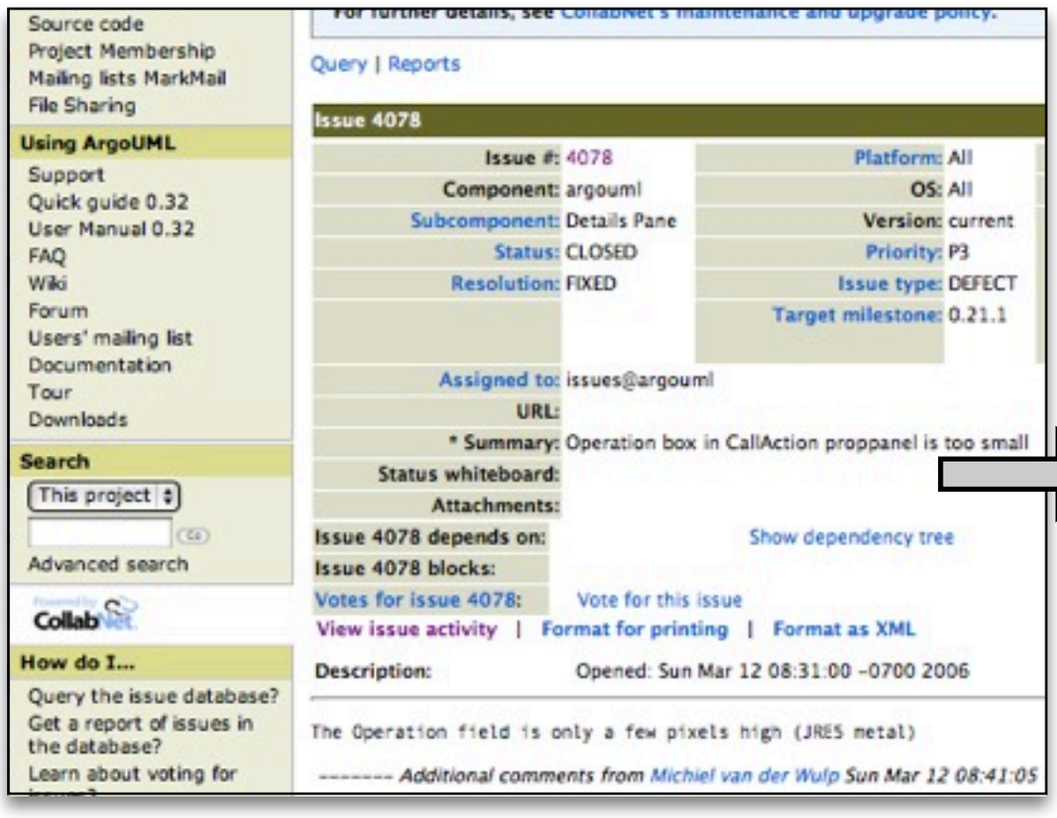
Triaging Incoming Change Requests: Bug or Commit History, or Code Authorship ?

Mario Linares-Vásquez, Kamal Hossen, Hoang Dang,
Huzefa Kagdi, Malcom Gethers, Denys Poshyvanyk

Recommending developers



Denys?

Huzefa?

Malcom?

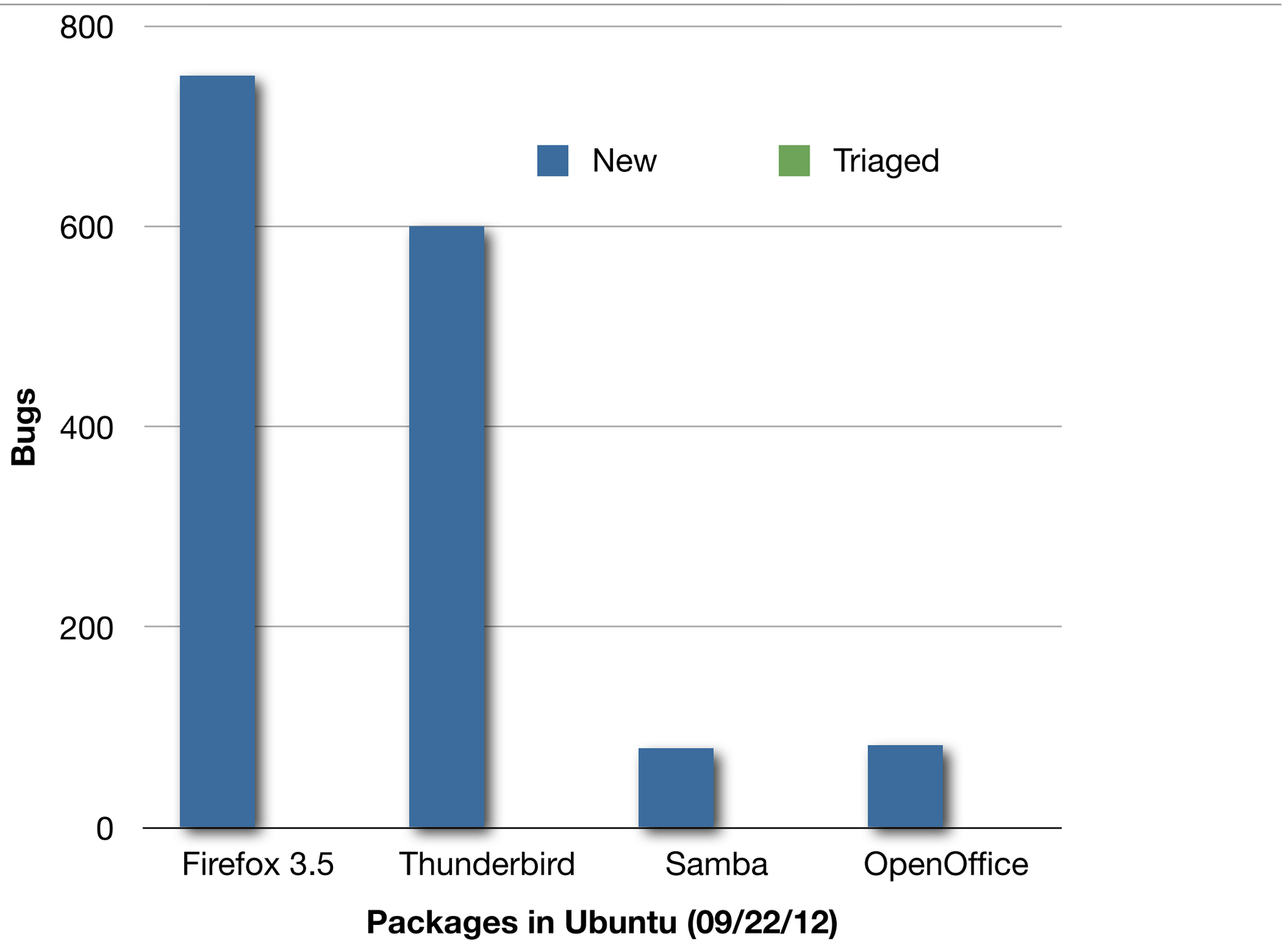
Kamal?

Hoang?

.....

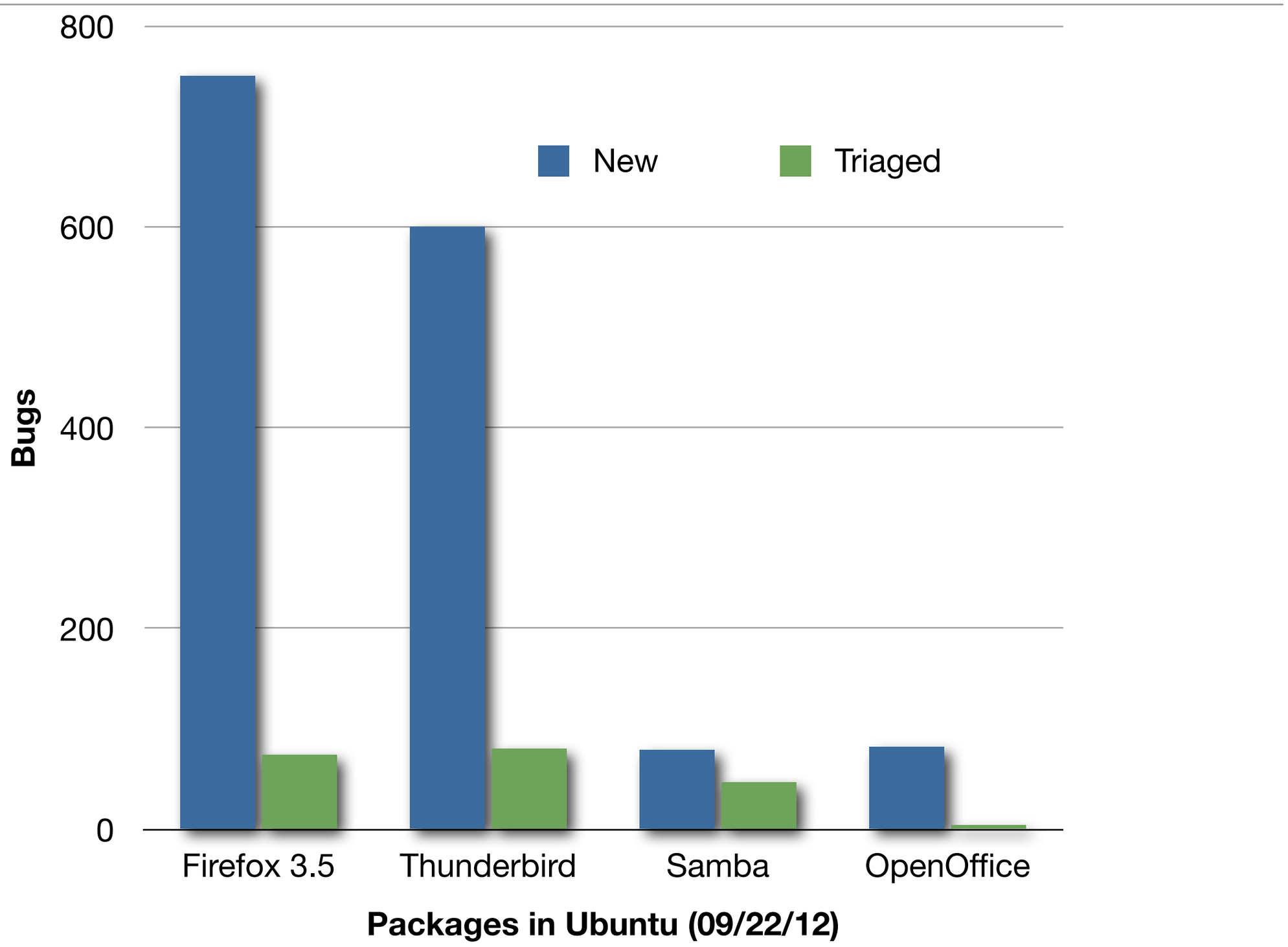
Mario?

Open source projects



http://blog.qa.ubuntu.com/qapkgstatus/

Open source projects



http://blog.qa.ubuntu.com/qapkgstatus/

Imagine you are a project manager...

200 Change requests

and more are coming....

You

Denys

Huzefa

Malcom

Kamal

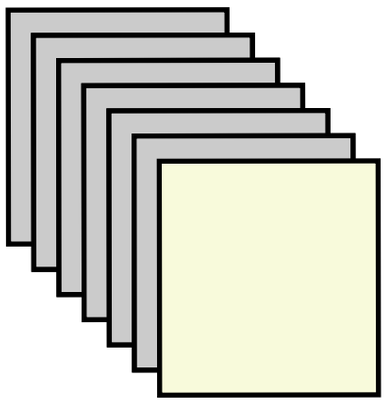
Hoang

.....

Mario

Imagine you are a project manager...

