**“ALEXANDRU IOAN CUZA” UNIVERSITY IAȘI**

**FACULTY OF COMPUTER SCIENCE**

C:\Users\Claudiu\Desktop\logo_fii3.png

DISSERTATION PAPER

**Semantic Analysis of Source Code in**

**Object Oriented Programming.**

**A Case Study for *C#***

Scientific Coordinator Candidate

***Dr. Adrian Iftene Claudiu Epure***

**Session:** July, 2015

**“ALEXANDRU IOAN CUZA” UNIVERSITY IAȘI**

**FACULTY OF COMPUTER SCIENCE**

**Semantic Analysis of Source Code in**

**Object Oriented Programming.**

**A Case Study for *C#***

***Claudiu Epure***

**Session:** July, 2015

Scientific Coordinator

***Dr. Adrian Iftene***

**Declarație privind originalitatea și respectarea drepturilor de autor**

Prin prezenta declar că Lucrarea de disertație cu titlul *Semantic Analysis of Source Code in Object Oriented Programming. A Case Study for* C# este scrisă de mine și nu a mai fost prezentată niciodată la altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sau indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă,

sunt scrise între ghilimele și dețin referința precisă a sursei;

* reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința

precisă;

* codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate

cu respectarea drepturilor de autor și dețin referințe precise;

* rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași, *29 iunie 2015* Absolvent: Claudiu Epure

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Declarație de consimțământ**

Prin prezenta declar că Lucrarea de disertație cu titlul *Semantic Analysis of Source Code in Object Oriented Programming. A Case Study for* C#, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de disertație.

Iași, *29 iunie 2015* Absolvent: Claudiu Epure

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Contents

[1 Introduction 7](#_Toc421910833)

[1.1 Motivation 7](#_Toc421910834)

[1.2 Proposed Solution 7](#_Toc421910835)

[1.3 Keywords 8](#_Toc421910836)

[2 Concepts 9](#_Toc421910837)

[2.1 Introduction 9](#_Toc421910838)

[2.2 Definitions 9](#_Toc421910839)

[3 State of the Art 10](#_Toc421910840)

[3.1 Introduction 10](#_Toc421910841)

[3.2 Existing Systems for extracting structured data from source files 10](#_Toc421910842)

[3.3 Existing Systems for Information Retrieval based on Questions 11](#_Toc421910843)

[Bibliography 14](#_Toc421910844)

# Introduction

## Motivation

Computer programs have become the most frequently used tools in our modern society. Nowadays, they are present at large scale in industry in form of applications, platforms and services, covering multiple areas such as science and education, finance, commerce, etc.

Developing a software system is not an easy action. Instead, it is a complex process comprised of several phases, which are completed during a significant period of time. However, factors like customers high demand and market competition lead to acceleration of the process with negative impact on quality.

As complex software systems are built at a fast pace, they need to remain maintainable through time. For this reason, software quality must be at its highest level, yet in most cases, it decreases as the systems are getting bigger.

Testing the code is the way for assuring the required functionality from the perspective of the users. From the programmers point of view, the code needs to be clean and easy to extend or reuse. Design patterns, coding standards, static code analysis are software engineering methodologies serving such a purpose. But still, there are old systems, hard to refactor and production source code that is not implementing any engineering technique, which is very easy to break at any small try to redesign.

Another key aspect of software development is the use of version control systems in order to keep track of changes and make possible for teams to collaborate. They also provide a general view on the projects and backup service as well. Although they help to keep track of the phisical file changes over time, they do not provide a way of tracking the logical structure inside a project.

## Proposed Solution

None of the above mentioned techniques address the problem of retrieving meta information from the code, in a semantic manner. Large software projects involving thousands of source code files would be easier to understand, control and extend if they would be complemented by a solid information retrieval system.

The idea behind the proposed solution is to incrementally build knowledge datasets from plain source code.

