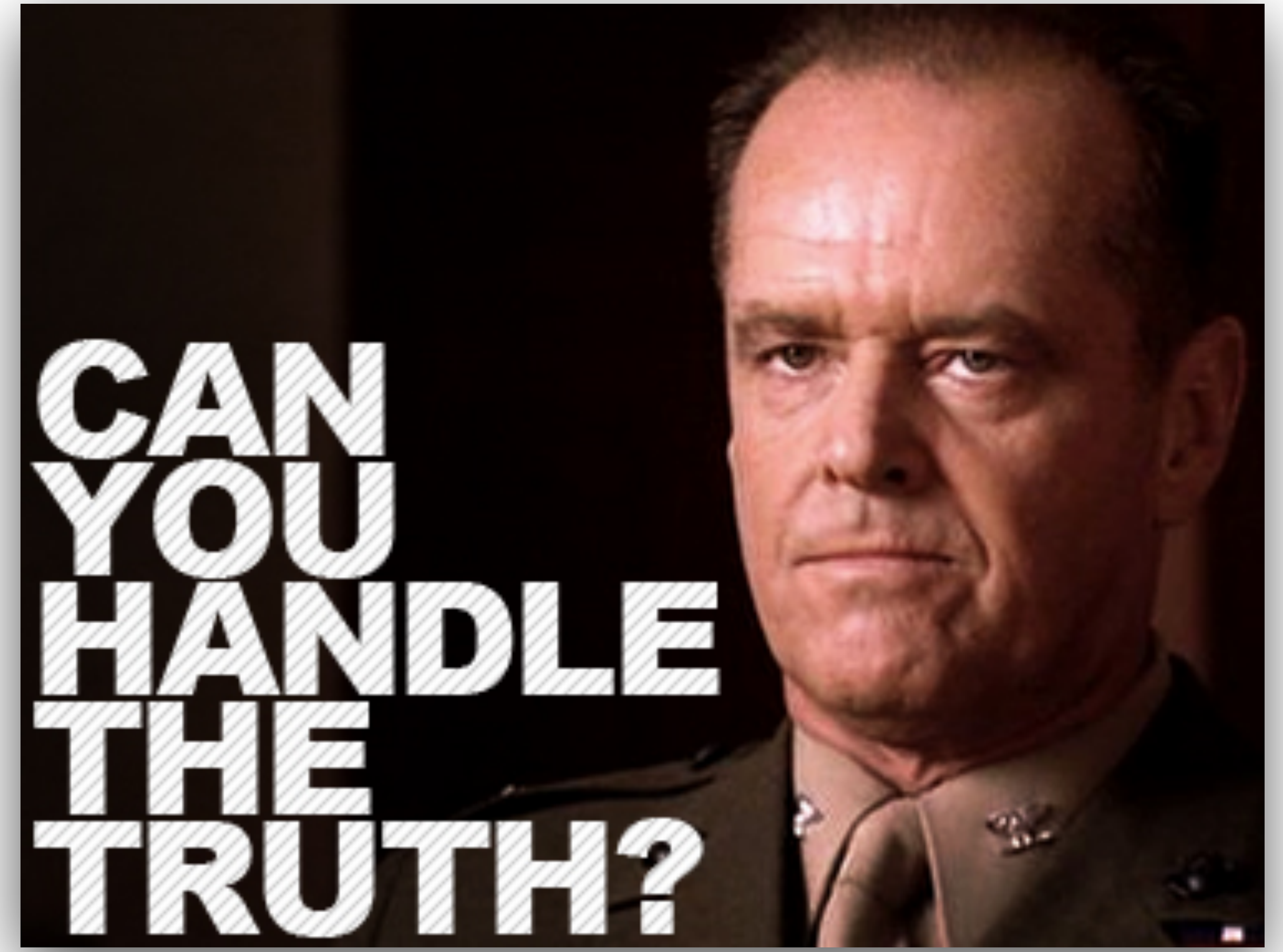


Quantitative: non-trivial partitions

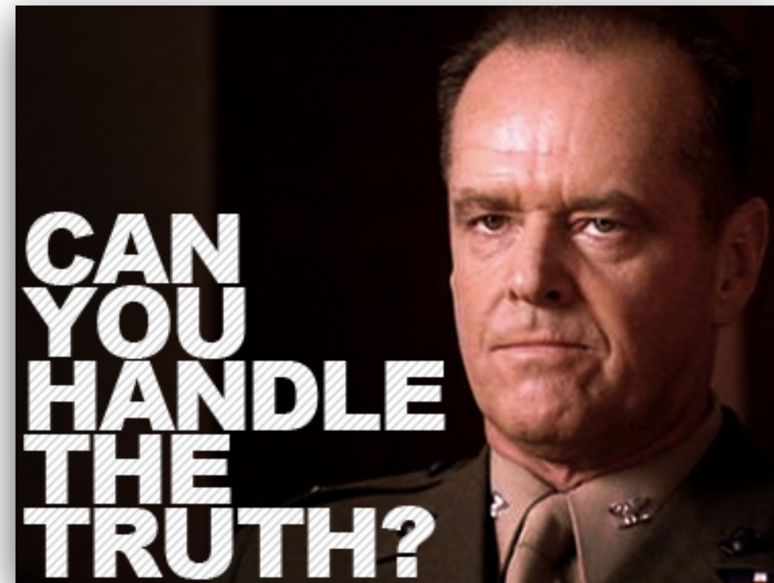


We found 3 false-negatives from 50 Changesets

Qualitative: The ground truth

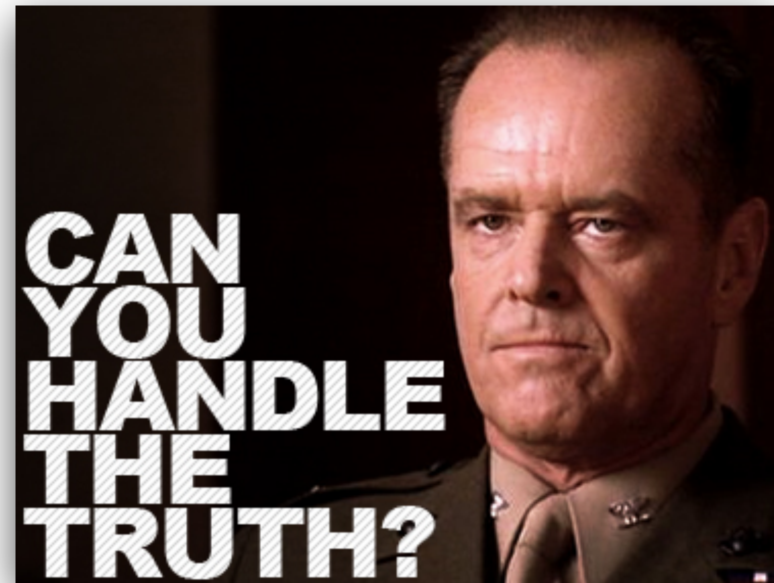


Qualitative: The ground truth



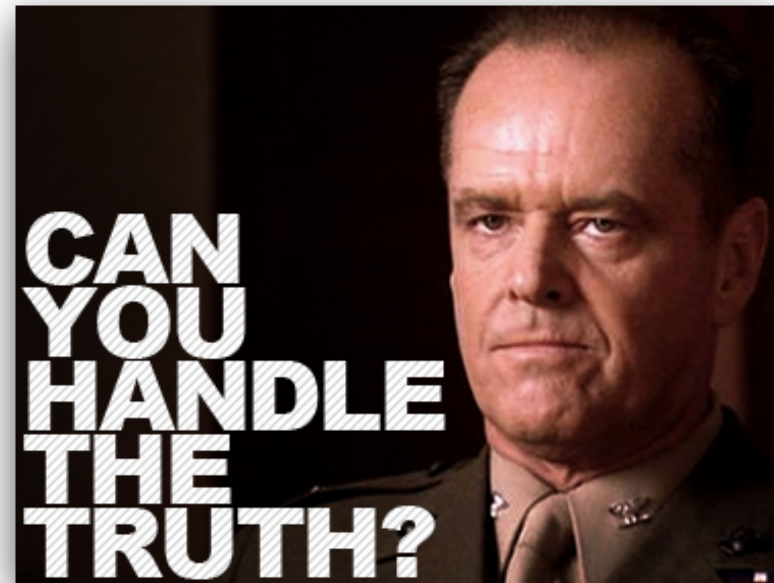
Qualitative: The ground truth

RQ1: Do developers agree with our decomposition?