200 Change requests



You

Denys

Huzefa

Malcom

Kamal

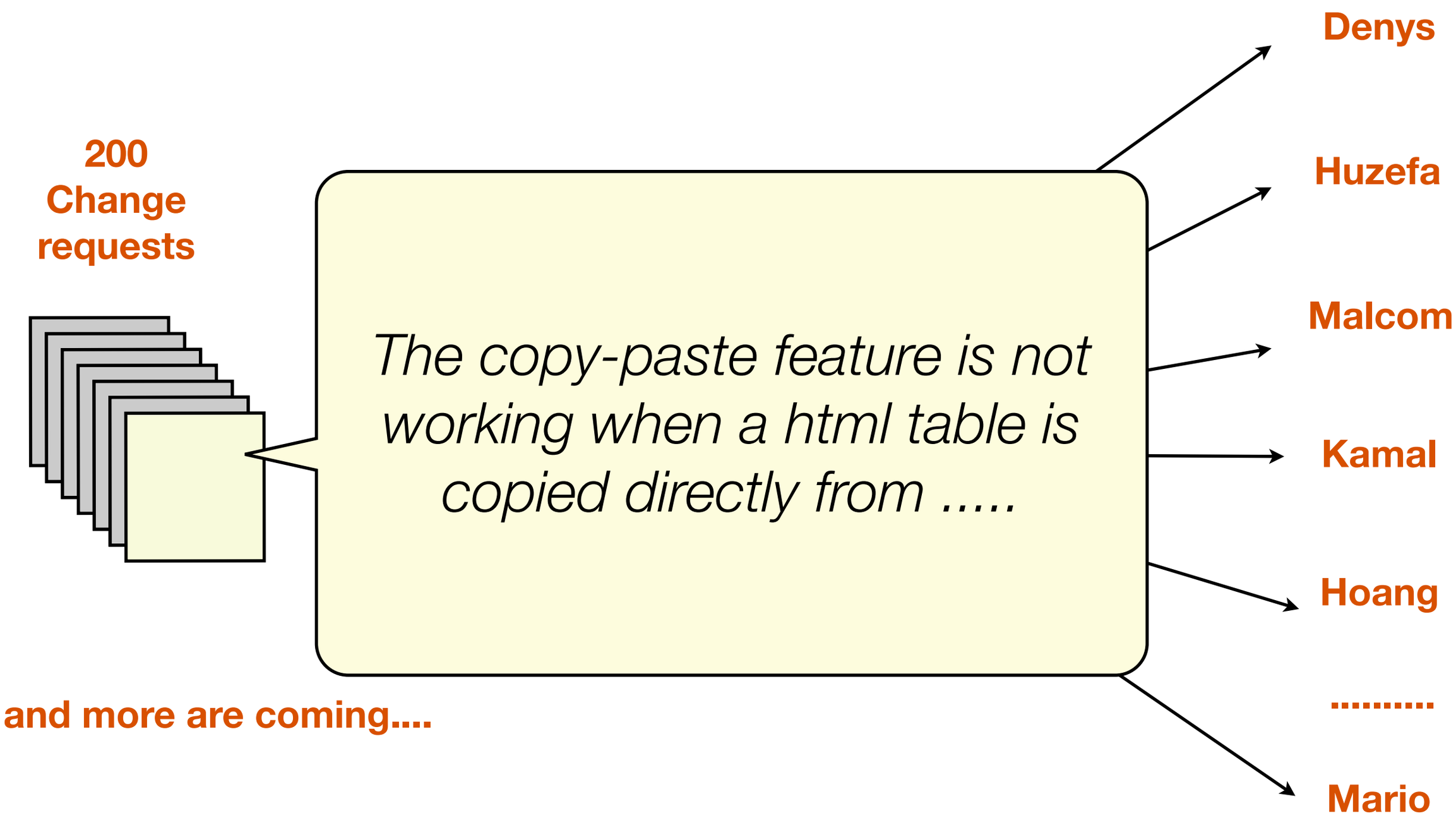
Hoang

.....

Mario

and more are coming....

Imagine you are a project manager...



Imagine you are a project manager...

Hey Doc.. Could you fix this bug ??

Denys

Imagine you are a project manager...

Hey Doc.. Could you fix this bug ??

Not mine.. please talk to Huzefa. He is the guy

Denys

Imagine you are a project manager...

Hey Doc.. Could you fix this bug ??

Not mine.. please talk to Huzefa. He is the guy

Denys

Huzefa.. Could you fix this bug ??

Huzefa

Imagine you are a project manager...

Hey Doc.. Could you fix this bug ??

Not mine.. please talk to Huzefa. He is the guy

Denys

Huzefa.. Could you fix this bug ??

Sorry man. I was watching the EuroCup final... Mario is in charge of the copy-paste feature

Huzefa

Imagine you are a project manager...

Hey Doc.. Could you fix this bug ??

Not mine.. please talk to Huzefa. He is the guy

Denys

Huzefa.. Could you fix this bug ??

Sorry man. I was watching the EuroCup final... Mario is in charge of the copy-paste feature

Huzefa

however... Mario is on holiday at Cartagena....

Challenges

- You should know the features implemented in the application
- You should know the skills of your developers
- You should know about commit and change request history

Challenges

What if the system is an open source project ?

- You should know the features implemented in the application.
- You should know the skills of your developers
- You should know about commit and change request history

Challenges

What if the system is an open source project ?

..or you have to deal with many incoming change requests every day?

Challenges

What if the system is an open source project ?

..or you have to deal with many incoming change requests every day?

Do you still have time to assign change requests manually?

Recommending developers

- **ICSE 2006 - Anvik et al.**

Who Should Fix This Bug?

John Anvik, Lyndon Hiew and Gail C. Murphy
Department of Computer Science
University of British Columbia
{janvik, lyndonh, murphy}@cs.ubc.ca

ABSTRACT

Open source development projects typically support an open bug repository to which both developers and users can report bugs. The reports that appear in this repository must be triaged to determine if the report is one which requires attention and if it is, which developer will be assigned the responsibility of resolving the report. Large open source developments are burdened by the rate at which new bug reports appear in the bug repository. In this paper, we present

However, this potential advantage also comes with a significant cost. Each bug that is reported must be *triaged* to determine if it describes a meaningful new problem or enhancement, and if it does, it must be assigned to an appropriate developer for further handling [13]. Consider the case of the Eclipse open source project[1] over a four month period (January 1, 2005 to April 30, 2005) when 3426 reports were filed, averaging 29 reports per day. Assuming that a triager takes approximately five minutes to read and handle each report, two person-hours per day is being spent

Recommending developers

- **ICSM 2008 - Kagdi et al.**

Who Can Help Me with this Source Code Change?

Huzefa Kagdi
Department of Computer Science
Missouri University of Science and Technology
Rolla Missouri 65409
kagdih@mst.edu

Maen Hammad, Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent Ohio 44242
{mhammad, jmaletic}@cs.kent.edu

Abstract

An approach to recommend a ranked list of developers to assist in performing software changes to a particular file is presented. The ranking is based on change expertise, experience, and contributions of developers, as derived from the analysis of the previous commits involving the specific file in question. The commits are obtained from a software system's version control repositories (e.g., Subversion). The basic premise is that a developer who has substantially

Fortunately, all this knowledge does not completely disappear when developers or managers leave a project. Version control systems keep an excellent record of who changed a file and when the change occurred. Here, we present an approach and tool, called *xFinder*, that recommends a ranked list of developers who are very likely to have good knowledge of the file(s) planned to be modified. This ranked list is obtained by mining the historical records found in the commits that are stored in software repositories of the project.

Recommending developers

- **ICPC 2009 - Kagdi and Poshyvanyk**

Who Can Help Me with this Change Request?

Huzefa Kagdi[1], Denys Poshyvanyk[2]

[1]*Department of Computer Science*
Missouri University of Science and Technology
Rolla, MO 65409
kagdih@mst.edu

[2]*Computer Science Department*
The College of William and Mary
Williamsburg, VA 23185
denys@cs.wm.edu

Abstract

An approach to recommend a ranked list of developers to assist in performing software changes given a textual change request is presented. The approach employs a two-fold strategy. First, a technique based on information retrieval is put at work to locate the relevant units of source code, e.g., files, classes, and methods, to a given change request. These units of source code are then fed to a technique that recommends developers based on their source code change expertise, experience, and contributions, as derived from the analysis of the previous commits. The commits are obtained from a software system's version control repositories (e.g., Subversion). The approach is

best suited to help with an incoming change request. The combined techniques are an Information Retrieval (IR) based technique that uses Latent Semantic Indexing (LSI) [9] for *concept location* [16] and an approach that is based on Mining Software Repositories (MSR) [14] to recommend a ranked list of candidate developers for source code change [15].

We use the umbrella term *concept* to generally refer to the textual description of the change request irrespective of its specific intent (e.g., description of a new feature that needs to be added or a bug that needs to be fixed). In a nutshell, our approach is a two-step procedure:

1. Given a concept description, we use LSI technique to locate a ranked list of relevant units of source

Recommending developers

- **FSE 2009 - Jeong et al.**

Improving Bug Triage with Bug Tossing Graphs

Gaeul Jeong *
Seoul National University
gejeong@ropas.snu.ac.kr

Sunghun Kim †
Hong Kong University of Science and Technology
hunkim@cse.ust.hk

Thomas Zimmermann
Microsoft Research
tz@acm.org

ABSTRACT

A bug report is typically assigned to a single developer who is then responsible for fixing the bug. In Mozilla and Eclipse, between 37%-44% of bug reports are "tossed" (reassigned) to other developers, for example because the bug has been assigned by accident or another developer with additional expertise is needed. In any case, tossing increases the time-to-correction for a bug.

In this paper, we introduce a graph model based on Markov chains, which captures bug tossing history. This model has several desirable qualities. First, it reveals developer networks which can be used to discover team structures and to find suitable experts for a new task. Second, it helps to better assign developers to bug

1. INTRODUCTION

The timely identification and correction of bugs are very important software engineering practices. To handle a large number of bugs, bug tracking systems such as Bugzilla [9] are widely used. However, most bugs are assigned manually to developers, which is a labor-intensive task, especially for large software projects. For example, the Eclipse and Mozilla projects receive several hundred bug reports per day and assign each of them to one of the several thousand developers. This is not an easy task and is often error-prone.

Once a bug report has been assigned, developers can reassign the bug to other developers; we call this process *bug tossing*. For this

Recommending developers

- **FSE 2011 - Tamrawi et al.**

Fuzzy Set and Cache-based Approach for Bug Triaging

Ahmed Tamrawi
atamrawi@iastate.edu

Tung Thanh Nguyen
tung@iastate.edu

Jafar M. Al-Kofahi
jafar@iastate.edu

Tien N. Nguyen
tien@iastate.edu

Electrical and Computer Engineering Department
Iowa State University
Ames, IA 50011, USA

ABSTRACT

Bug triaging aims to assign a bug to the most appropriate fixer. That task is crucial in reducing time and efforts in a bug fixing process. In this paper, we propose Bugzie, a novel approach for automatic bug triaging based on fuzzy set and cache-based modeling of the bug-fixing expertise of developers. Bugzie considers a software system to have multiple technical aspects, each of which is associated with technical terms. For each technical term, it uses a fuzzy set to represent the developers who are capable/competent of fixing the bugs relevant to the corresponding aspect. The fixing correlation of a developer toward a technical term is represented by his/her membership score toward the corresponding fuzzy set. The score is calculated based on the bug reports that (s)he has fixed, and is updated as the newly fixed bug reports are available. For a new bug report, Bugzie combines the fuzzy sets corresponding to its terms and ranks the developers.

To support developers in this task, we propose Bugzie, a novel fuzzy set and cache-based approach for automatic bug triaging. Bugzie considers a software system to have a collection of technical aspects/concerns, which are described via the corresponding technical terms appearing in software artifacts. Among the artifacts, a bug report describes an issue(s) related to some technical aspects/concerns via the corresponding technical terms. Therefore, in Bugzie, the key research question is that *given a bug report, how to determine who have the most bug-fixing capability/expertise with respect to the reported technical aspect(s)/issue(s).*

The key idea of Bugzie is to model the *fixing correlation/association* of developers toward a technical aspect via fuzzy sets [25]. The fixing correlation/association represents the bug-fixing capability/expertise of developers with respect to the technical aspects in a project. To realize that, the fuzzy sets are defined for the corresponding technical

Our motivation...

- Previous approaches require mining of either commit or change request repositories
- Location of relevant files using solely LSI is prone to false positives

Our motivation...

- Previous approaches require mining of either commit or change request repositories
- Location of relevant files using solely LSI is prone to false positives

Could we assign developers to change requests without mining repositories ?

