HAnS: Feature Visualisation - Experiment

David Stechow and Philipp Kusmierz
Supervisor: Prof. Dr. Thorsten Berger, Kevin Hermann

RUHR
UNIVERSITÄT
BOCHUM

RUB

Procedure of the Experiment

1. Background information on features and feature annotations
2. Introduction to the HAnS-Plugin
3. Introduction to the HAnS-Viz-Plugin
4. Experiment

Motivation

- Developers often need to navigate the system to locate and understand the implementation of certain features.

- Challenges:
  - Recognizing where a feature is implemented
  - Understanding a features correlation with other components or features

- Tackling these challenges is one of the most time-consuming tasks of a developer [1].

What is a Feature?

- „Feature“ commonly refers to a specific and encapsulated piece of logic that provides a specific value or capability to the overall software system.

- A Feature could be e.g.
  - Changing themes for your web browser (web browser)
  - Git integration within an IDE (IDE)
  - Sorting E-Mails by a specific filter (E-Mail program) etc.

How can we find Features in our Project?

The location of a feature is usually found by:

- Know-how (experienced developers may already know the location because they have worked on it)
- Read the documentation (if there is any)
- Manually scanning the codebase and locating the feature by searching for related classes, methods, documents, etc.

But when there is no documentation and no one to ask, the search for a feature can be a very **time-consuming** process [1].

Is there a way to overcome this problem?

Feature Annotations

One way to trace features, is by documenting them and labeling their corresponding software assets with feature annotations.

But what should feature annotations offer?

They should be[2]:

- Programming language independent (Java, C++, JavaScript, …)
- Easy-to-learn
- Intuitive
- Robust
- Useable with minimal developer effort

HAnS Plugin

- **H**elping **An**notate **S**oftware
- Open Source plugin for JetBrains IDEs.
- Provides lightweight feature annotations.
- Allows developers to structure their code into features by mapping assets to specific features.

HAnS – Feature Annotation

- Currently HAnS supports the following feature annotation types:
  - Code
    - Maps code regions
  - File
    - Maps a whole file
  - Folder
    - Maps a folder and its contents

```java
public DataOfSquare(int color) {
    C.add(Color.darkGray);// &line[Tile::Snake]
    C.add(Color.BLUE);    // &line[Food]
    C.add(Color.white);   // &line[Playing_Area]
    square = new SquarePanel(C.get(color));
}

// &begin[Update]
public void lightMeUp(SquareToLightUp c) {
    square.ChangeColor(C.get(c.ordinal()));
}
// &end[Update]
```

Code-Annotation

.feature-to-file

```
1  Direction.java
2  Controls, Move
3
```

File-Annotation

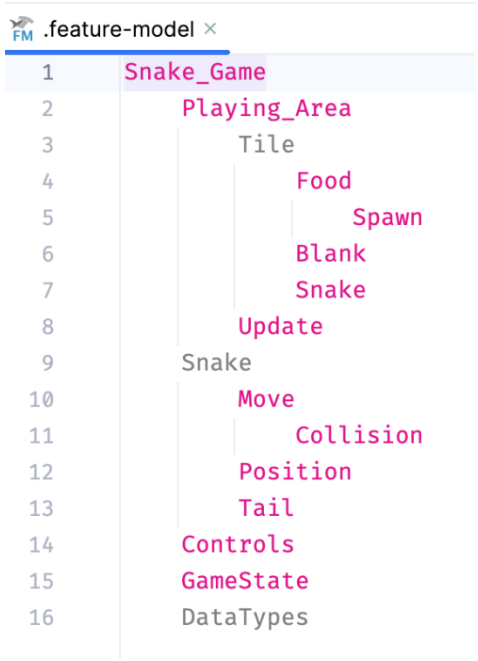
.feature-to-folder

```
1  Snake_Game::Snake, Controls
2
```

Folder-Annotation

HAnS – Feature Model

- Features annotated with HAnS are documented within a .feature-model file.
- This .feature-model file holds information of all defined features and their hierarchical relation.