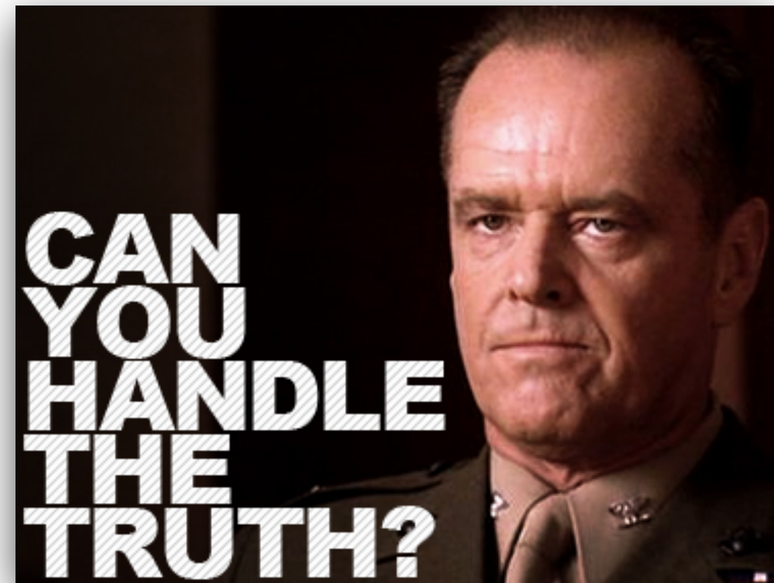
Qualitative: The ground truth

RQ1: Do developers agree with our decomposition?



Qualitative: The ground truth

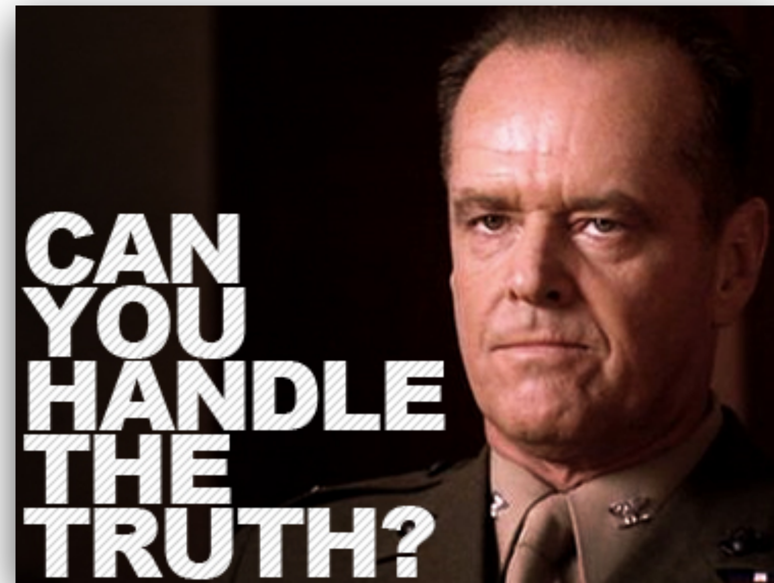
RQ1: Do developers agree with our decomposition?



Qualitative: The ground truth

RQ1: Do developers agree with our decomposition?

RQ2: Can organizing a changeset using our decomposition help reviewers?