Our premise..... Code authorship

*OperationNotationUml.java

```
// $Id: OperationNotationUml.java,v ... mvw Exp $
package org.argouml.uml.notation.uml;
import java.text.ParseException;
/**
 * The UML notation for an Operation.
 *
 * @author mvw@tigris.org
 */
public class OperationNotationUml extends OperationNotation {
 ...
    * properties are shown/not shown.
    *
    * @author jaap.branderhorst@xs4all.nl
    * @see java.lang.Object#toString()
    */
   public String toString() {
    ...
   }
   ...
}
```

Authors are typically found in the header comments of source code entities

Our premise..... Code authorship

*OperationNotationUml.java

```
// $Id: OperationNotationUml.java,v ... mvw Exp $
package org.argouml.uml.notation.uml;
import java.text.ParseException;
/**
 * The UML notation for an Operation.
 *
 * @author mvw@tigris.org
 */
public class OperationNotationUml extends OperationNotation {
    /**
     *Generates an operation according to the UML 1.3 notation:
     *
     *         stereotype visibility name (parameter-list) :
     *                          return-type-expression {property-string}
     *
     * For the return-type-expression: only the types of the return parameters
     * are shown.  Depending on settings in Notation, visibility and
     * properties are shown/not shown.
     *
     * @author jaap.branderhorst@xs4all.nl
     * @see java.lang.Object#toString()
     */
    public String toString() {
        .....
    }
    .....
}
```

Our premise..... Code authorship

*OperationNotationUml.java

```
// $Id: OperationNotationUml.java,v ... mvw Exp $
package org.argouml.uml.notation.uml;
import java.text.ParseException;
/**
 * The UML notation for an Operation.
 *
 * @author mvw@tigris.org
 */
public class OperationNotationUml extends OperationNotation {
```

Authors of source code entities are best equipped to tackle any changes needed in them

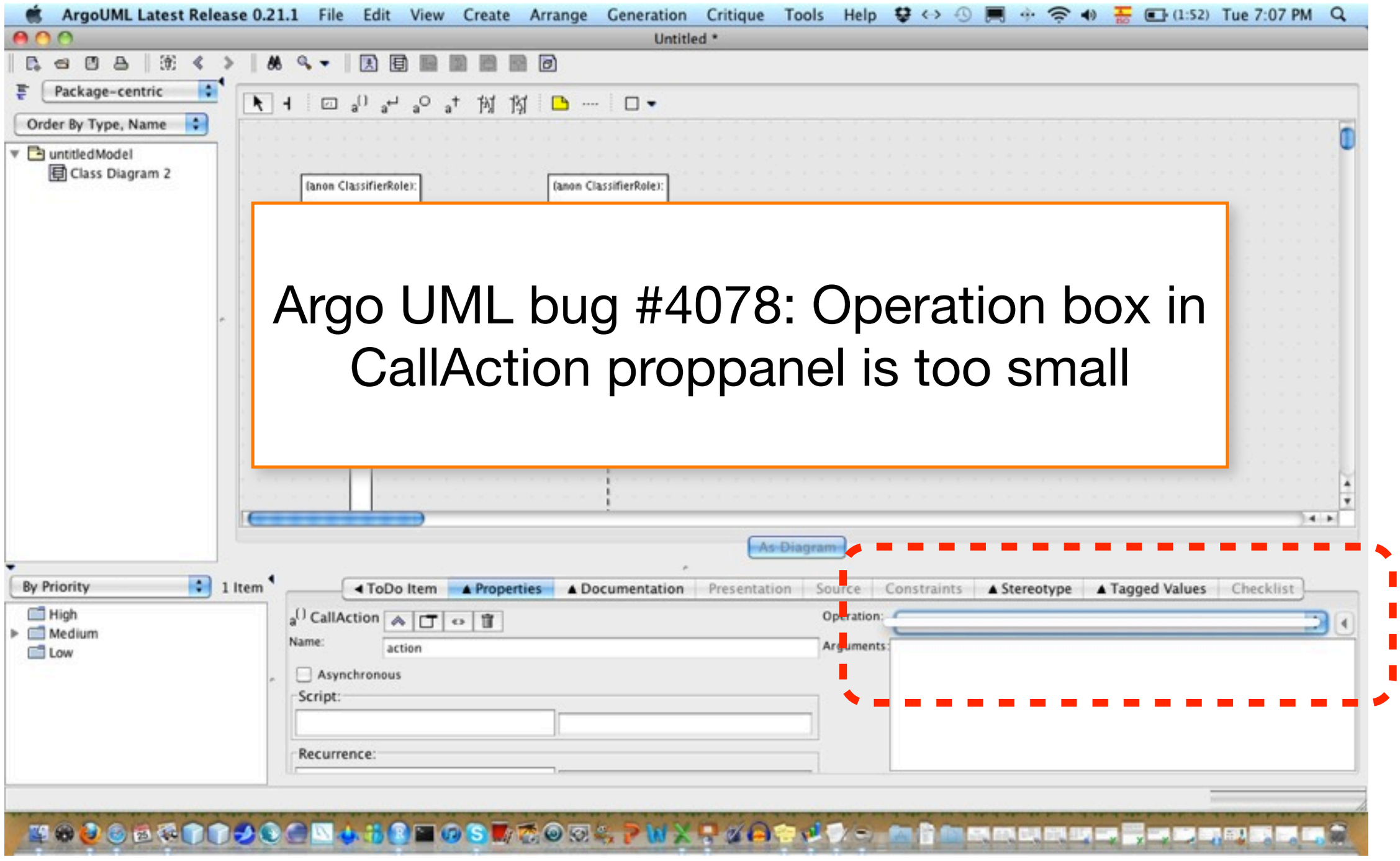
```
eters
     * properties are shown/not shown.
     *
     * @author jaap.branderhorst@xs4all.nl
     * @see java.lang.Object#toString()
     */
    public String toString() {
        ....
    }

    ....
}
```

Our solution...

1. Find the relevant code for a given change request using an IR based concept location technique

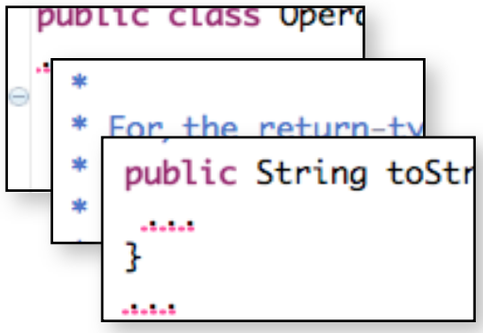
Our solution...



Our solution...

Argo UML bug #4078:
Operation box in CallAction proppanel is too small

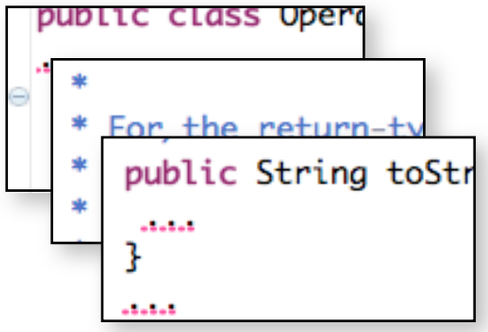
Release/version where the bug was reported



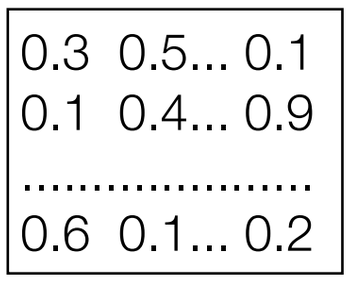
Our solution...

Argo UML bug #4078:
Operation box in CallAction proppanel is too small

Release/version where the bug was reported



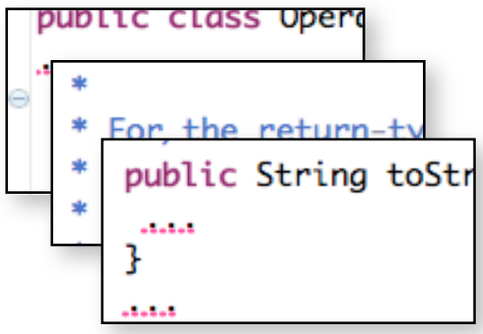
LSI



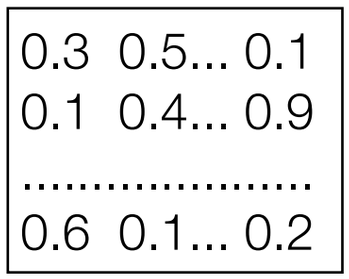
Our solution...

Argo UML bug #4078: Operation box in CallAction proppanel is too small

Release/version where the bug was reported



LSI



Top files relevant to the bug

1. mdr/CommonBehaviorHelperMDRImpl.java
2. uml/OperationNotationUml.java
3. common_behavior/PropPanelCallAction.java

.....

10. mdr/CommonBehaviorFactoryMDRImpl.java

.....

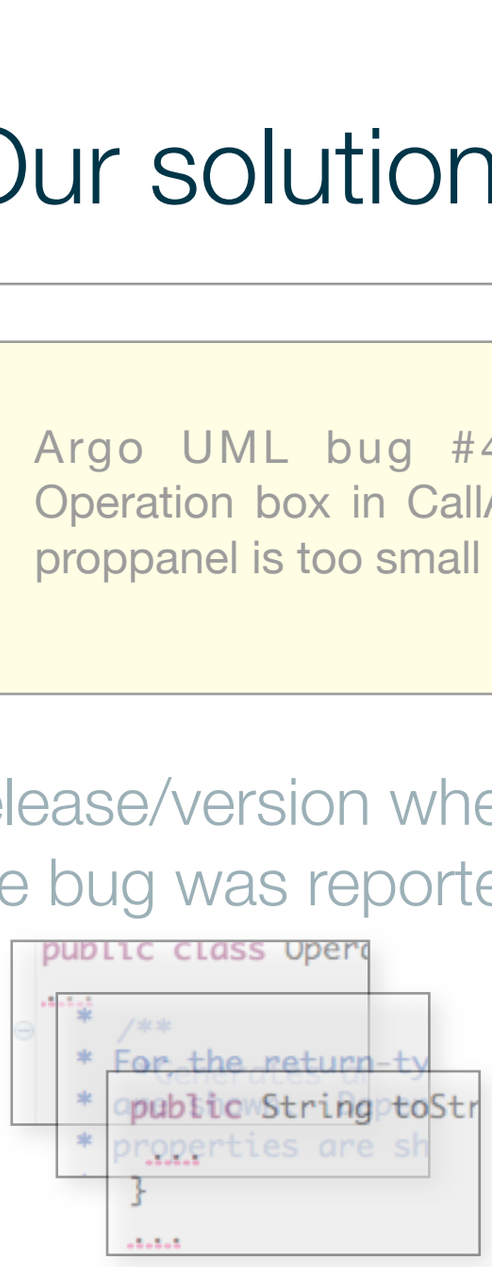
Our solution...

2. Extract authorship information from relevant code to recommend a ranked list of developers.

Our solution...

Argo UML bug #4078: Operation box in CallAction proppanel is too small

Release/version where the bug was reported



LSI

0.3 0.5... 0.1
0.1 0.4... 0.9
.....
0.6 0.1... 0.2

Top files relevant to the bug

1. mdr/CommonBehaviorHelperMDRImpl.java
2. uml/OperationNotationUml.java
3. common_behavior/PropPanelCallAction.java

.....

10. mdr/CommonBehaviorFactoryMDRImpl.java

.....

Authors extracted from relevant files

1. tfmorris, rastaman
2. mvw, jaap
3. mvw

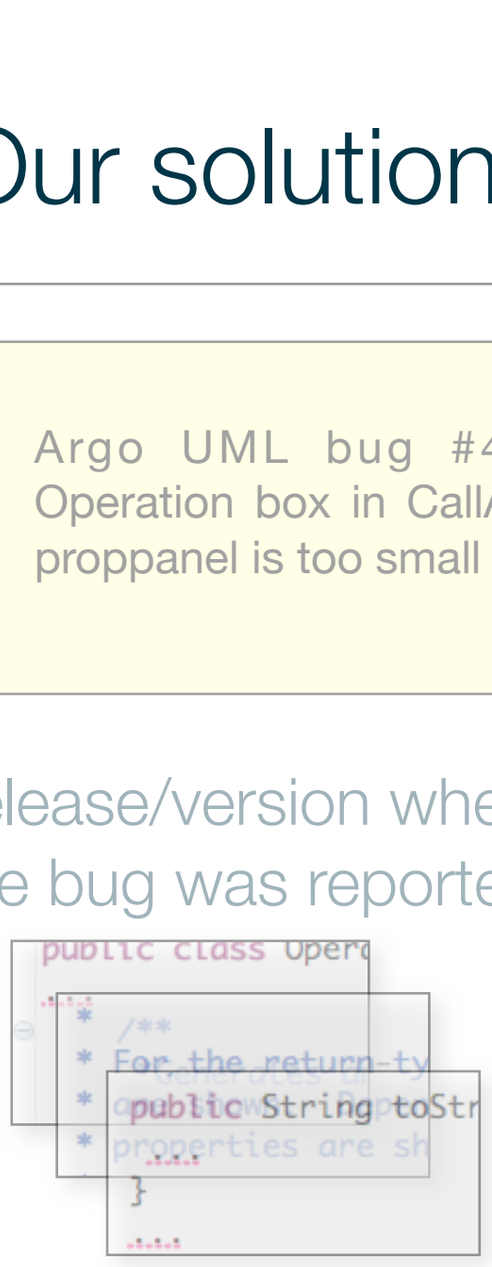
.....