HAnS Plugin – Demo Video

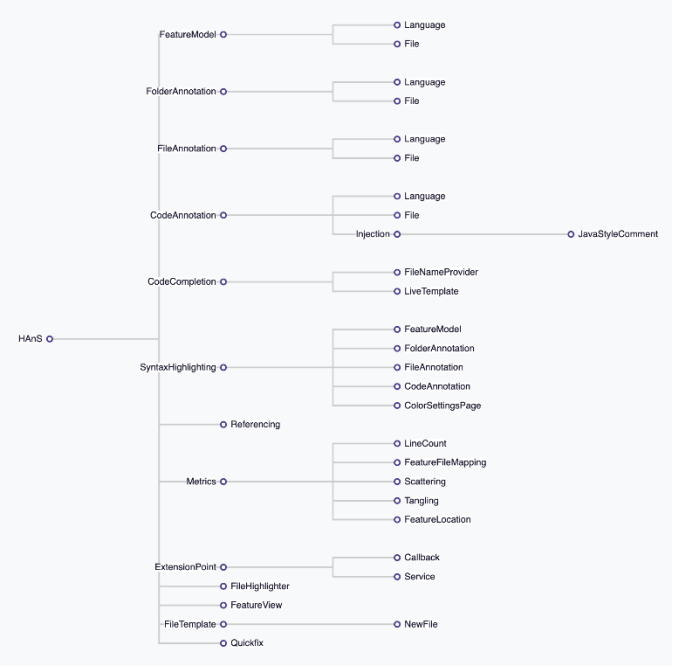
https://www.youtube.com/watch?v=cx_-ZshHLgA

HAnS-Viz

- Is a plugin developed for HAnS
- Visualizes metrics
- The metrics are provided by HAnS

HAnS-Viz – Tree View

- Is the first view
- Provides an hierarchical overview of the .feature-model
- The nodes can be expanded or collapsed



HAnS-Viz – Tree Map View

The Tree Map view visualizes:

- First level features
- Total line count of a feature

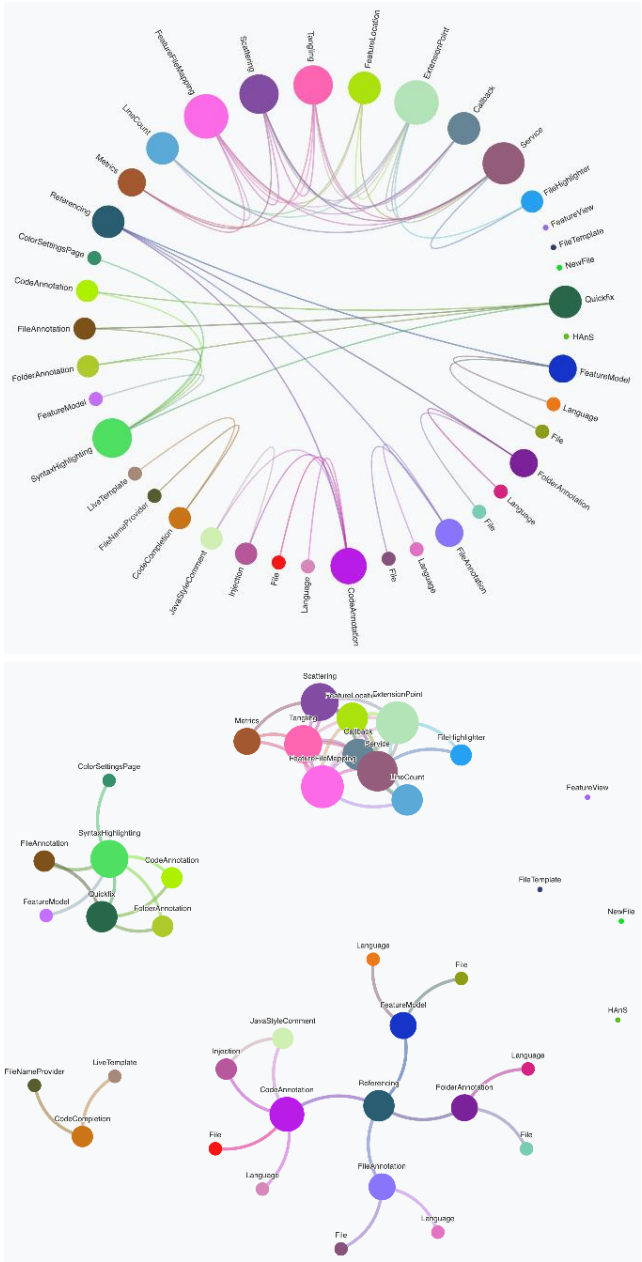
It is possible to:

- Change focus to a given feature by clicking on it.
- Show the total line-count of a feature and its underlying features

HAnS-Viz – Tangling View

The Tangling view is a graphical representation of the tangled features within the project

- can be displayed in either a circular or non-circular view.
- The non-circular view uses a force-directed layout algorithm to place the nodes in such a way that it is easy to spot isolated clusters of features.
- Hovering over a feature will highlight its tangled features and show the tangling- and scattering-degree.
- The colors are derived from a hash so that they stay consistent

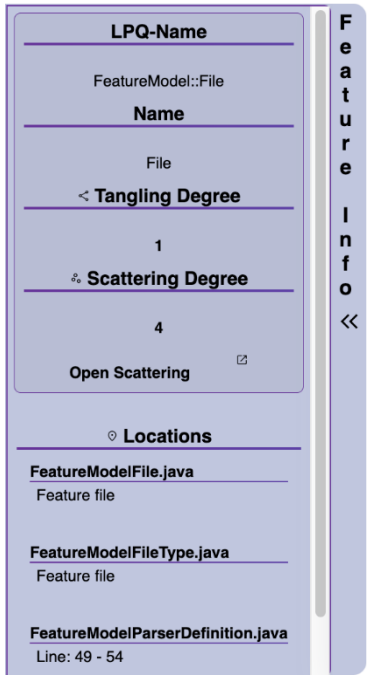


HAnS-Viz – Feature Info Window

A click on a feature within any view will display its information within the Feature Info Window (FIWi)

The FIWi:

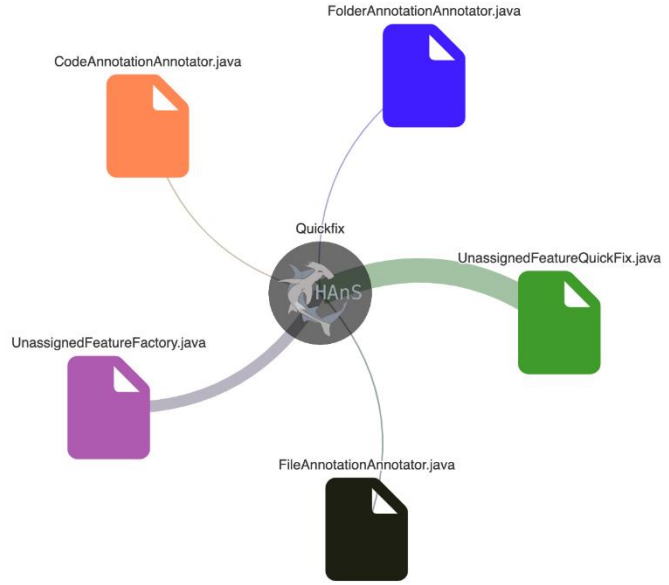
- Is a compact widget which display feature based information
- provides a list of all feature-locations.
- Each location is clickable and will navigate the user to its corresponding location within the IDE‘s editor.



HAnS-Viz – Scattering View

The Scattering View provides an overview of all files which are associated with the given feature.

- The width of the edges is relative to feature coverage of each file
- The file-symbols are also clickable and will navigate the user to the corresponding file within the IDE's editor
- A double click on a feature within any view (except the Tree View) will open the Scattering View.