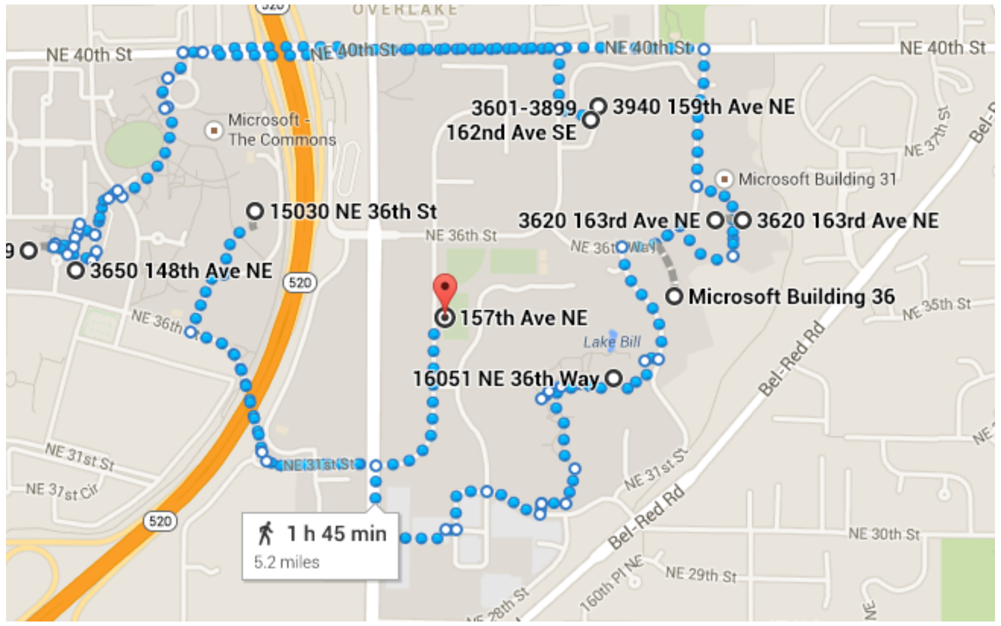
Firehouse interviews

- Monitored review submissions
- Criteria: distance and #partitions

Firehouse interviews

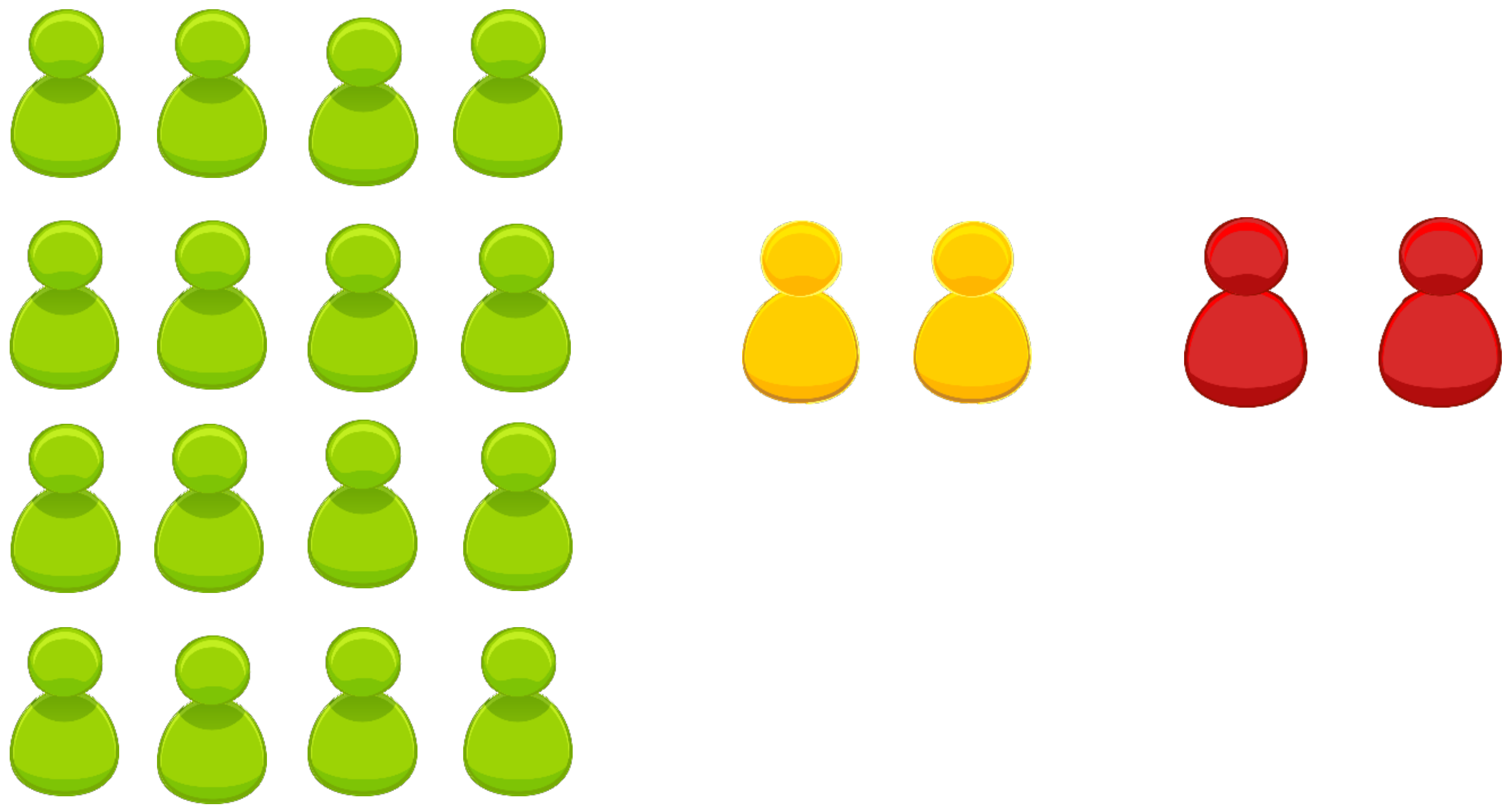
- Monitored review submissions
- Criteria: distance and #partitions

Rush to the scene!