10. tfmorris, rastaman, thierrylach

Our solution...

Argo UML bug #4078: Operation box in CallAction proppanel is too small

Release/version where the bug was reported



LSI

0.3 0.5... 0.1
0.1 0.4... 0.9
.....
0.6 0.1... 0.2

Top files relevant to the bug

1. mdr/CommonBehaviorHelperMDRImpl.java
2. uml/OperationNotationUml.java
3. common_behavior/PropPanelCallAction.java
.....
10. mdr/CommonBehaviorFactoryMDRImpl.java
.....

Authors extracted from relevant files

1. tfmorris, rastaman
2. mvw, jaap
3. mvw
.....
10. tfmorris, rastaman, thierrylach

Final ranked list

1. mvw (5)
2. tfmorris (3)
3. rastaman (2)
.....
8. thierrylach (1)

Our solution...

Argo UML bug #4078: Operation box in CallAction proppanel is too small

Top files relevant to the bug

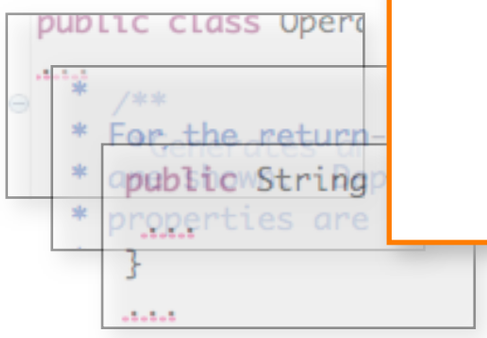
1. mdr/CommonBehaviorHelperMDRImpl.java
2. uml/OperationNotationUml.java
3. common_behavior/PropPanelCallAction.java

.....

10. mdr/CommonBehaviorFactoryMDRImpl.java

In fact, the bug was fixed by **mvw**.

Release/version
the bug was rep



Authors extracted from relevant files

1. tfmorris, rastaman
2. mvw, jaap
3. mvw

.....

10. tfmorris, rastaman, thierrylach

Final ranked list

1. **mvw (5)**
2. tfmorris (3)
3. rastaman (2)

.....

8. thierrylach (1)

Research questions

- **RQ1:** How does the accuracy of our approach compare to the other techniques based on software repository mining [Anvik et al. 2006, Kagdi and Poshyvanyk 2009]?

- **RQ2:** Is there an impact of filtering IR-based results using execution traces on the proposed approach?

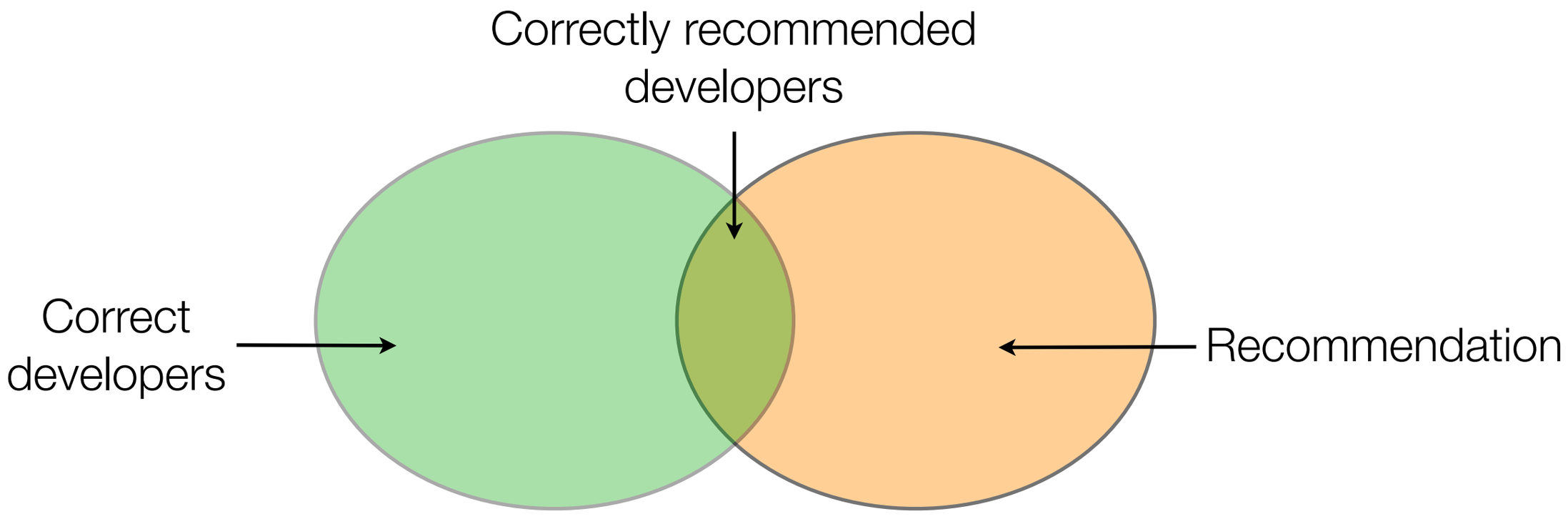
Software systems and benchmarks

System	Version	LOC	Files	Methods	Terms	Change requests (goldset)
jEdit	4.3	103896	503	6413	4372	143
ArgoUML	0.22	148892	1439	11000	5488	91
muCommander	0.8.5	76649	1069	8187	4262	92

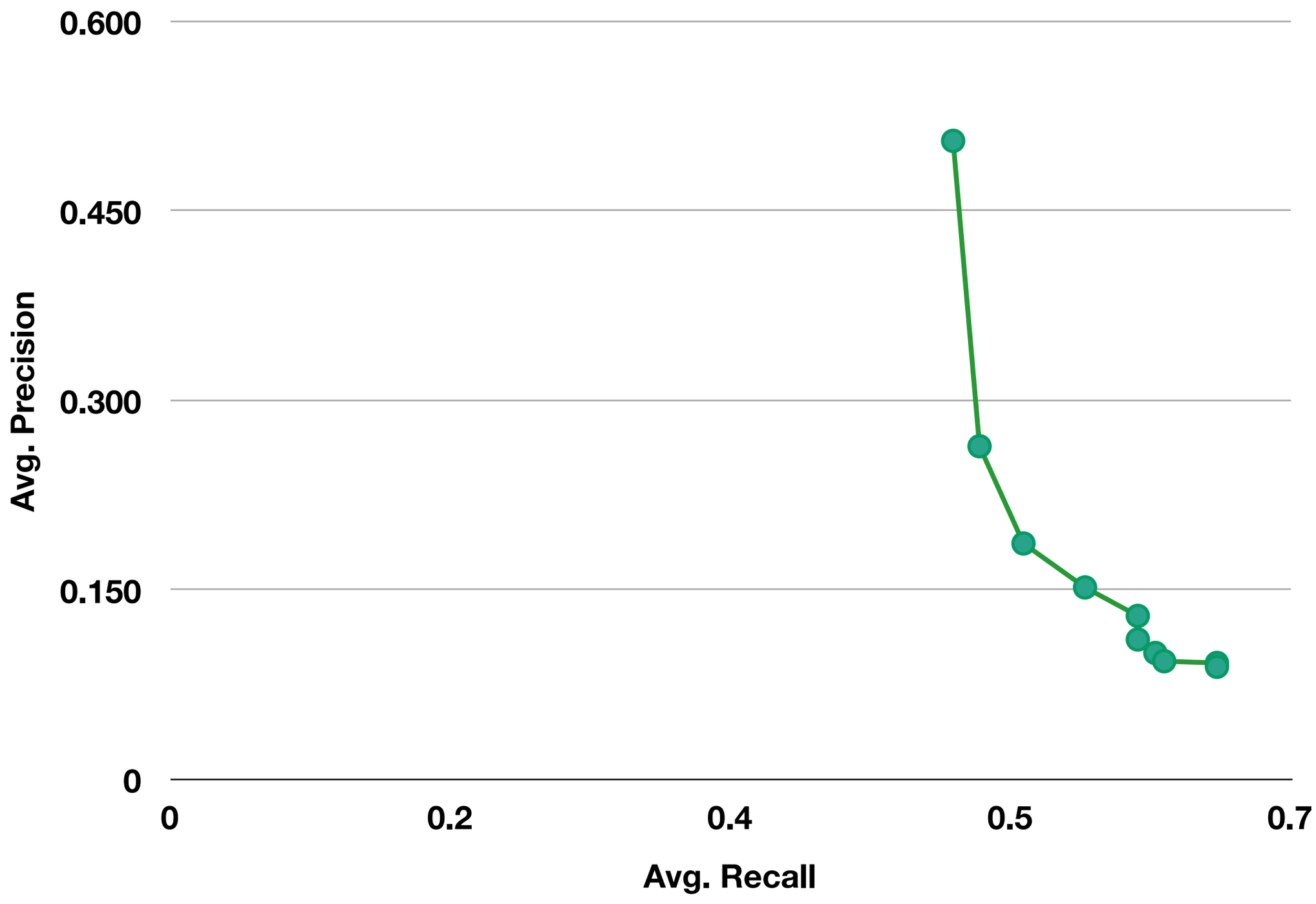
Evaluation metrics

- **Precision:** proportion of the correctly recommended developers over the total of recommendations.

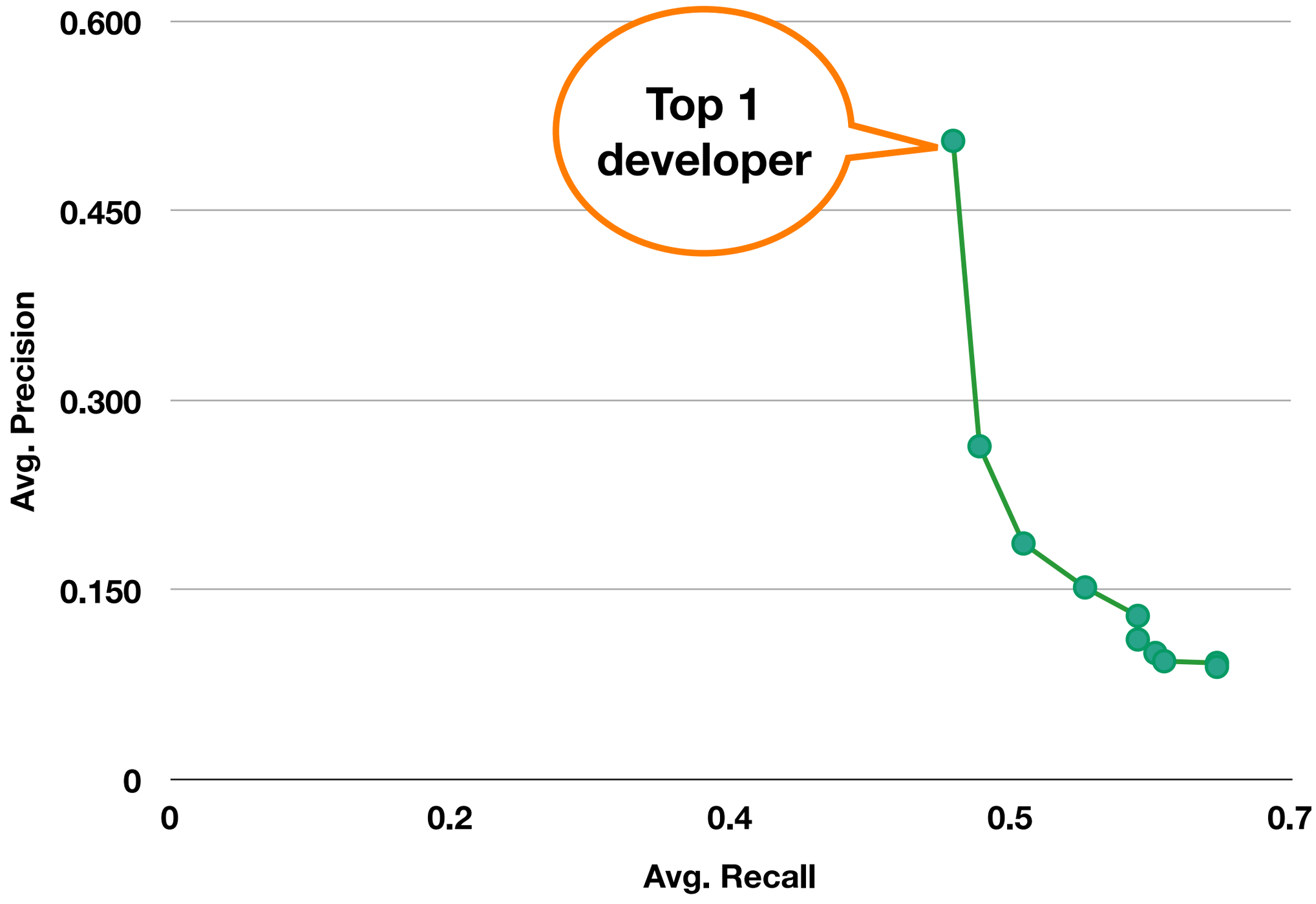
- **Recall:** proportion of the correctly recommended developers over the total of correct developers.



