CADV: A software visualization approach for code annotations distribution

–

Supplementary Material

Phyllipe Lima[a,b], Jorge Melegati[c], Everaldo Gomes[d], Nathalya Stefhany Pereira[a], Eduardo Guerra[c], Paulo Meirelles[d]

[a] *National Institute for Telecommunications – Inatel*
[b] *National Institute for Space Research – INPE*
[c] *Free University of Bolzano-Bolzen – UniBZ*
[d] *Federal University of ABC – CMCC-UFABC*

Abstract

This document contains the supplementary material for the manuscript: *CADV A software visualization approach for code annotations distribution*

1. Sample Projects with the AVisualizer

To further demonstrate the AVisualizer tool, we present the visualization for two open-source web applications. Figure 1 displays the **System View** for Geostore[1], an open-source Java enterprise application for storing, searching and retrieving data. It has 34K lines of code on 277 classes from which 210 (75%) have annotations. On Figure 2 we have Guj[2], a Brazilian Q&A forum about programming. It has 59K lines of code distributed among 437 classes of which 52 (12%) are annotated.

⋆Fully documented templates are available in the elsarticle package on CTAN.

Email addresses: phyllipe@inatel.br (Phyllipe Lima), jorge@jmelegati.com (Jorge Melegati), everaldogjr@gmail.com (Everaldo Gomes), nathalya.stefhany@gec.inatel.br (Nathalya Stefhany Pereira), eduardo.guerra@unibz.it (Eduardo Guerra), paulo.meirelles@ufabc.edu.br (Paulo Meirelles)

[1] https://github.com/geosolutions-it/geostore

[2] https://github.com/caelum/guj.com.br

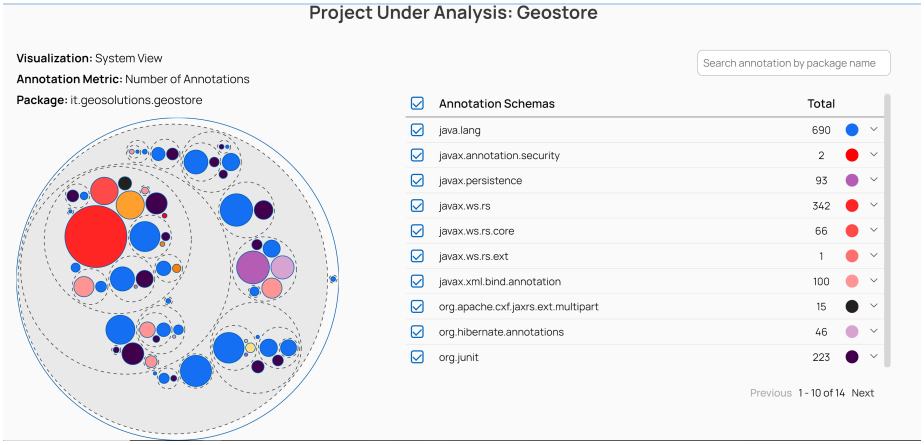

Figure 1: System View for Geostore

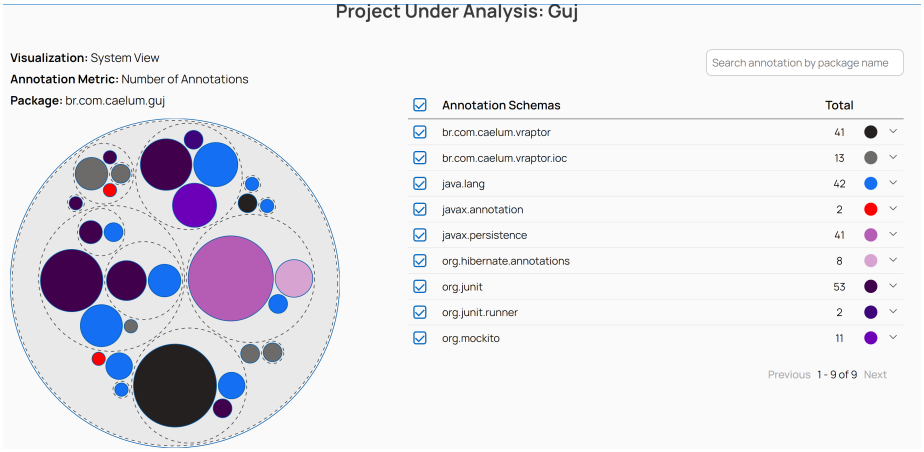

Figure 2: System View for Guj

Table 1: Projects summary

Project	Total Classes	Annotated Classes	Total LOC	LOC (Annotated Class)
Guj	437	52	59811	2479
Geostore	277	210	34701	4494

2. Coding

As we mentioned in Section 6 in the paper, we performed a coding of the interview and open-ended questions from the questionnaire. The goal was to

perform a qualitative analysis to support the results obtained from the quantitative analysis. We obtained code that supported the ***perceived ease of use*** and ***perceived usefulness***. Furthermore, for each of these we extracted codes that sustained the strengths and weakness. We now present four example of how we performed the coding.