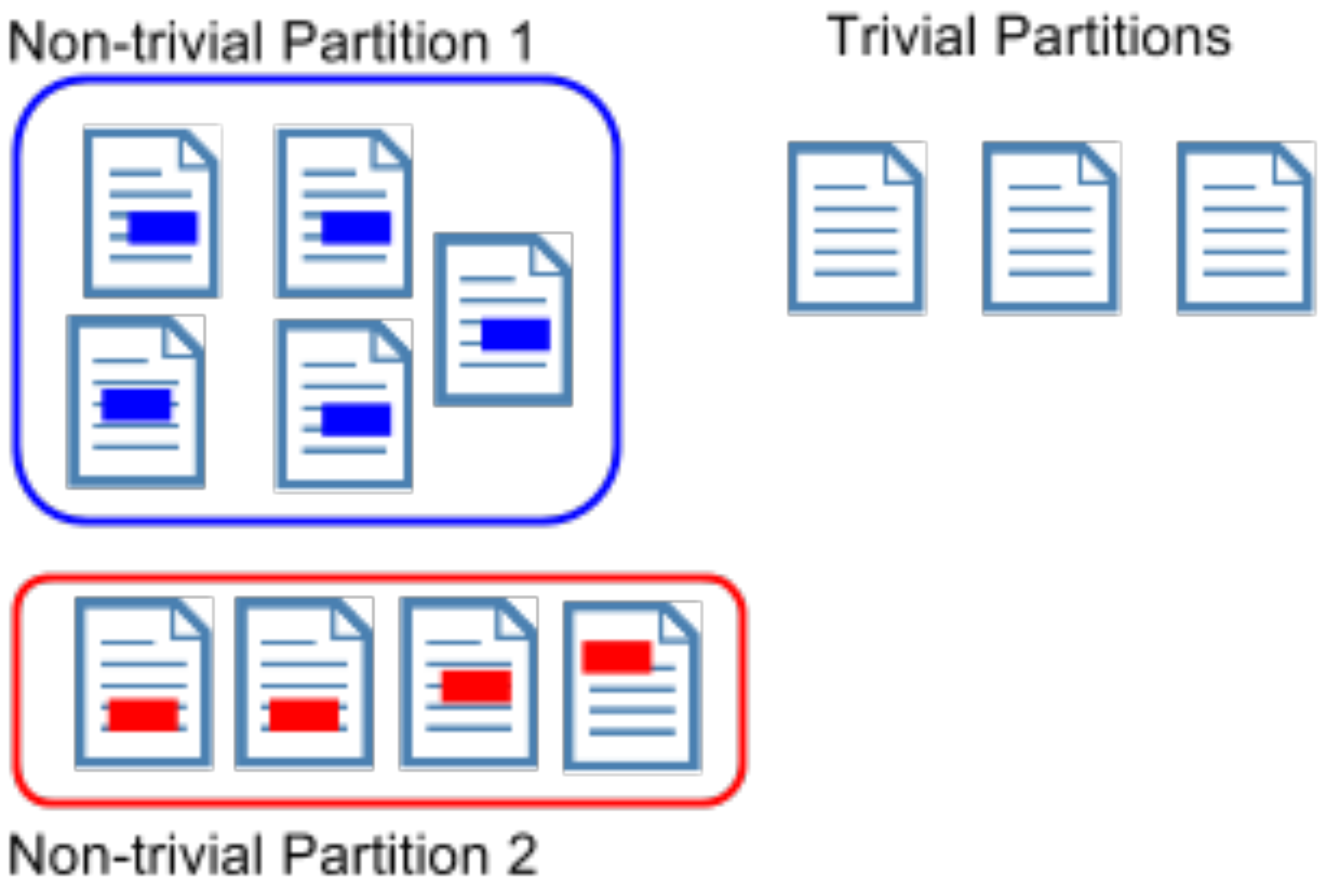


50mi -> 80.5 km

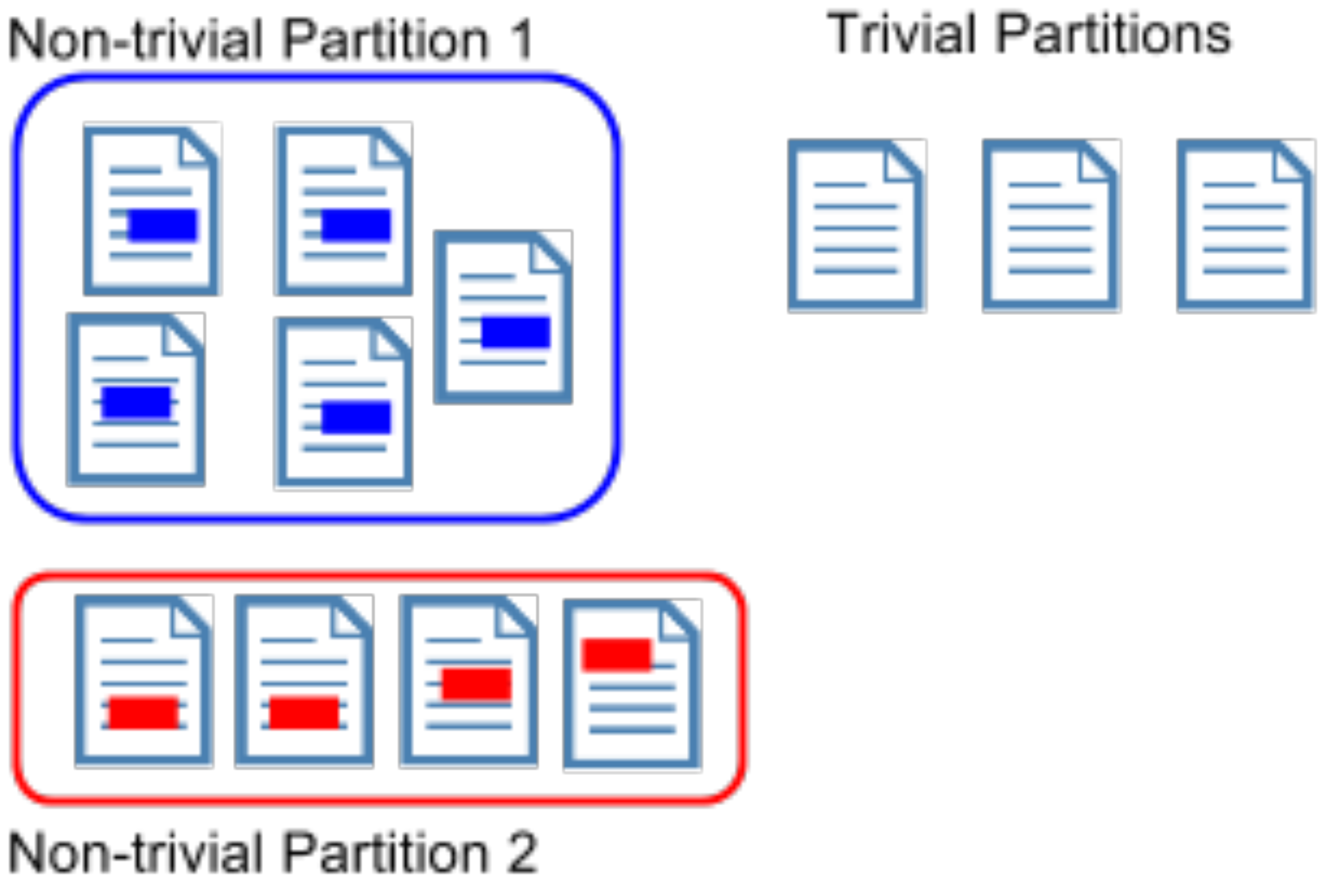
RQ1: Do developers agree with our decomposition?



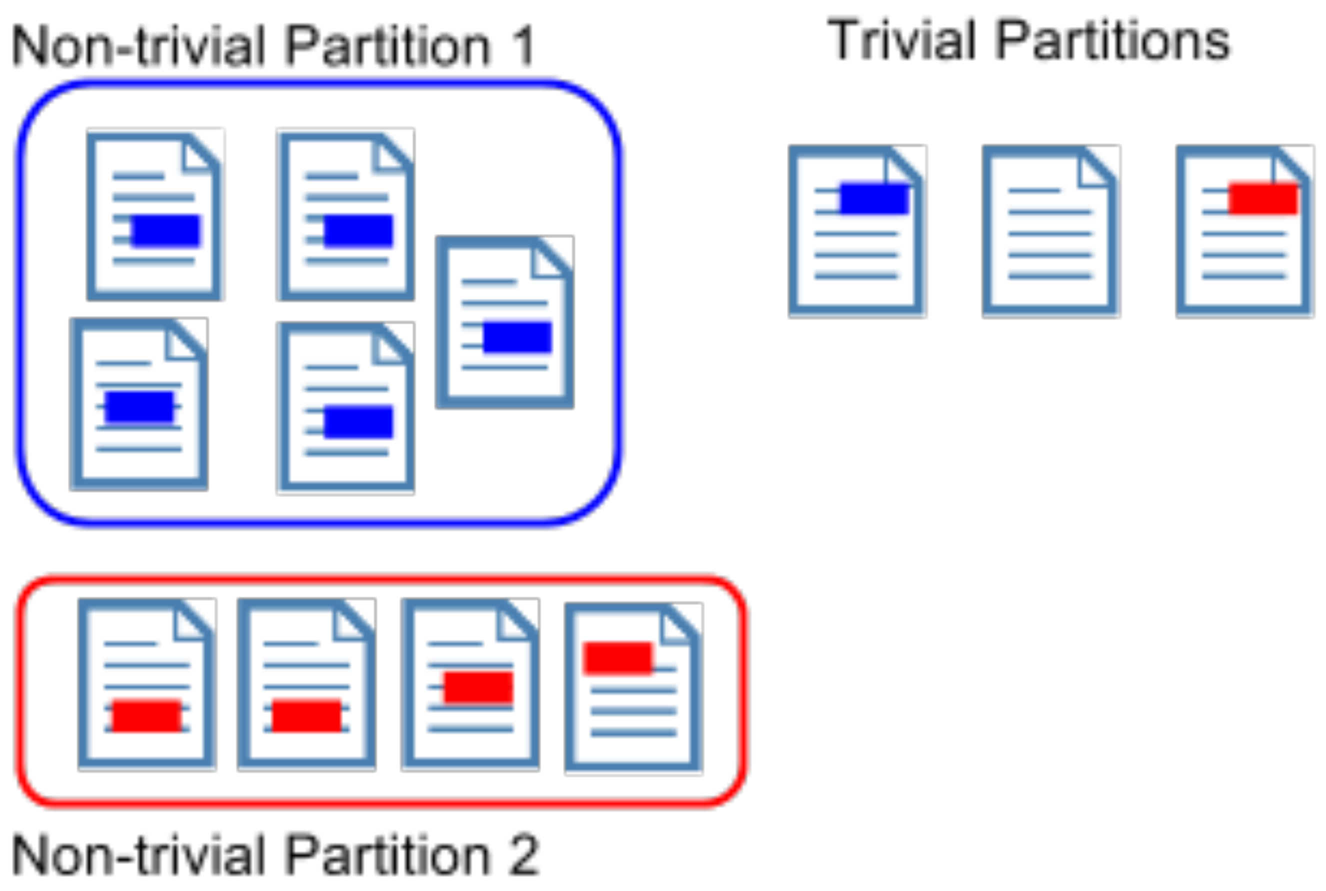
RQ1: Do developers agree with our decomposition?



RQ1: Do developers agree with our decomposition?



RQ1: Do developers agree with our decomposition?



Some trivial should be moved to one of non-trivial partitions.

RQ1: Do developers agree with our decomposition?

RQ1: Do developers agree with our decomposition?