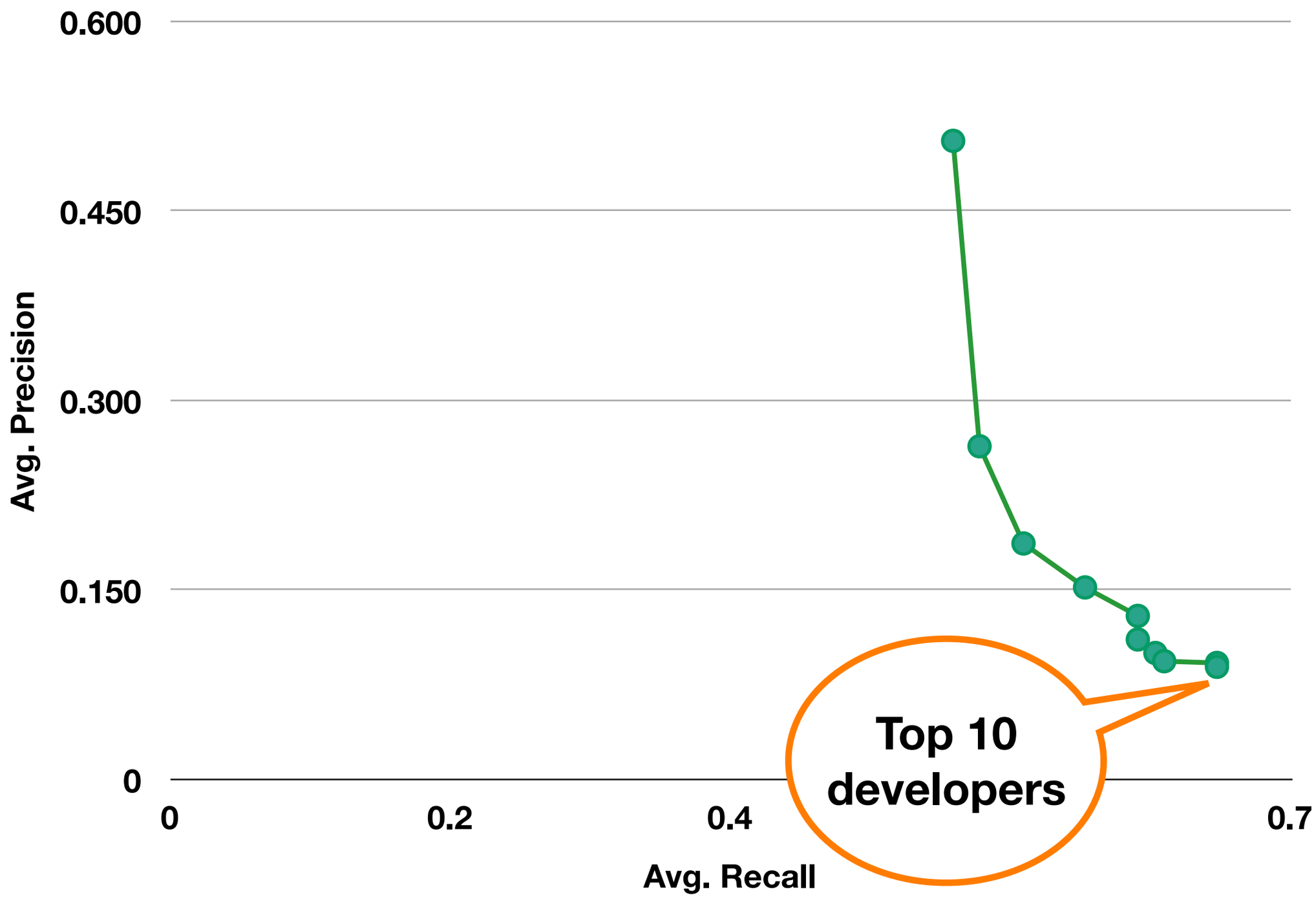
Results (ArgoUML)



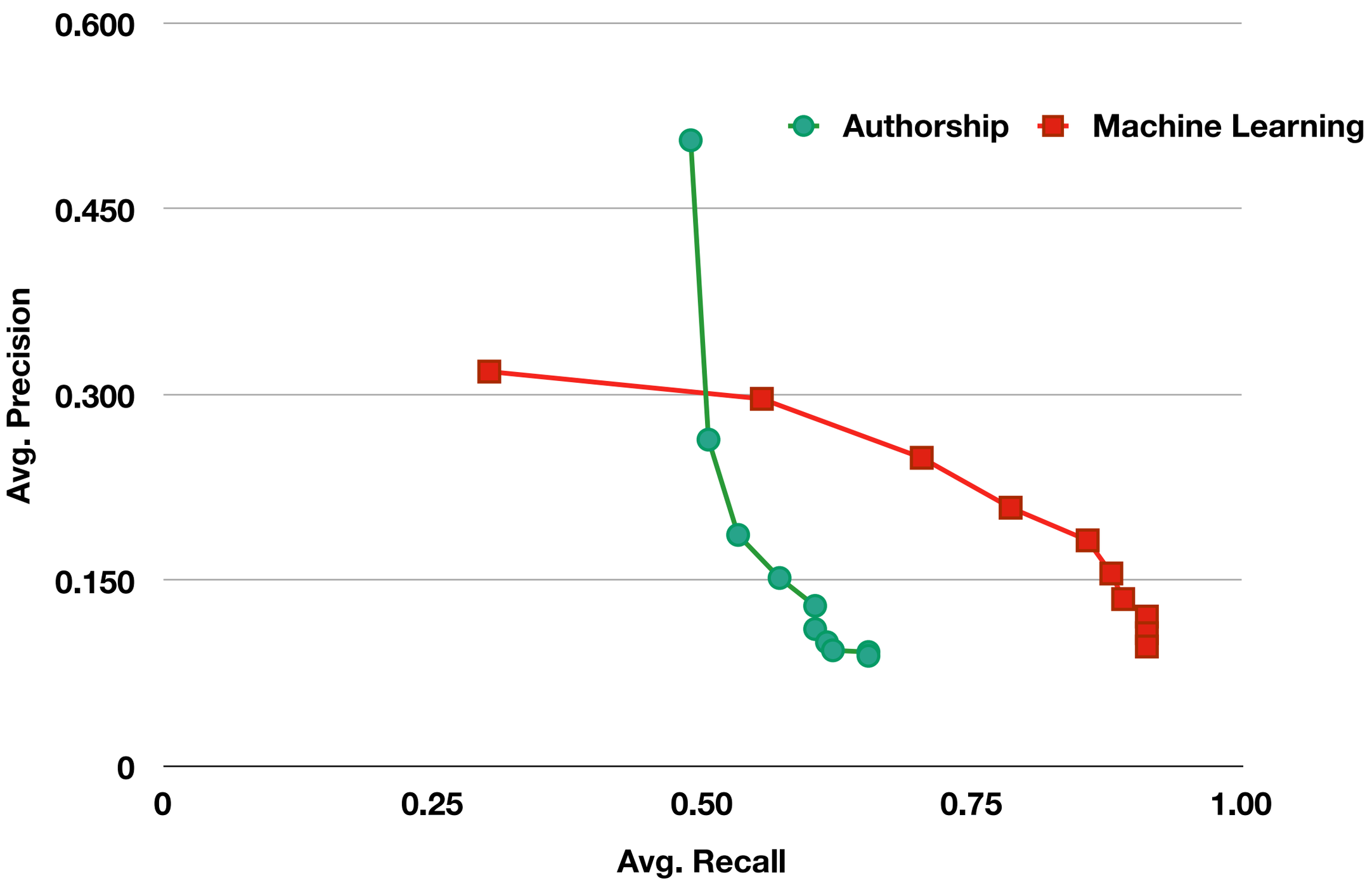
Results (ArgoUML)



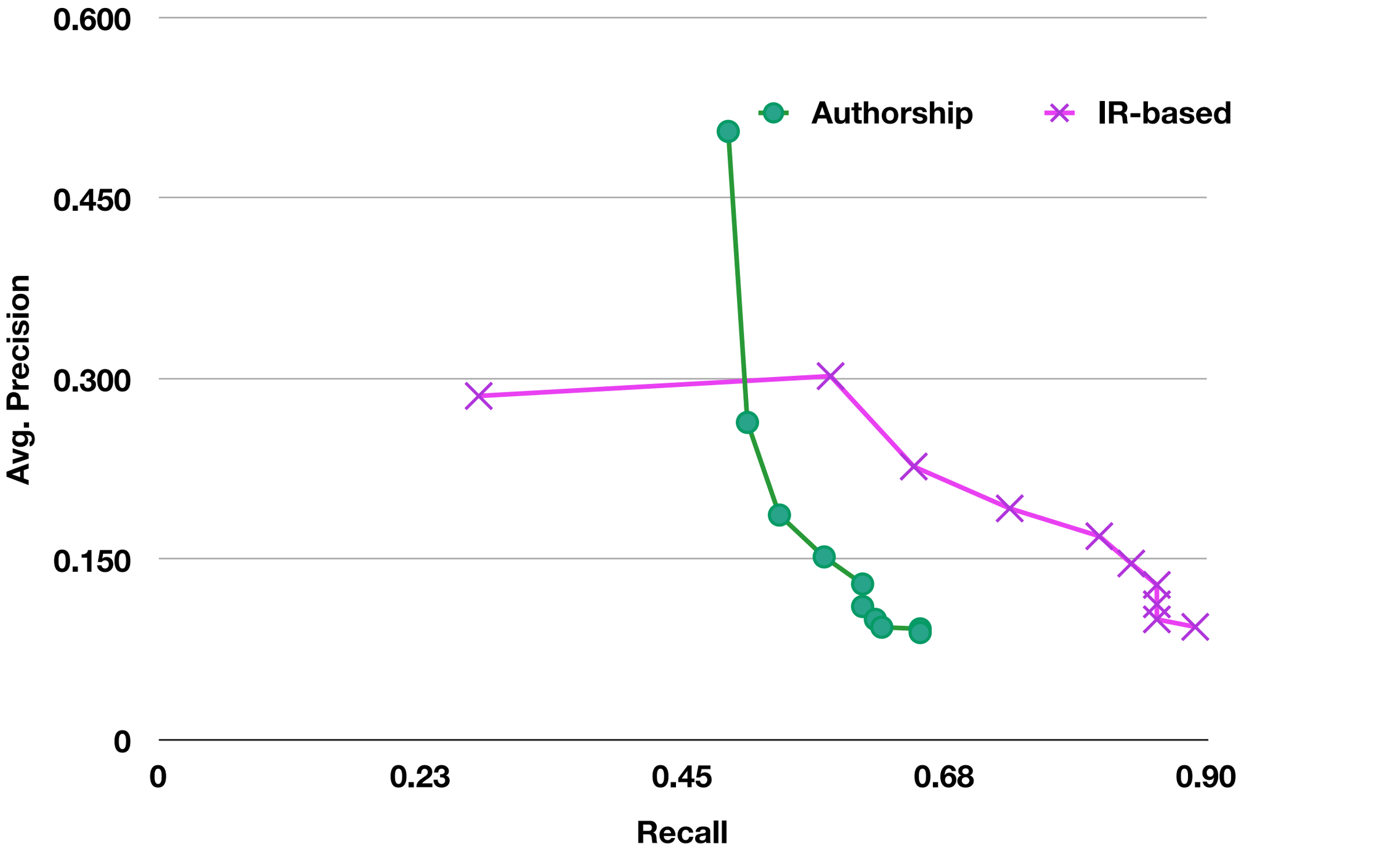
Results (ArgoUML)