The first step to achieve this is to create a graph oriented knowledge base from the entities present in code. This can be built on top of specific ontologies, using a convenient format like the *Resource Description Framework* (*RDF*)[[1]](#footnote-1). Having the knowledge base in place, it would be easy to query the system (e.g. *SPARQL*)[[2]](#footnote-2) about its interacting components and services. Going further, an answering mechanism could be applied for enabling natural language questions on the knowledge base.

This idea can be applied on every programming language, no matter the paradigm. However, proposed solution is focusing on the *Object Oriented Programming* paradigm and the programming language of choise in this case is *C#*. There main reason for choosing *C#* for this project is because the new *.NET Compiler Platform (“Roslyn”)[[3]](#footnote-3)* is just fit for this purpose, having a clean and easy to use *API*.

## Keywords

C#, static analysis, ontology, RDF, triple store, linked data, natural language, SPARQL

# Concepts

## Introduction

Regăsirea imaginilor (Image Retrieval) reprezintă obținerea imaginilor digitale dintr-o anumită colecție (bază de date). Procesul este format din mai multe etape, după cum urmează: introducerea interogării (utilizator), procesarea interogării, căutarea imaginilor și returnarea acestora (calculator). Diversificarea rezultatelor presupune, în plus, gruparea (clusterizarea) imaginilor în funcție de anumite criterii, astfel încât, pentru fiecare cluster va fi returnată doar o imagine reprezentativă. Scopul este eliminarea duplicatelor din prima mulțime de rezultate, orice cluster putând fi parcurs în adâncime, ulterior, la cerere.

## Definitions

*Regăsire de imagini (Image Retrieval)*:

* procesul de obținere a imaginilor digitale dintr-o bază de date (Weyand, 2005), (Deselaers, 2003).

*Regăsire de imagini bazată pe text (Text-Based Image Retrieval (TBIR))*:

* procesul de obținere a imaginilor atunci când este dată o descriere/specificație verbală a conținutului imaginii dorite (Nordbotten, 2008);

*Regăsire de imagini bazată pe conținut (Content-Based Image Retrieval (CBIR))*:

* orice tehnologie care, în principiu, ajută la organizarea arhivelor de imagini digitale, în funcție de conținutul lor vizual (Datta, Joshi, Li, & Wang, 2008);
* interogarea bazată pe conținut vizual; informația vizuală poate fi o imagine (interogare după exemplu), dar poate fi și o schiță a rezultatului dorit sau o descriere a proprietăților imaginii (distribuirea culorilor, forme conținute, etc) (Weyand, 2005).

*Clusterizare*:

* procesul de partiționare a unei mulțimi de date în grupuri omogene, pe baza unor criterii, cum ar fi similaritatea (Naik, 2010);

*Diversificare*:

* procesul de selectare și afișare a imaginilor reprezentative a clusterelor formate în urma clusterizării (mai sus definite) setului de rezultate obținut într-o interogare din cadrul regăsirii imaginilor.

# State of the Art

## Introduction

Some of the existing approaches that are based on similar ideas are mentioned below. They address singular or specific problems, so for the proposed system, the intent is to adapt, extend and combine some of the ideas, in order to achieve the goal.

## Existing Systems for extracting structured data from source files

***RDF Coder***

*RDF Coder* is a tool able to generate *RDF* models of code libraries, entirely written in *Java*. *RDF Coder* can be used to perform multi-level code inspection, create code dependency graphs and generate custom documentation. Currently is supported the *Java* language only at version *1.5*. This library can be used by Java programmers as a tool to deal with huge classpaths, to find relationships across classes and objects. *RDF Coder* can be also used to develop more complex analysis tools leveraging on the flexibility of the *RDF* related technologies.

***Fuzzy Ontology Framework***

*Fuzzy Ontology Framework* is a library that helps to integrate a fuzzy ontology with object-oriented programming (OOP) classes written in .NET. It is a hybrid integration, i.e. some OWL concepts can be mapped directly to OOP classes, yet most OWL concepts are derived just from OOP instance properties, with no direct mapping to a .NET class. Hence the OOP instance-OWL concept(s) mapping can evolve dynamically in the course of time.