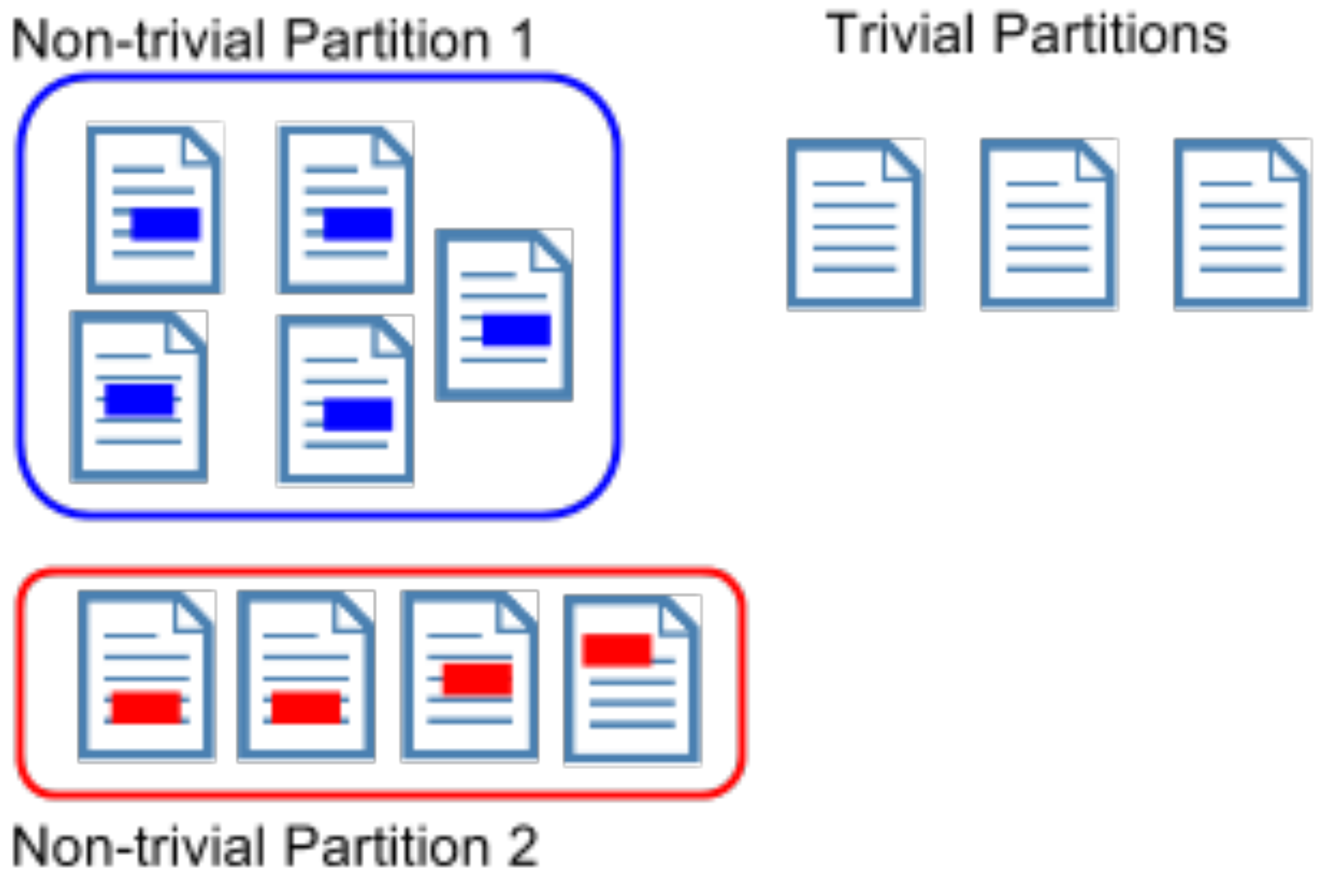
“These were actually two different changes and I actually split them in two different things after this review” [P7].

RQ1: Do developers agree with our decomposition?

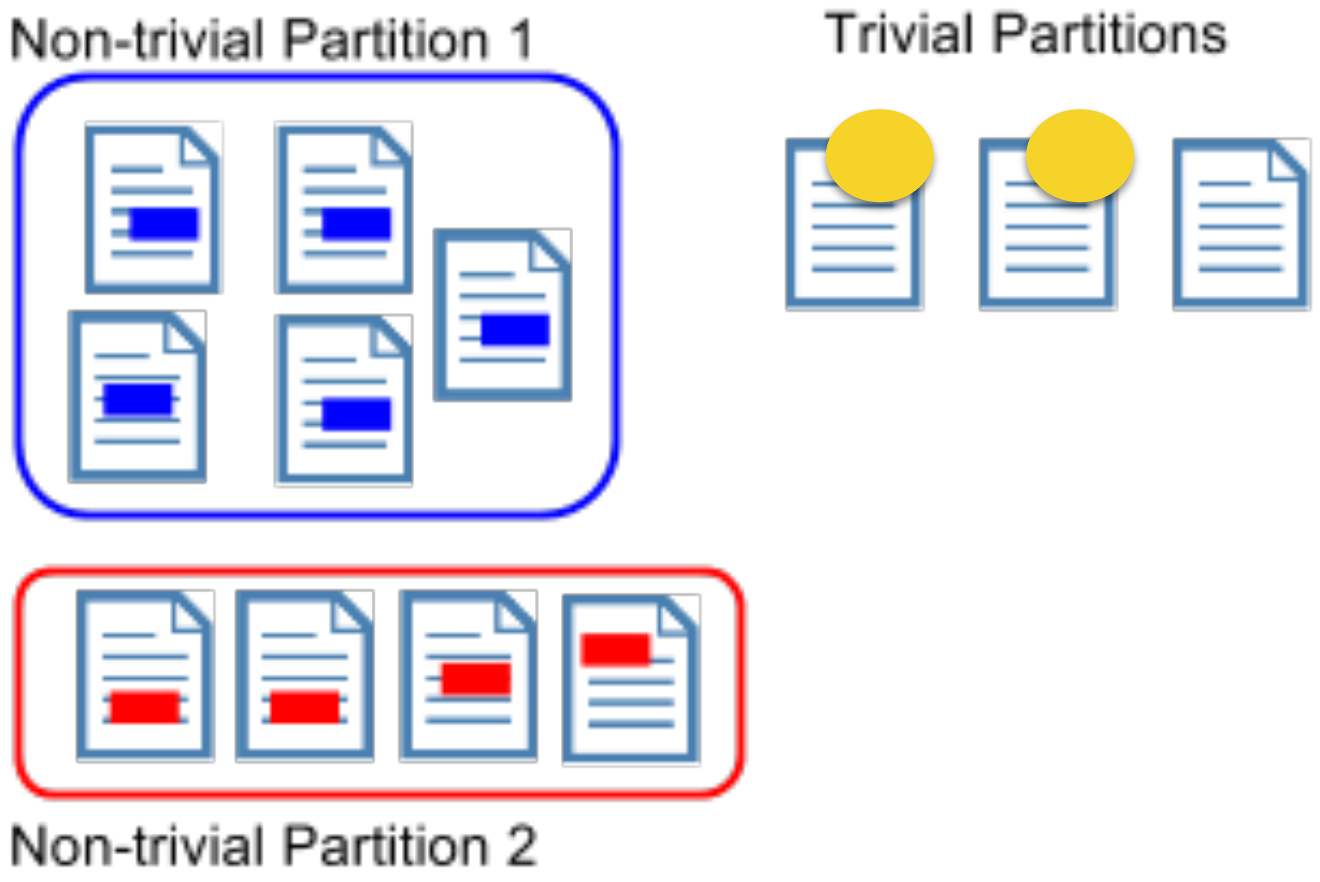
“These were actually two different changes and I actually split them in two different things after this review” [P7].