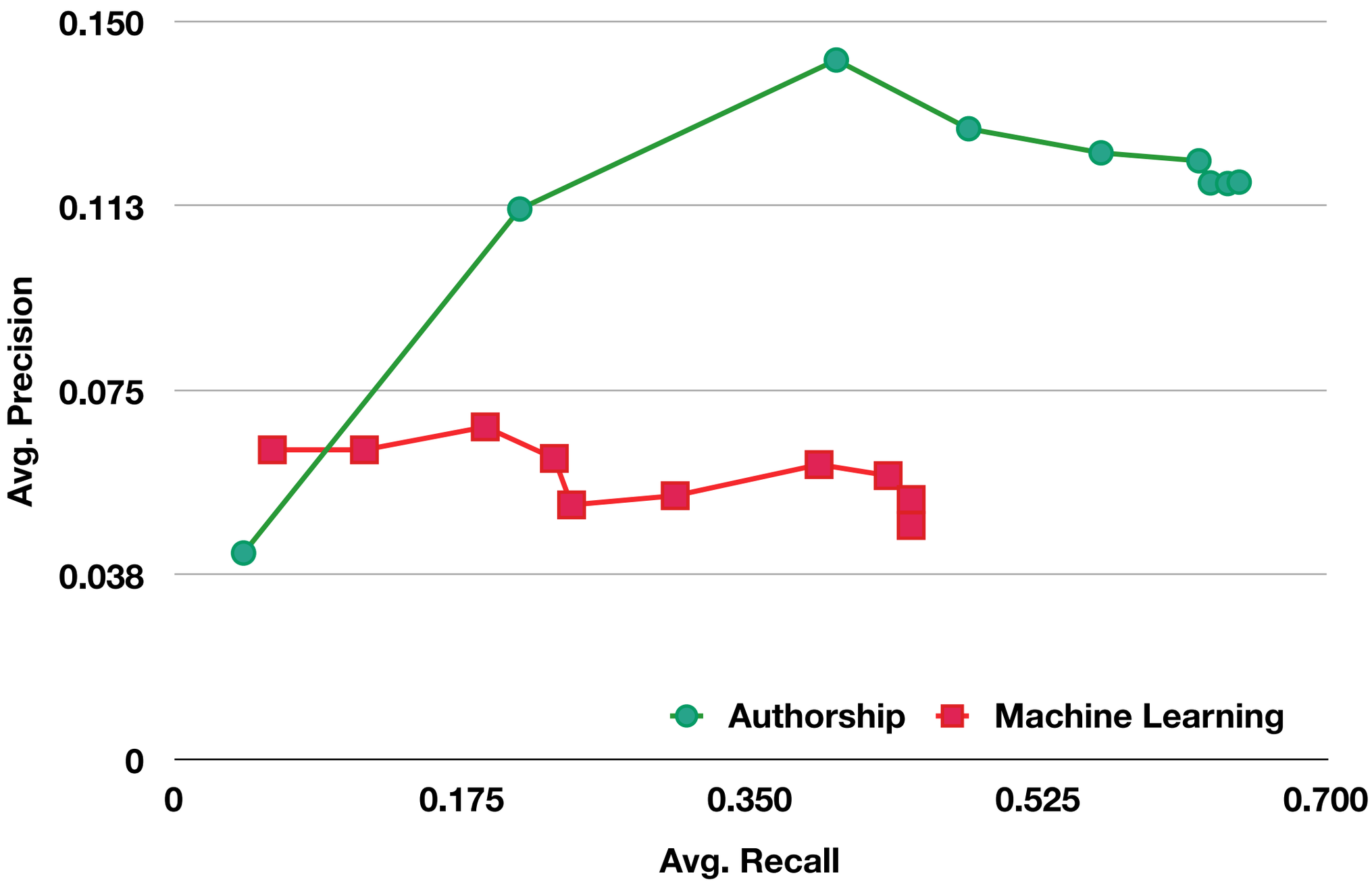
Authorship vs Machine Learning (ArgoUML)



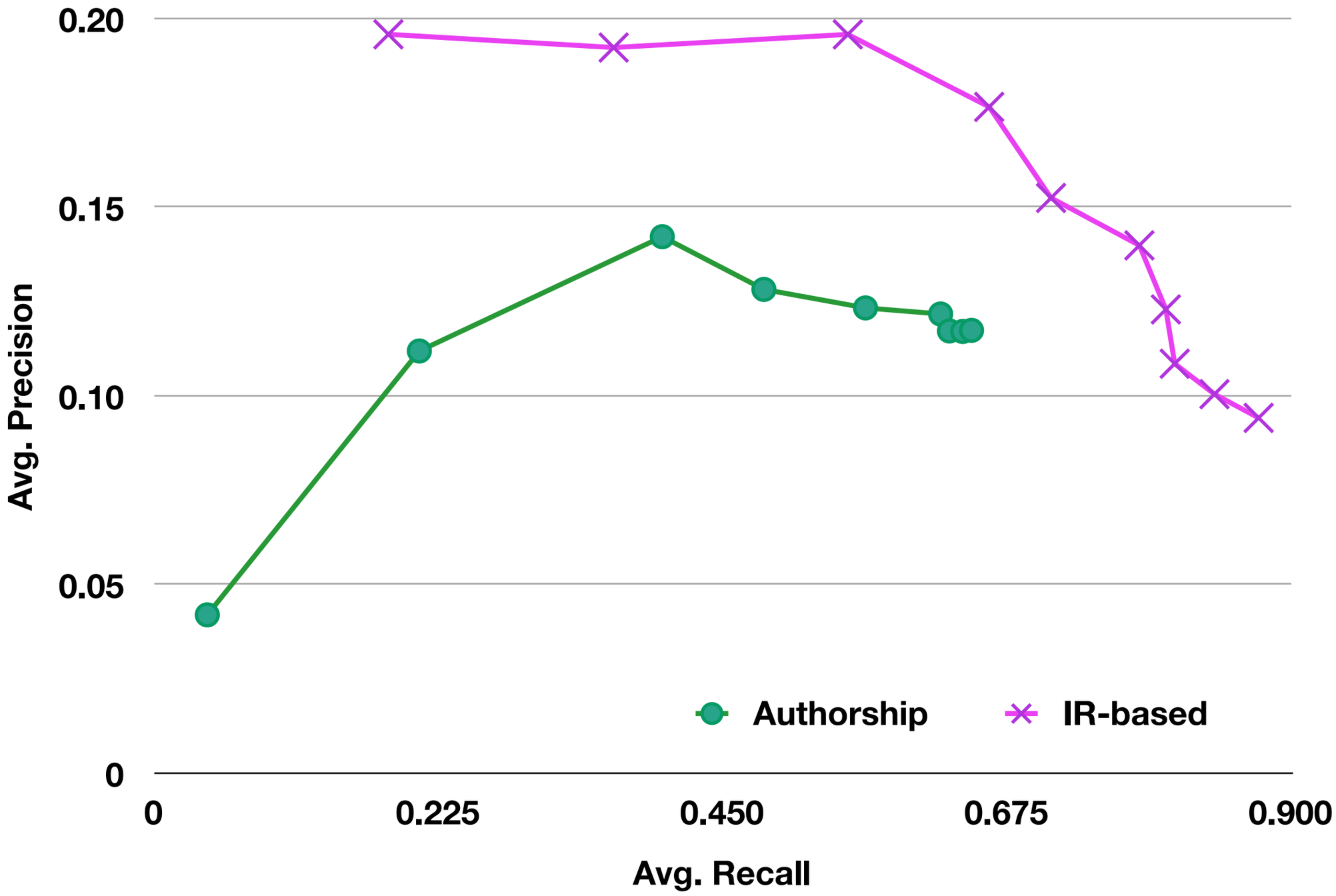
Authorship vs IR-based (ArgoUML)



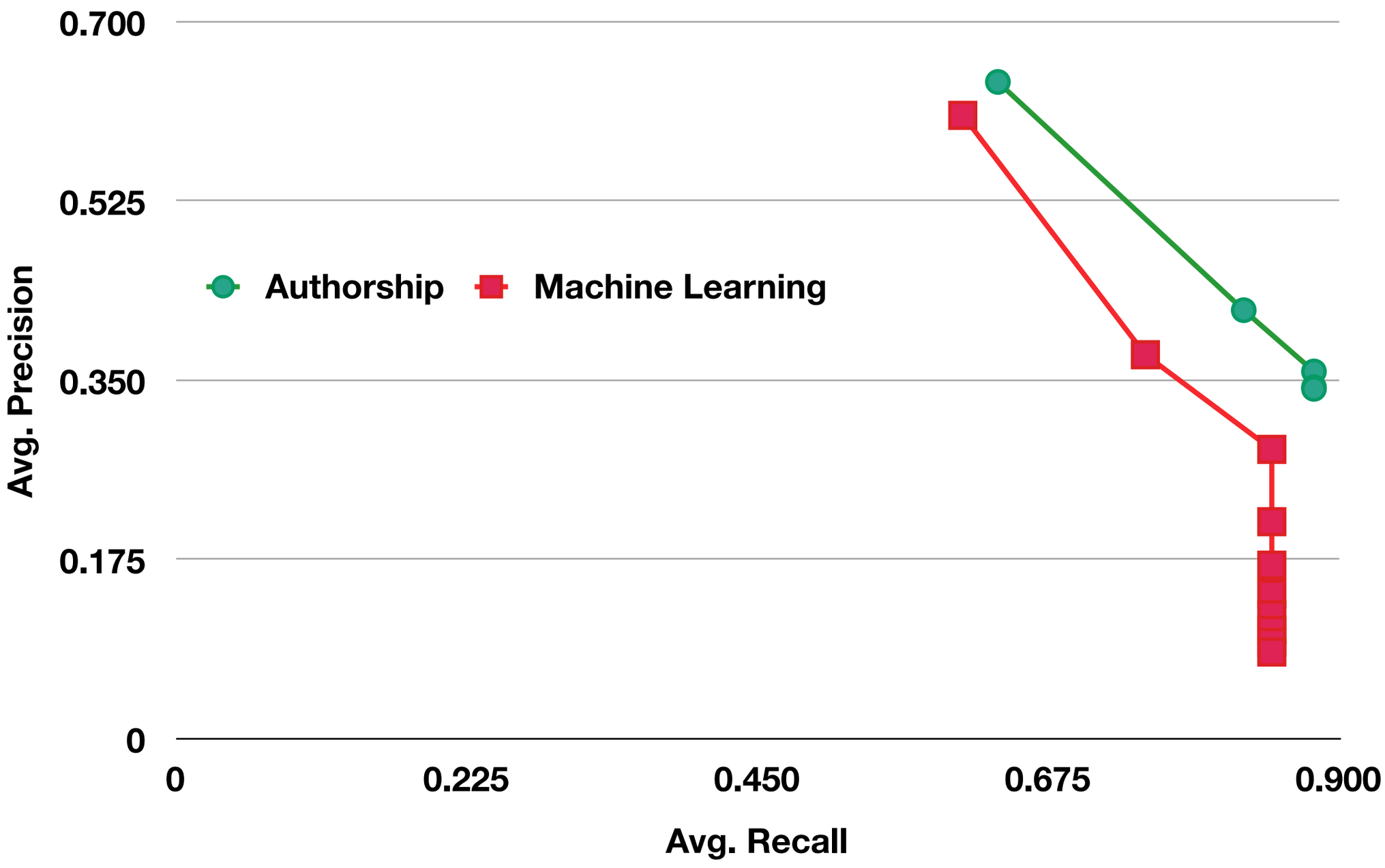
Authorship vs Machine Learning (JEdit)



