# Incorporating Structured Queries into Software Search

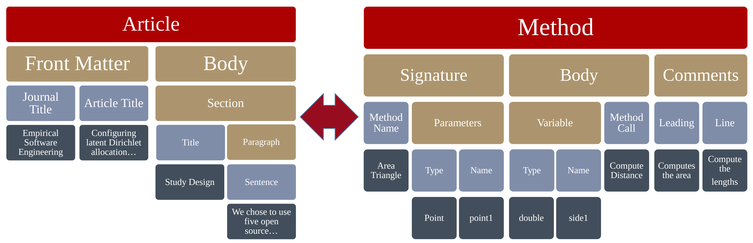

**Brian P. Eddy**
Department of Computer Science,
The University of Alabama

THE UNIVERSITY OF ALABAMA COMPUTER SCIENCE

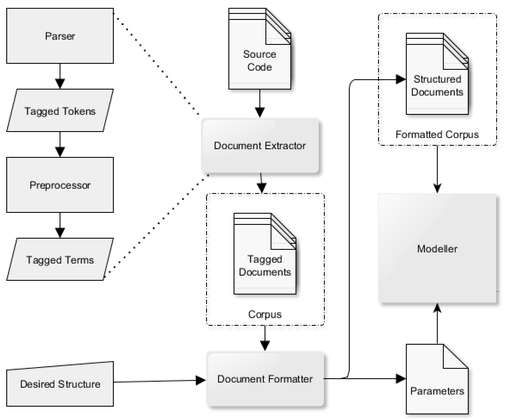
## Introduction

The use of text retrieval (TR) for software maintenance and program comprehension has been applied to search-oriented tasks such as feature location in source code. TR techniques allow for queries written in the developers' natural language (which feature requests and bug reports are typically written in) and have been shown to be more effective than keyword searches (which do not handle polysemy, synonymy, or non-exact term matches). A recent study showed that structural weighting of method names and method calls could improve the results of a TR technique in a feature location task. We introduce a framework to create structured queries for text retrieval on source code.

## Structured Retrieval




Documents such as scientific articles and source code contain inherent structure




Our framework involves converting source code into a set of TR models based on the source code's structure

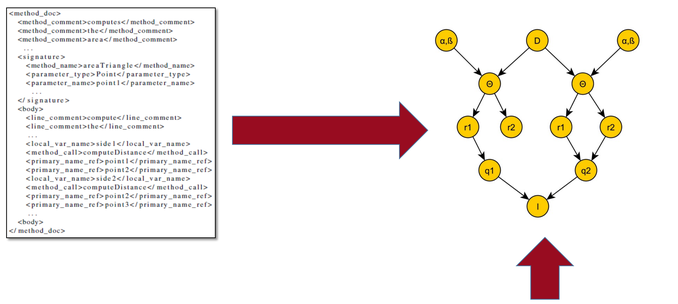
## Structured Method Documents

```
/*computes the area of a triangle using heron's formula*/
public static double areaTriangle(Point point1, Point point2, Point point3)
{
    //compute the lengths of the sides
    double side1 = computeDistance(point1, point2);
    double side2 = computeDistance(point2, point3);
    double side2 = computeDistance(point3, point1);

    //compute half of the perimeter
    double p = (side1 + side2 + side3) / 2;

    return Math.sqrt(p * (p - side1) * (p - side2) * (p - side3));
}
```

```
<method_doc>
  <method_comment>computes</method_comment>
  <method_comment>the</method_comment>
  <method_comment>area</method_comment>
  ...
  <signature>
    <method_name>areaTriangle</method_name>
    <parameter_type>Point</parameter_type>
    <parameter_name>point1</parameter_name>
    ...
  </signature>
  <body>
    <line_comment>compute</line_comment>
    <line_comment>the</line_comment>
    ...
    <local_var_name>side1</local_var_name>
    <method_call>computeDistance</method_call>
    <primary_name_ref>point1</primary_name_ref>
    <primary_name_ref>point2</primary_name_ref>
    <local_var_name>side2</local_var_name>
    <method_call>computeDistance</method_call>
    <primary_name_ref>point2</primary_name_ref>
    <primary_name_ref>point3</primary_name_ref>
    ...
  </body>
</method_doc>
```

Each method is converted into a structured representation of the original method's source (similar to the information in an AST, but allowing for TR models to be built)

## Content and Structure Queries




[signature]([method_name](area) (triangle))

Structured documents are converted into a retrieval model which can then be queried with CAS queries. The query in the above example looks for area in the method name as well as triangle anywhere in the method's signature. Queries may be longer (i.e., a feature request or bug report)

## Weighting Queries

- Weights the Results of Each Sub-query For Overall Similarity

  ( 2.0 [signature](area) 1.0 [body](area))

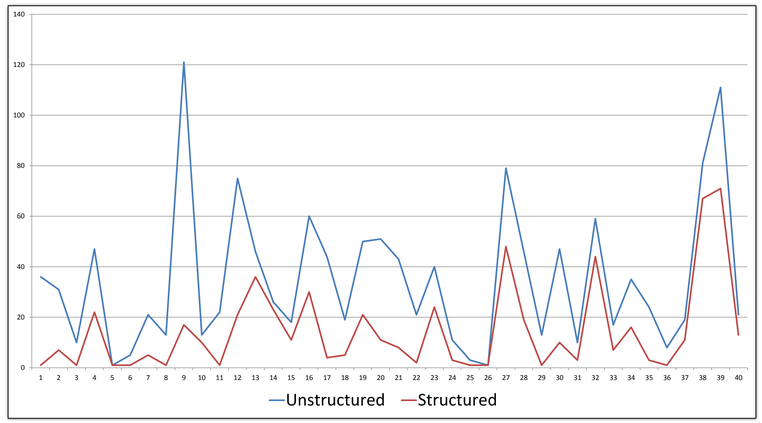
- An example of a Possible Weighting Function

0.67 log(likelihood([signature](area)))+ 0.33 log(likelihood([body](area))).

## Effects of Weighting

Results of 36 weighting schemes on four different subject systems
The common approach in feature location of using the rank of first relevant method is used here. Queries are obtained from the titles of feature requests

| System | Version | SLOC | CLOC | Methods |
|---|---|---|---|---|
| jEdit | 4.3 | 98460 | 42589 | 6550 |
| muCommander | 0.8.5 | 76649 | 68367 | 8811 |
| ArgoUML | 0.22 | 117649 | 104037 | 11348 |
| JabRef | 2.6 | 74350 | 25927 | 5323 |
| Total | | 367108 | 240920 | 32032 |




Ranks of the first relevant methods across 40 features ($p$<.01;$d$=1.27)

| | Unweighted | Leading | Signature | Body |
|---|---|---|---|---|
| # Features | 2 (tied) | 18 | 8 | 14 |

Component with highest performance when weighted independently