“I would like tag partitions” [P14].

RQ1: Do developers agree with our decomposition?



RQ1: Do developers agree with our decomposition?



"**In some sort of hypothetical perfect splitting** that read my mind, there is one change in one line which was a variable changed (regular expression) that could be in a different partition. **But I would not expect that, because it is difficult**" [P9].

RQ1: Do developers agree with our decomposition?

RQ1: Do developers agree with our decomposition?

[P13] “There is no reason to commit unrelated changes.”

“The tool should be 95% correct or else I would not use it because it would be annoying.”

RQ1: Do developers agree with our decomposition?

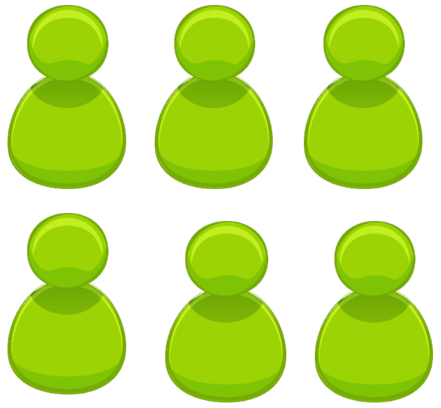
[P13] "There is no reason to commit unrelated changes."

"The tool should be 95% correct or else I would not use it because it would be annoying."

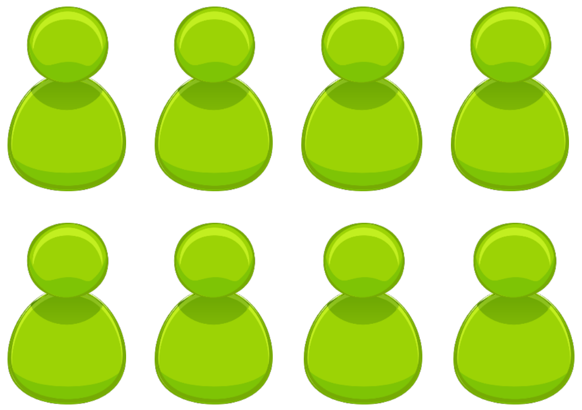
[P17] "What is the why behind it?"

"If you do not have something showing why/how the partitions were created, it is difficult to see its value."

RQ2: Can organizing a changeset using our decomposition help reviewers?



“For large sets it it would be very helpful"



“it is useful because allow different reviewers with different purposes to focus on what they want.”

RQ2: Can organizing a changeset using our decomposition help reviewers?

It would speed up the review process.

pre-commit usage

"If I had a way to run this tool before I commit, I would have even considered splitting this partition 2 into a second commit" [P4].

And, more importantly…

Developers mentioned CodeFlow and ask to use or integrate.

Microsoft internal event with developers [Techfest]

- hundreds signed up to be notified
- quite a few asked for the prototype
- a few have joined the project and are now contributing

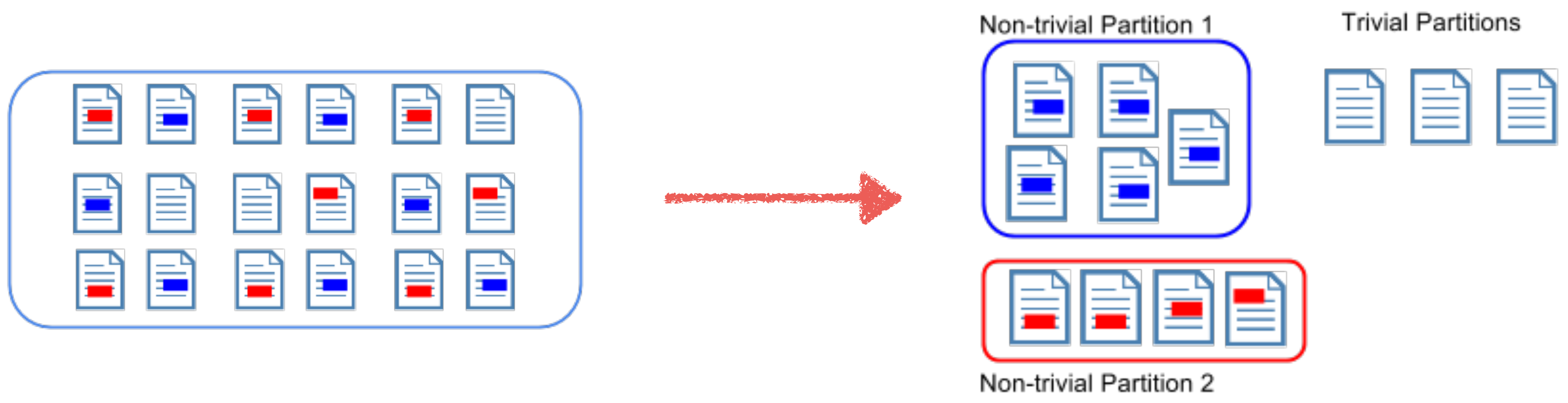
Related Work

Tao et al. *How do software engineers understand code changes? An exploratory study in industry*. In FSE, 2012.

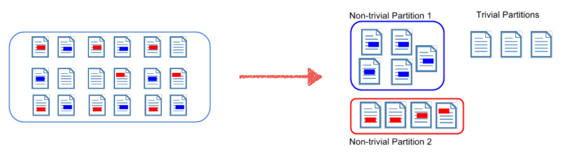
Kirinuki et al. *Hey! are you committing tangled changes?* In ICPC, 2014.

Herzig and Zeller: *The impact of tangled code changes*. In MSR, 2013.