The implementation currently supports FuzzyOWL2 ontologies together with FuzzyDL reasoner to infer affiliation of OOP instances to particular OWL concepts. It can be however easily modified to support any fuzzy ontology notation as well as any fuzzy reasoner.

***SCRO***

*SCRO* is created to support major Software Understanding tasks by explicitly representing the conceptual knowledge structure found in source code.

*SCRO* captures major concepts of object-oriented programs and helps understand the relations and dependencies among source code artifacts. Supported features include, encapsulation, inheritance (subclassing and subtyping), method overloading, method overriding, and method signature information.

***Source Code Plagiarism Detection Method***

*Ion Smeureanu, Bogdan Iancu -* *Source Code Plagiarism Detection Method Using Protégé Built Ontologies.* In this paper *the authors demonstrate how source code plagiarism could be detected with the help of an ontology which models software programs.*

## Existing Systems for Information Retrieval based on Questions

***Treo***

The main ideas behind this system are:

- Entity Recognition and Pivot Determination through Entity Search

- Query Syntactic Analysis: Partial Ordered Dependency Structure (*PODS*) Determination

- Spreading Activation using Semantic Relatedness

The algorithm first determines the *key entities* present in the natural language query. They can be potentially mapped to instances or classes in the *Linked Data Web*. The entity search engine receives the *key entities* and resolves *pivot entities* (URIs) in the Linked Data Web. The query is then analyzed and parsed to obtain *partial ordered dependency structure (PODS).* This is the reduced representation of the query targeted towards maximizing the matching probability between the structure of the terms present in the query and the subject, predicate, object structure of *RDF*.

The *spreading activation search* takes the URIs of the pivots and the PODS structure, and thus, starting from the pivot node, the algorithm navigates through neighboring nodes in the Linked Data Web computing the *semantic relatedness* between query terms and vocabulary terms in the node exploration process. The navigation process builds the answer of the query. The algorithm returns a set of ranked triple paths determined by the navigation from the pivot entity to the final resource representing the answer, ranked by the average of the relatedness scores over each triple path. Answers are displayed to users using a list of triple paths merged in a graph after a simple post-processing phase.

***TBSL***

The ideas of this system are:

- SPARQL Template from question

- mapping between NL expressions to the domain vocabulary

The *TBSL* approach to question answering over *RDF* data relies on a parse of the question to produce a *SPARQL* template that directly mirrors the internal structure of the question and that, in a second step, is instantiated by mapping the occurring natural language expressions to the domain vocabulary.

The user's input is a natural language question which is processed by a POS tagger. The result is the semantic representation of the natural language query, based on lexical entries that are created using a set of heuristics. In the next step, this is converted into a SPARQL query template which contains slots: missing elements of the query that have to be filled with URIs. The URIs are determined using sophisticated entity identification approaches, based on string similarity as well as on natural language patterns which are compiled from existing structured data in the Linked Data cloud and text documents.

***Squall2sparql***

*SQUALL* (Semantic Query and Update High-Level Language) is a *Controlled Natural Language (CNL)* for English that has full compliance with *Linked Open Data* (*LOD*), and covers nearly all features of *SPARQL* *1.1*, for both queries and updates. The advantage of *CNLs* is to provide a natural language syntax while retaining the precision and lack of ambiguity of formal languages like *SPARQL*.

The main drawback of *CNLs* is that users have to learn the language and its disambiguation rules. The system *squall2sparql2* is a Web application that supports the translation from *SQUALL* to *SPARQL*, as well as the direct querying of *SPARQL* endpoints, like *DBpedia*. Given a *SQUALL* sentence, the system first translates it into an intermediate logical representation using a *Montague* grammar. The intermediate representation is then translated into *SPARQL*, simply mapping logical constructs to combinations of *SPARQL* constructs. The produced query can then be sent to any *SPARQL* endpoint, and results returned.