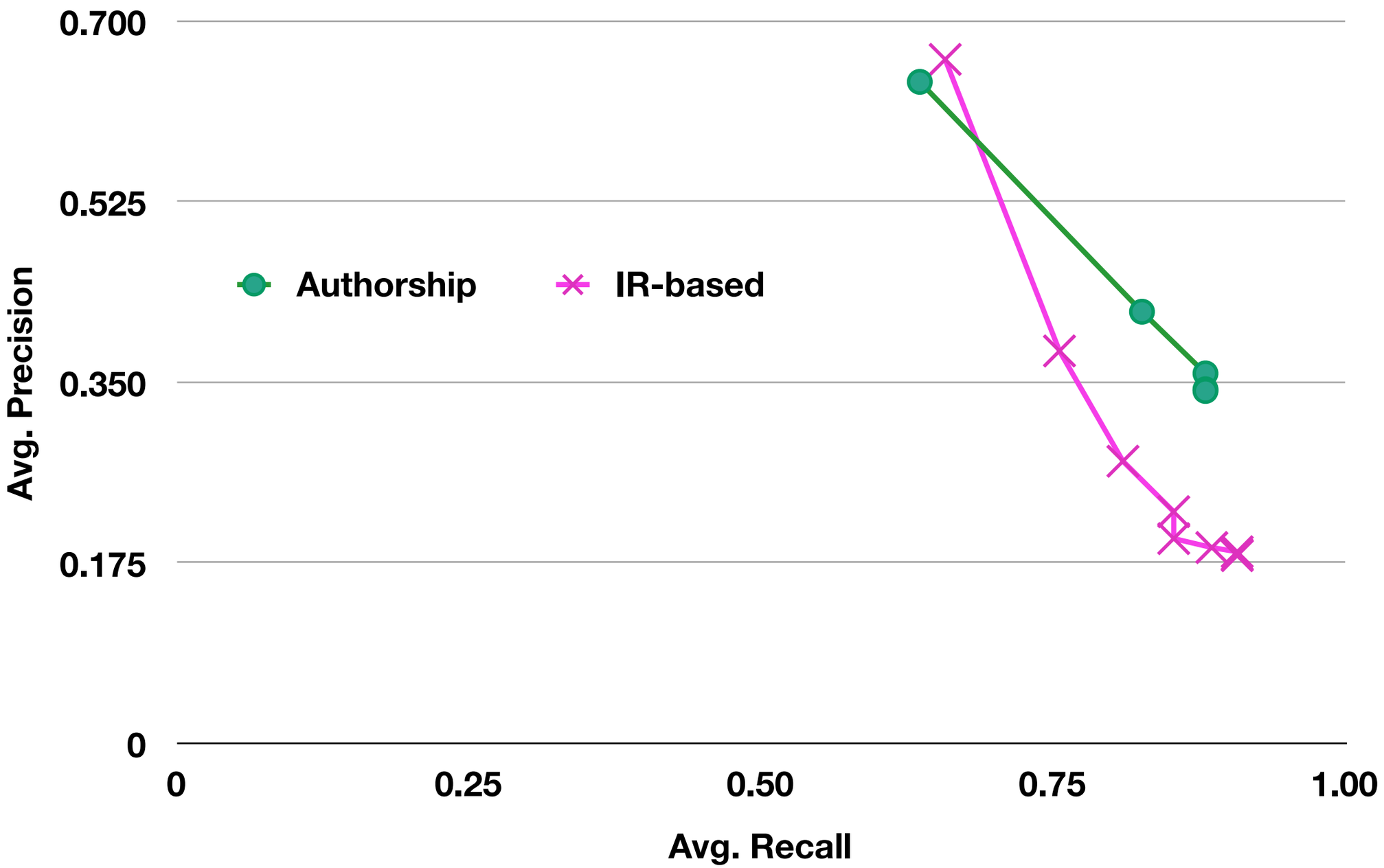
Authorship vs IR-based (JEdit)



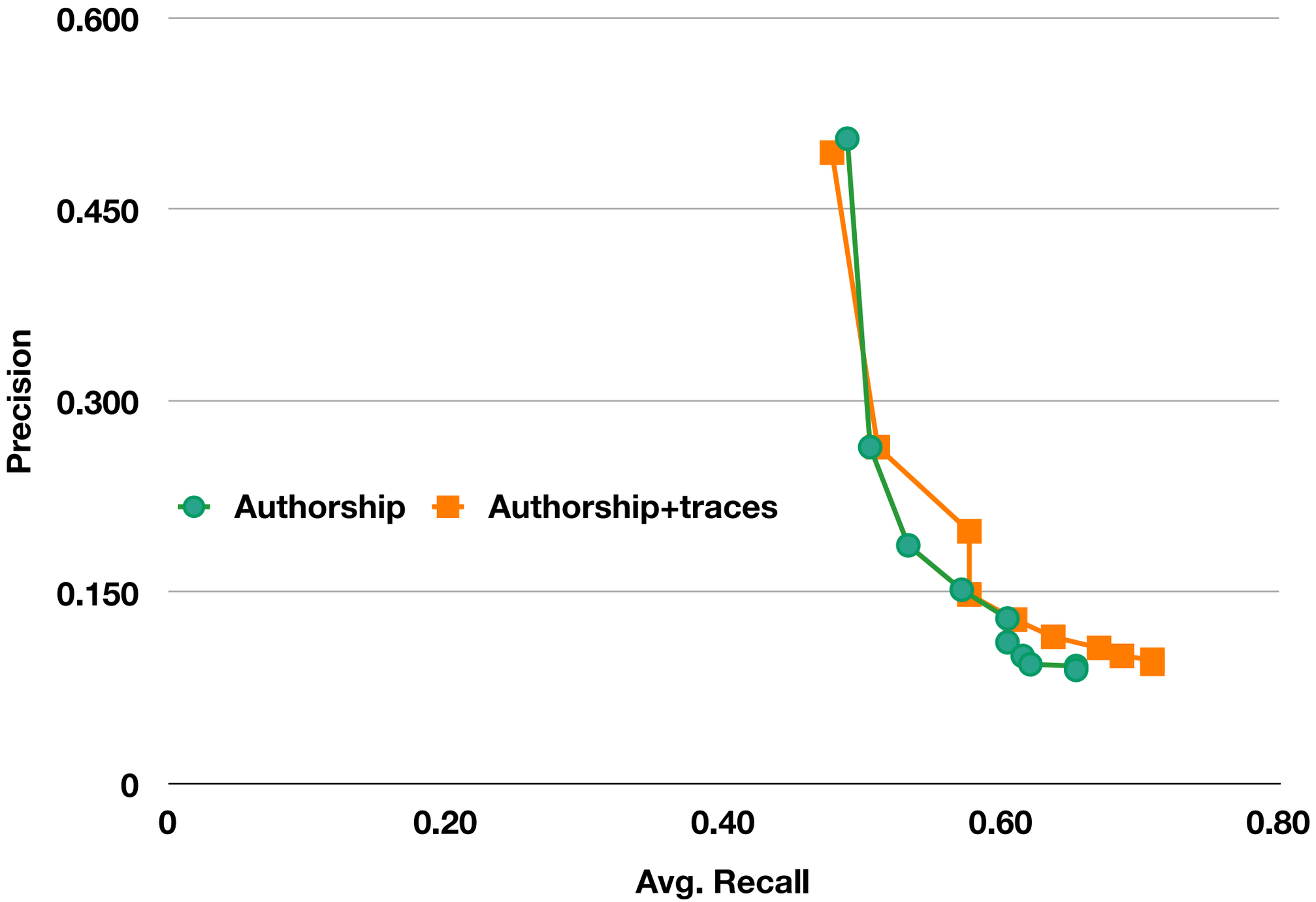
Authorship vs Machine Learning (MuCommander)



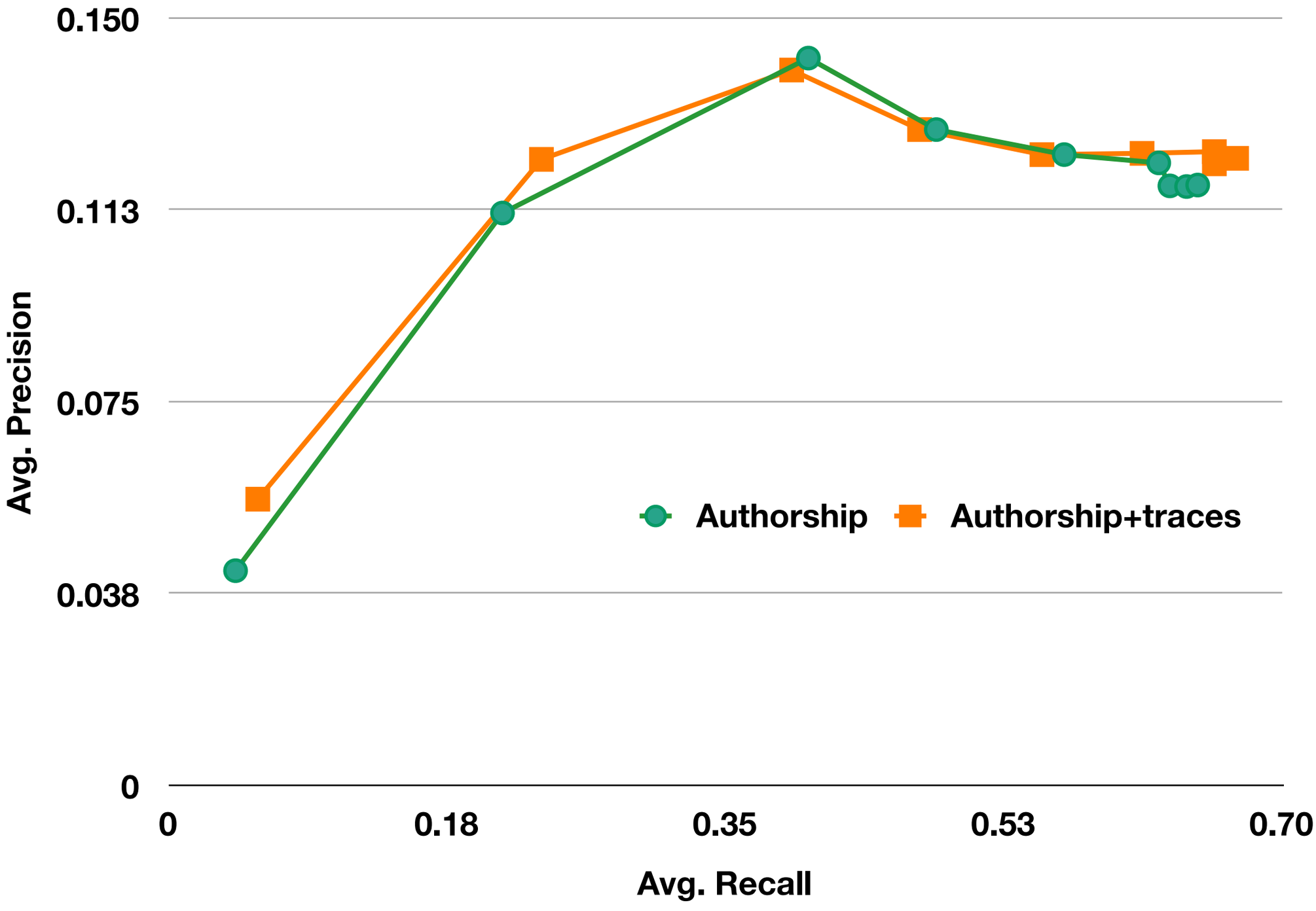
Authorship vs IR-based (MuCommander)