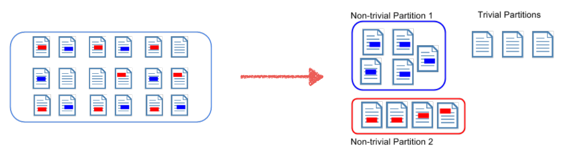
Summary

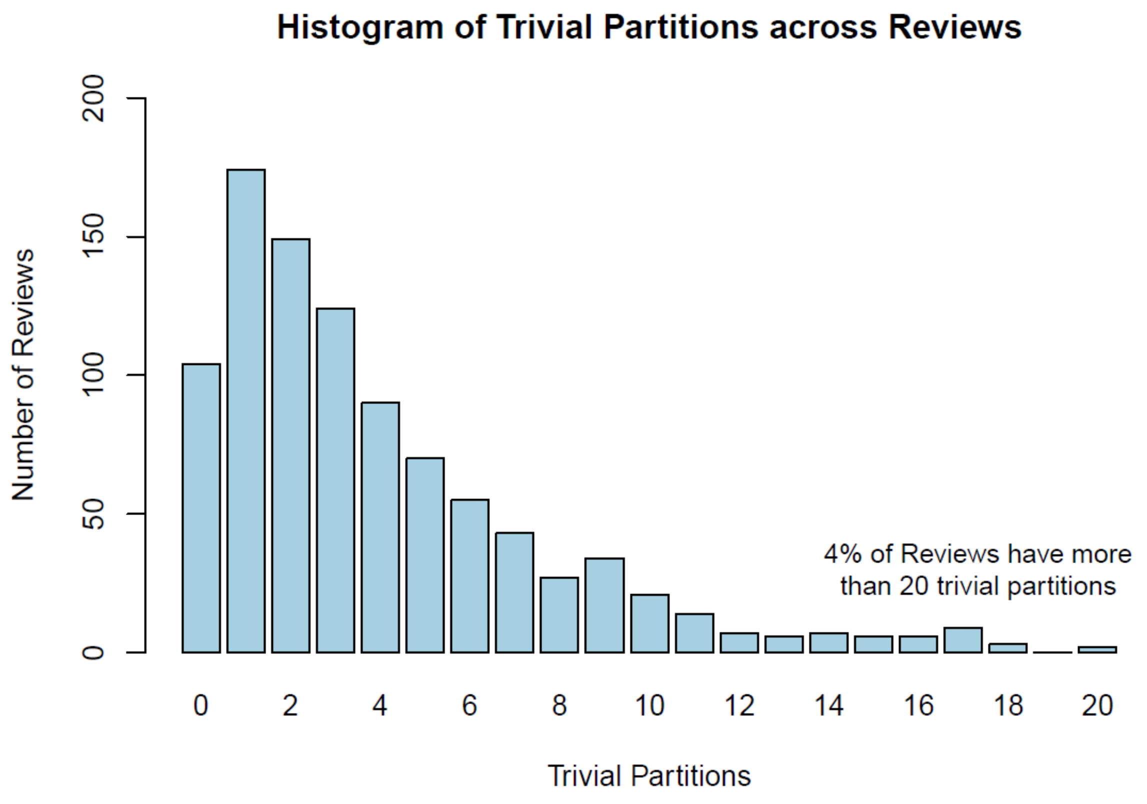

Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2

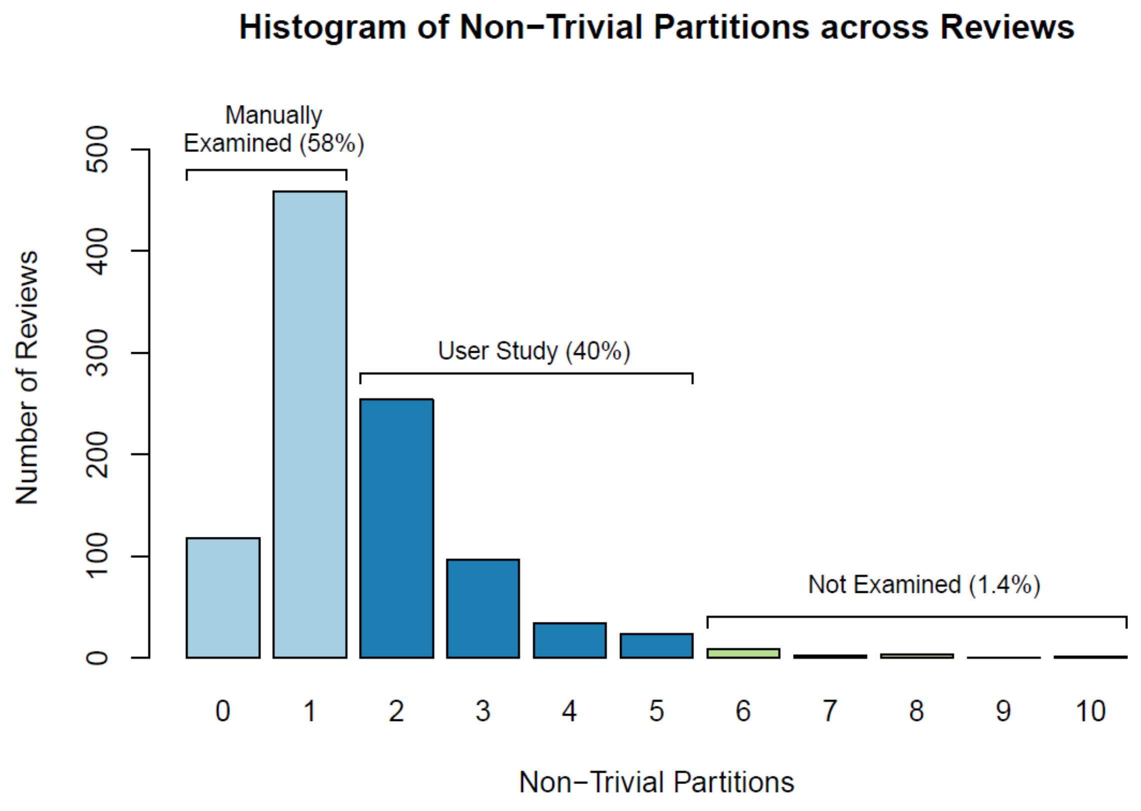
Summary


Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2

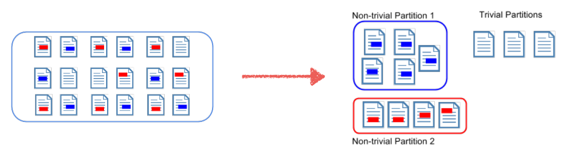
Summary


Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2

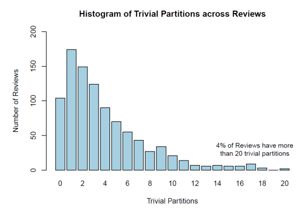

Histogram of Trivial Partitions across Reviews
Number of Reviews
0
50
100
150
200
0
2
4
6
8
10
12
14
16
18
20
Trivial Partitions
4% of Reviews have more than 20 trivial partitions


Histogram of Non−Trivial Partitions across Reviews
Number of Reviews
0
100
200
300
400
500
Manually Examined (58%)
User Study (40%)
Not Examined (1.4%)
0
1
2
3
4
5
6
7
8
9
10
Non−Trivial Partitions

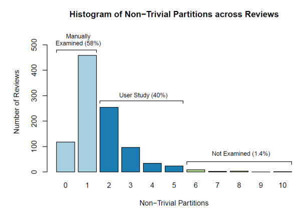
Summary



Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2

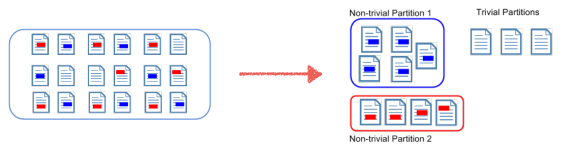


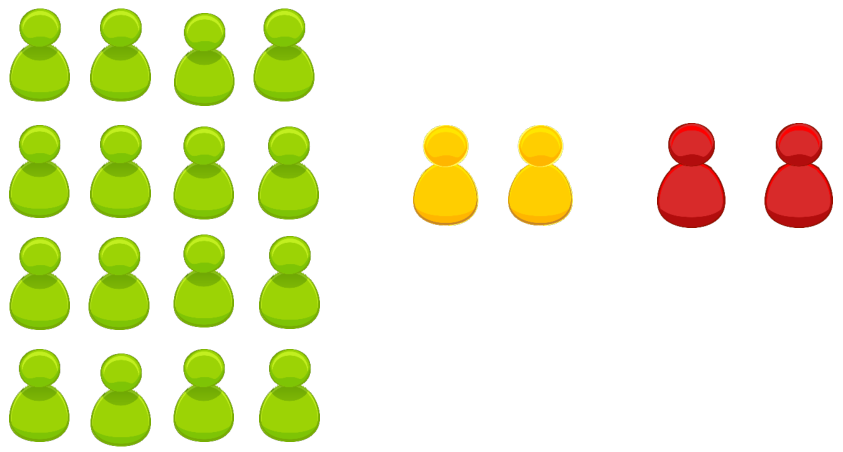
Histogram of Trivial Partitions across Reviews
Number of Reviews
Trivial Partitions
4% of Reviews have more than 20 trivial partitions