***GFMed***

The main ideas of this system are:

- use of grammars created in *Gramatical Framework*

- use of medical datasets such as *SIDER*, *Diseasome* and *DrugBank*

*GFMed* system is focused on medical area questions and knowledge base.It consists mainly of a *Gramatical Framework* grammar for the application domain given by *SIDER*, *Diseasome* and *DrugBank* datasets. There is a minor preprocessing of questions and postprocessing of translation results, mainly in order to deal with structures involving numeric values, e.g. values for water solubility, or free text, like different names of foods. The system uses an abstract grammar and two concrete grammars. For each concrete grammar, lexicons derived from *SIDER*, *Diseasome* and *DrugBank* were generated. Both English and SPARQL grammars are based on the datatsets terminology.

# Bibliography

*WordNet*. (2012, Decembrie). Preluat pe Iunie 2013, de pe wordnet: http://wordnet.princeton.edu/

*/wiki/Levenshtein\_distance*. (2013, Mai). Preluat pe Iunie 2013, de pe rosettacode.org: http://rosettacode.org/wiki/Levenshtein\_distance

Atnafu, S., Chbeir, R., & Brunie, L. (2002, Februarie). *Efficient Content-Based and Metadata Retrieval in Image Database.* Preluat de pe jucs.org: http://www.jucs.org/jucs\_8\_6/efficient\_content\_based\_and/Atnafu\_S.pdf

Boucher, R. (2012, Noiembrie). *Windows Azure*. Preluat pe Iunie 2013, de pe msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/dd163896.aspx

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008, Aprilie). *Image retrieval: Ideas, influences, and trends of the new.* Preluat de pe infolab.standford.edu: http://infolab.stanford.edu/~wangz/project/imsearch/review/JOUR/datta.pdf

Deok-Hwan Kim, C.-W. C. (2005, Martie 9). *Relevance feedback using adaptive clustering for image similarity retrieval.* Preluat pe Iunie 2013, de pe http://islab.kaist.ac.kr/: http://islab.kaist.ac.kr/chungcw/relevance.pdf

Deselaers, T. (2003, Decembrie). *Features for Image Retrieval.* Preluat de pe thomas.deselaers.de: http://thomas.deselaers.de/publications/papers/deselaers\_diploma03.pdf

Finn, R. (1994). *Querying by image content.* Preluat de pe ibm.com: http://domino.research.ibm.com/comm/wwwr\_thinkresearch.nsf/pages/image396.html

Fischer, R. (1938). The statistical utilization of multiple measurements. *Annals of Eugenics 8*, 376–386.

Kai Song, Y. T. (2006). *Diversifying the Image Retrieval Results.* Preluat pe Iunie 2013, de pe http://idm.pku.edu.cn/: http://idm.pku.edu.cn/lunwen/2006\_2008626110631.pdf

Leetham, K. (2012, Septembrie). *Designing responsively*. Preluat pe Iulie 2013, de pe netmagazine.com: http://www.netmagazine.com/opinions/designing-responsively

Naik, A. (2010). Preluat de pe dataclusteringalgorithms: https://sites.google.com/site/dataclusteringalgorithms/

Nordbotten, J. (2008, Decembrie). *Multimedia Information Retrieval Systems.* Preluat de pe nordbotten.com: http://nordbotten.com/ADM/ADM\_book/MIRS-frame.htm

*Walkthrough: Creating a Web Site using Razor Syntax in Visual Studio*. (fără an). Preluat pe Iunie 2013, de pe msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/gg606533(v=vs.100).aspx

Weyand, T. (2005, Octombrie). *Combining Content-Based Image Retrieval with Textual Information Retrieval.* Preluat de pe tobw.net: http://tobw.net/work/cbir/weyand--research-project--05.pdf

1. *Resource Description Framework (RDF)*  is the standard model for data interchange on the Web and it is the *de facto* framework for storing data in a graph-oriented manner. [↑](#footnote-ref-1)
2. *SPARQL Protocol and RDF Query Languag (SPARQL)* is a semantic query language usedfor manipulating data stored in *RDF* format. [↑](#footnote-ref-2)
3. *.NET Compiler Platform*, having the codename "*Roslyn*", is an open-source project created by *Microsoft* which includes compiler and code analysis *API*s for *C#* and *Visual Basic.NET* programming languages. [↑](#footnote-ref-3)