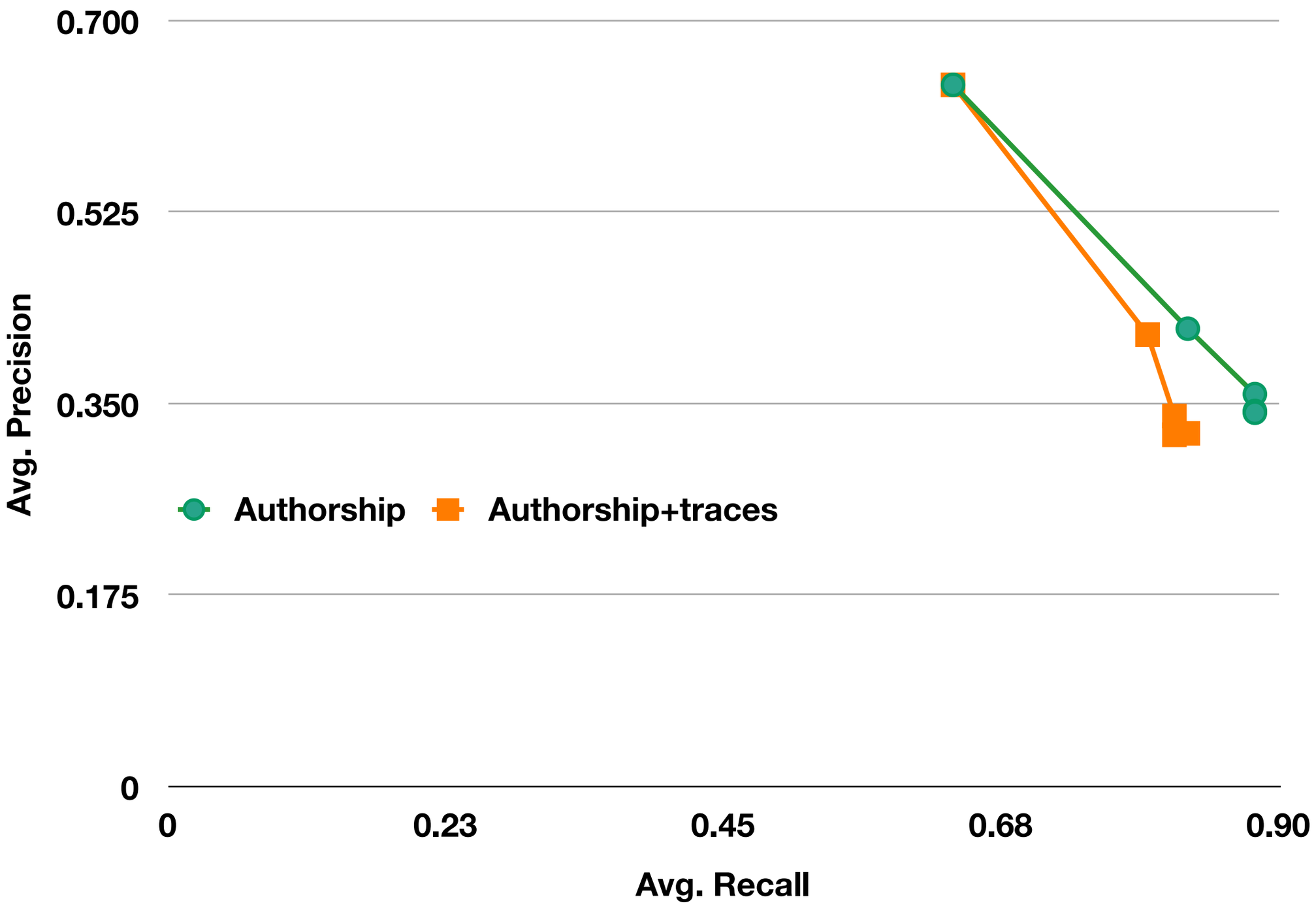
Authorship vs Authorship+traces (ArgoUML)



Authorship vs Authorship+traces (JEdit)



Authorship vs Authorship+traces (MuCommander)



Statistically Significant Difference

RQ1:

- **Authorship vs. Machine Learning**: statistically significant differences between precisions on JEdit and MuCommander

- **Authorship vs. IR-based**: statistically significant differences between precisions on MuCommander

Statistically Significant Difference

RQ2:

No statistically significant difference between authorship and using execution traces for filtering relevant files

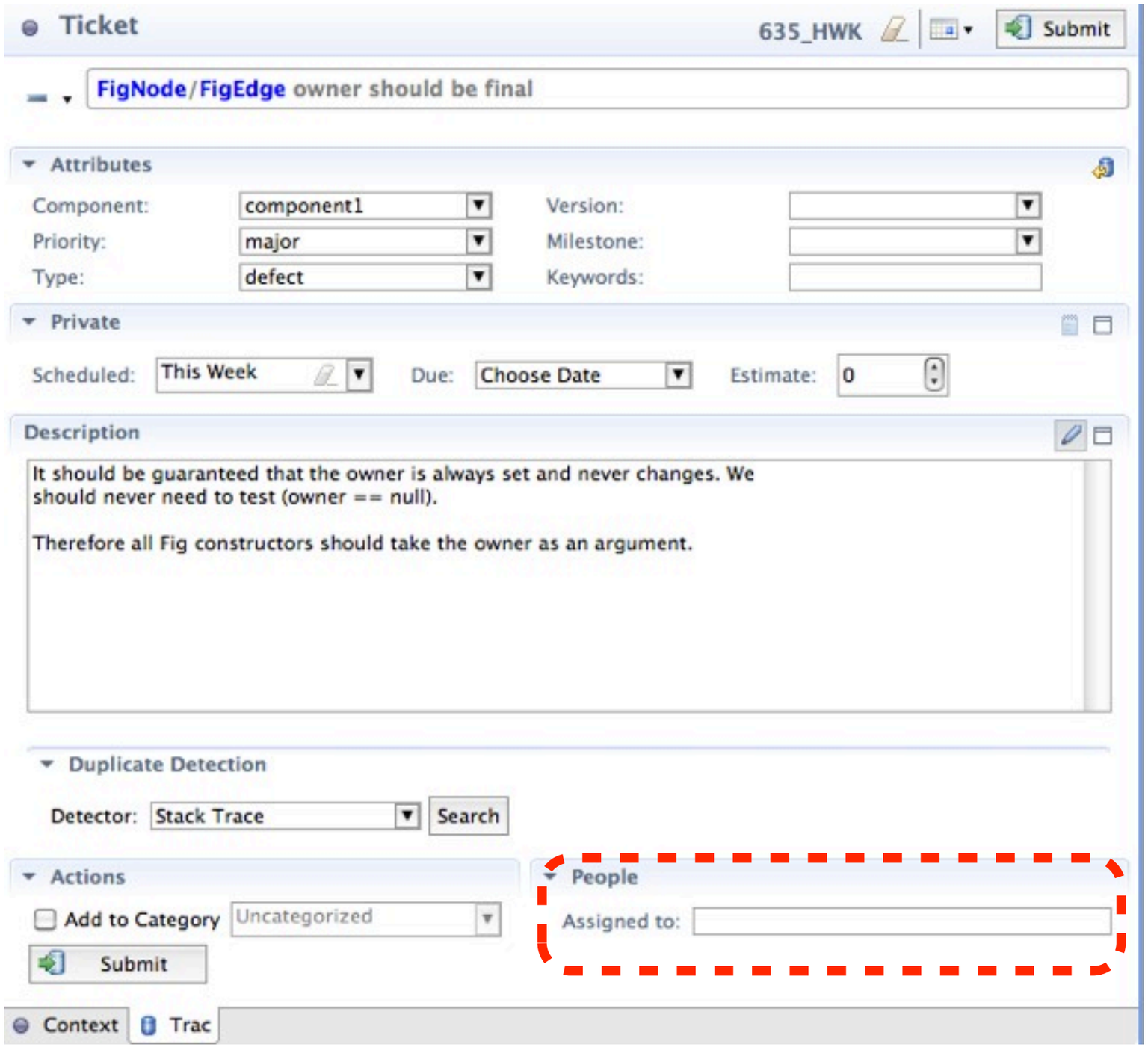
Conclusions

- Our approach **does not require mining** of either a bug or commit repository

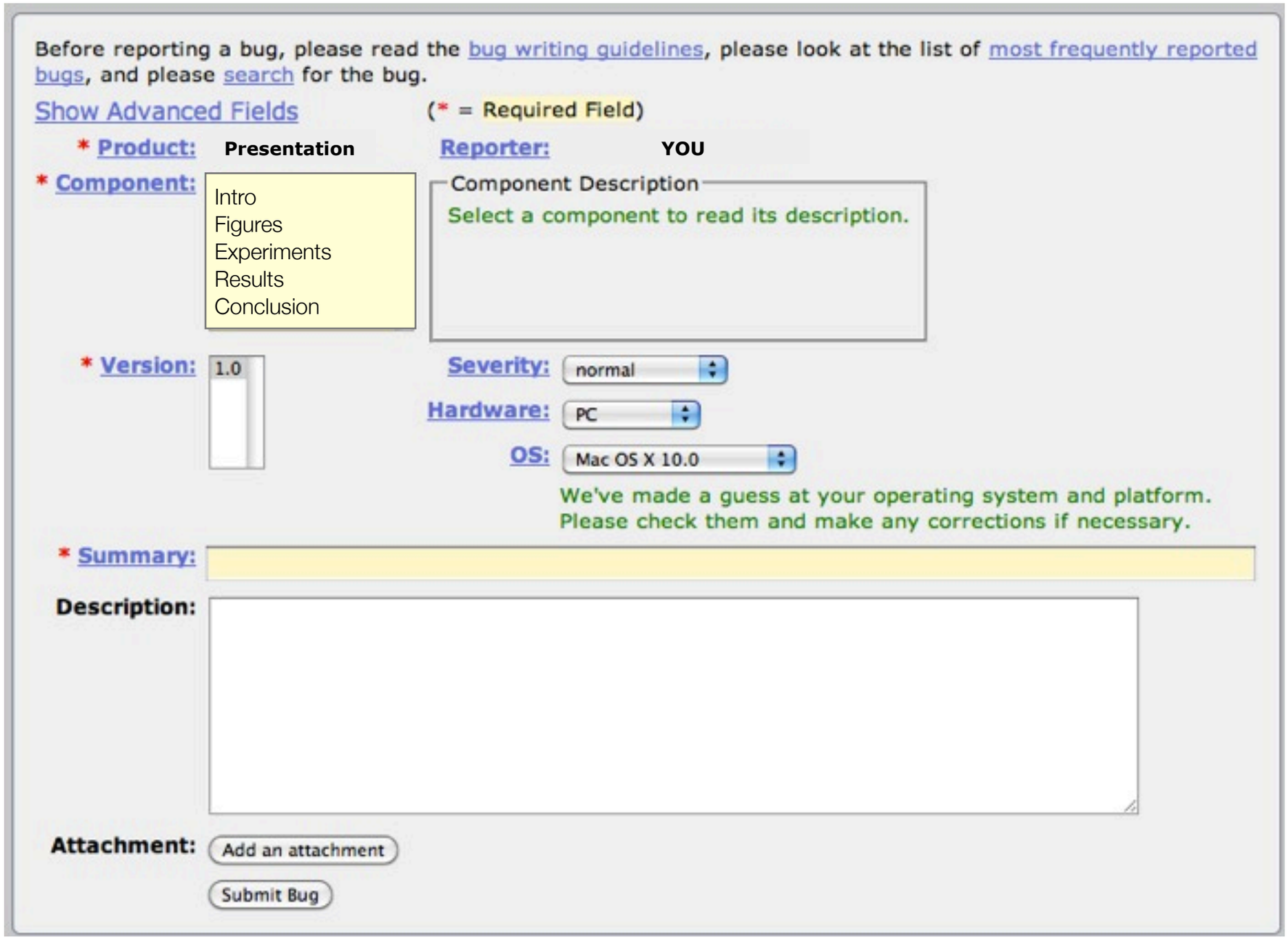
- Our approach perform **as well as, or better than**, the two other approaches in terms of recommendation accuracy

- Additional overhead of dynamic analysis **was not justified**

We are working on...



Questions ??


Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently reported bugs, and please search for the bug.
Show Advanced Fields
(* = Required Field)
* Product:
Presentation
Reporter:
YOU
* Component:
Intro
Figures
Experiments
Results
Conclusion
Component Description
Select a component to read its description.
* Version:
1.0
Severity:
normal
Hardware:
PC
OS:
Mac OS X 10.0
We've made a guess at your operating system and platform.
Please check them and make any corrections if necessary.
* Summary:
Description:
Attachment:
Add an attachment
Submit Bug