Histogram of Non−Trivial Partitions across Reviews
Manually Examined (58%)
User Study (40%)
Not Examined (1.4%)
Number of Reviews
Non−Trivial Partitions

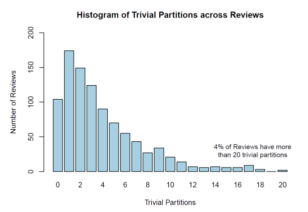
Summary

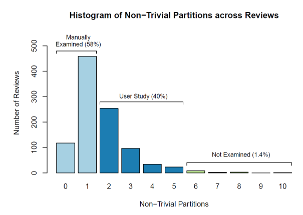

Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2



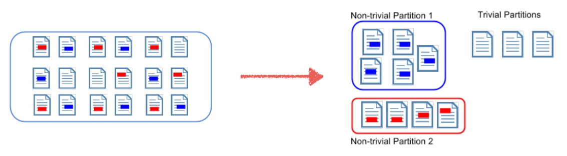
False
Positives

FALSE
TRUE
FALSE

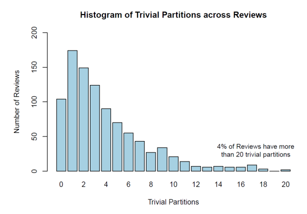

Histogram of Trivial Partitions across Reviews
Number of Reviews
Trivial Partitions
4% of Reviews have more than 20 trivial partitions
0
50
100
150
200
0
2
4
6
8
10
12
14
16
18
20


Histogram of Non−Trivial Partitions across Reviews
Manually Examined (58%)
User Study (40%)
Not Examined (1.4%)
Number of Reviews
Non−Trivial Partitions
0
100
200
300
400
500
0
1
2
3
4
5
6
7
8
9
10

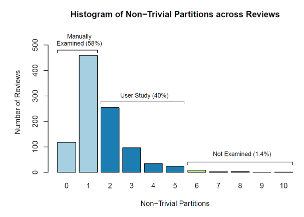
Summary



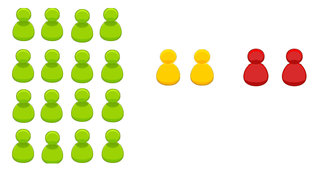
Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2



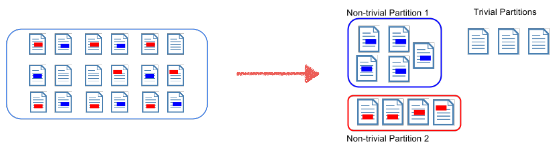
Histogram of Trivial Partitions across Reviews
Number of Reviews
4% of Reviews have more than 20 trivial partitions
Trivial Partitions



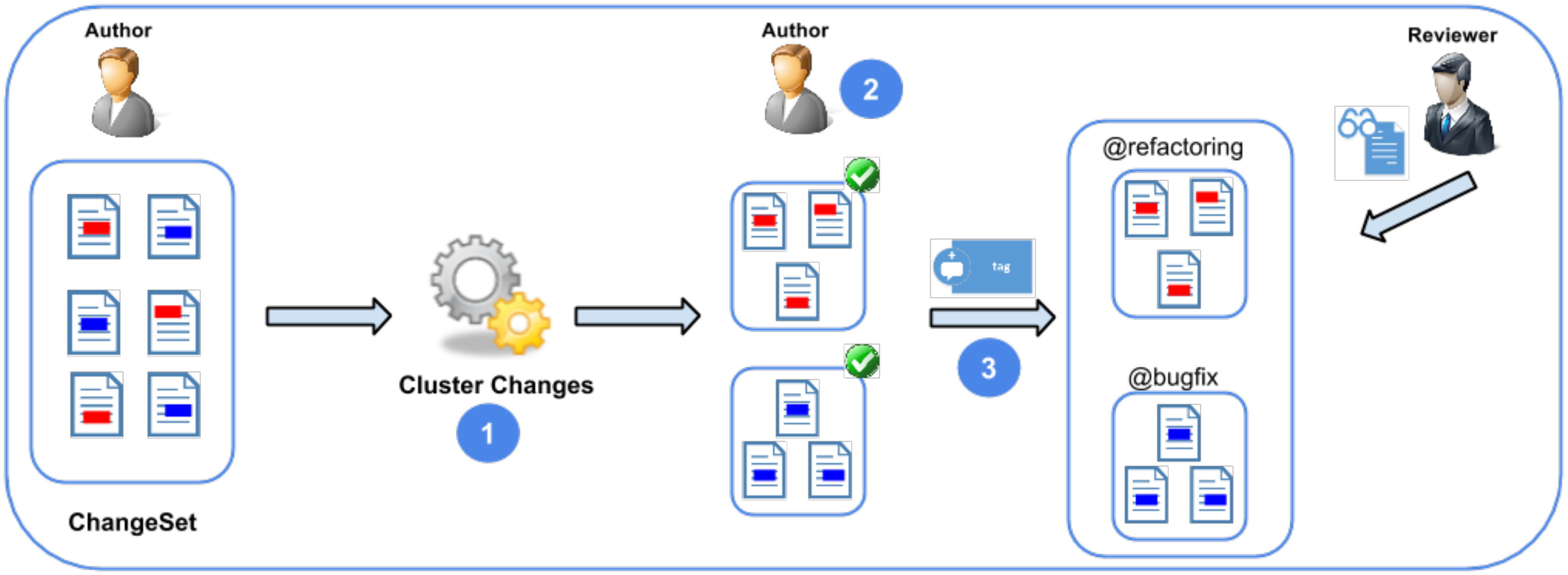
Histogram of Non−Trivial Partitions across Reviews
Number of Reviews
Manually Examined (58%)
User Study (40%)
Not Examined (1.4%)
Non−Trivial Partitions



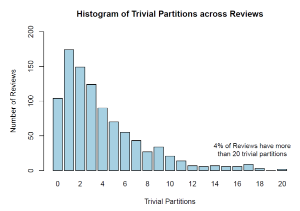
Summary



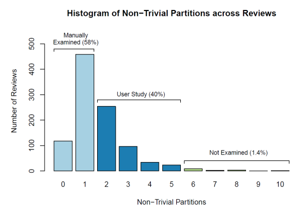
Non-trivial Partition 1
Trivial Partitions
Non-trivial Partition 2



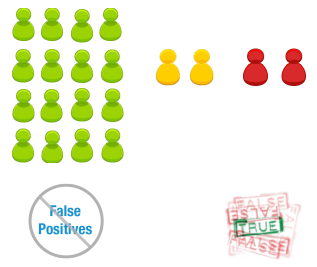
Author
ChangeSet
Cluster Changes
1
Author
2
tag
3
@refactoring
@bugfix
Reviewer



Histogram of Trivial Partitions across Reviews
Number of Reviews
Trivial Partitions
4% of Reviews have more than 20 trivial partitions



Histogram of Non−Trivial Partitions across Reviews
Number of Reviews
Non−Trivial Partitions
Manually Examined (58%)
User Study (40%)
Not Examined (1.4%)



False Positives