2.1. Perceived Usefulness

An interviewee stated [translated from Portuguese]: " *[...] it is very simple to detect potentially misplaced code annotations. If I spot a pink circle in a package with only orange circles, I would say something is wrong. Why is a javax.persistence code annotation alone in a package with only org.springframework code annotations?*"

We understood this was perceived as something useful to the user, in other words, a **P**erceived **U**sefulness **S**trength that we labeled PUS. To further label this code, it was related to the color of schemas helping identifying potential misplaced code annotatios, thus we obtained

- PUS-6: **Color strategy helps identify potentially misplaced or inconsistent annotation**

During the interview and questionnaire, every other answer that mentioned that, somehow the colors were helping identifying misplaced annotations we checked this code.

On the other hand, we also identified comments that was a weakness in something that tool was providing. For instance, when we were discussing how to find "large annotations" or "annotations used excessively" one interviewee stated: "*I would check the size of the circle, but that does not mean much*". Another interviewee stated: "[...] by the size of the circle. But the tool is not adequate for that. It depends on other factors and design decisions. It may be complete adequate for some classes to have more annotations than other".

We understood these quotes as something that may harm the usefulness of the tool. So we labeled as in the category PUW (**P**erceived **U**sefulness **W**eakness). Thus, these and other quotes generated the code:

- PUW-1: **I can find large annotations, but this is not very useful isolated.**

2.2. Perceived Ease

To obtain codes to support the "perceived ease of use" we identified, from the interview and open-questions comments that either made the tool easy to use (strengths) or difficult to use (weakness). For instance, when discussing if the circle packing approach was suitable to see the hiearchy, one interviewee stated: "*Yes, in theory it is very simple. The larger outlined circles are parents of the smaller outlined outlined circles contained in them*".

This was measure as something that is "easy" for the user. So we labeled as a PES ("**P**erceived **E**ase of Use **S**trenghts").

The code we generated for this quote was:

- PES-3: **Easy to understand package hierarchy**

We also noticed comments that highlighted that some points made the tool difficult to use. For instance, when discussing the change of metric between views, one interviewee stated: "I feel it is a somewhat difficult to do this change of context in my mind [change Package View to Class View]. I know the header is there informing that we changed views and metric, its not a big stress, but it could be easier".

We classified this as: PEW ("**P**erceived **E**ase of Use **W**eakness"), and from this quote we generated the code:

The code we generated for this quote was:

- PEW-3: **Confusion when switching metrics between views**

3. Questionnaire

Table 2 contains the ten (10) questions related to the program comprehension tasks we asked the students. Table 3 contains the statements used to measure the **ease of use**. Table 4 contain the statements we used to measure the usefulness.

Table 2: Program Comprehension Questionnaire.

ID	**Question**	Goal
Q1	What annotation schema is located in a single package?	G1
Q2	What annotation schema is located in more packages ? (more distributed)	G1
Q3	Which annotation schema contains the largest amount of annotations being used?	G1
Q4	What class contains the highest number of javax.persistence annotations?	G2/G4
Q5	What package contains classes being mapped to databases (usage of javax.persistence)?	G3/G4
Q6	What package is mostly concerned with web controllers (usage of org.springframework.web.bind.annotation)?	G2
Q7	How many packages contains unit testing class(es)?	G2
Q8	What javax.persistence annotation has the highest LOCAD (lines of code per annotation) value?	G2
Q9	In the class br.inpe.climaespacial.tsi.entity.model.TsiHDU (fully-qualified name), what code element has more annotations configuring it?	G2
Q10	What annotation from the org.springframework.web.bind.annotation schema contains more attributes/arguments (Annotation Attribute metric - AA)?	G3

Table 3: Perceived Ease of Use - Statements.

ID	Statement
SE1	I can easily identify java packages with different responsibilities using the AVisualizer tool
SE2	I can easily see how code annotations are distributed in the system under analysis using the AVisualizer tool
SE3	Learning how to use the AVisualizer was easy to me
SE4	I can easily see how many annotation schemas are being used inside a java class using the AVisualizer
SE5	I can easily see how many annotation schemas are being used inside a java package using the AVisualizer
SE6	I can easily identify what java package I am currently inspecting using the AVisualizer
SE7	I can easily identify the class I'm inspecting in the AVisualizer tool
SE8	I can easily navigate to and from the packages and classes being analyzed with the AVisualizer tool

Table 4: Perceived Usefulness - Statements

ID	Statement
	Using the AVisualizer
SU1	I would easily detect all annotation schemas that is being used in a java system
SU2	I can easily spot large annotations
SU3	I can easily spot potential misplaced code annotations
SU4	I can easily identify classes highly coupled to annotation schemas
	Code Inspection
SU1	I would easily detect all annotation schemas that is being used in a java system
SU2	I can easily spot large annotations
SU3	I can easily spot potential misplaced code annotations
SU4	I can easily identify classes highly coupled to annotation schemas