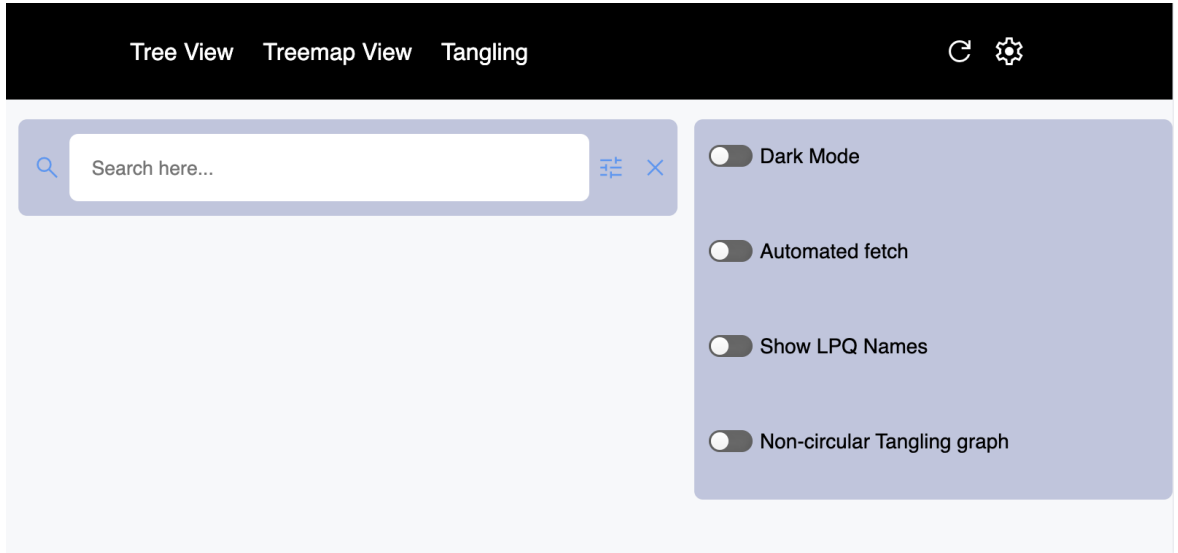
HAnS-Viz – Usability Features

HAnS-Viz also offers some usability features such as

- **a searchbar to highlight features**
- a settings window

The search filter currently supports the following options, which can individually be toggled:

- Incremental-search
- RegEx
- Exact-match
- Case-sensitive



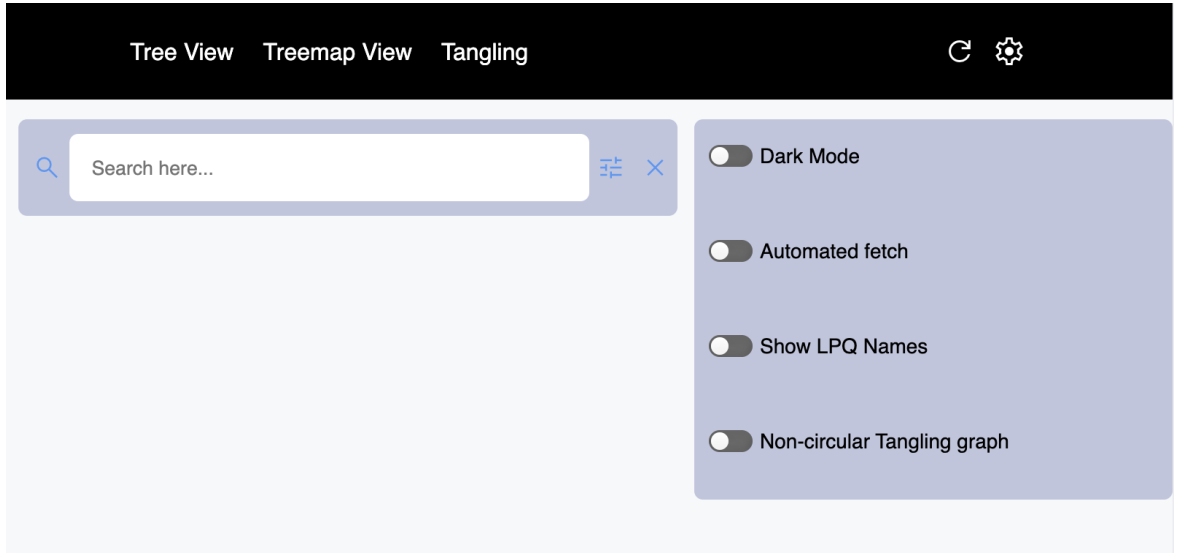
HAnS-Viz – Usability Features

HAnS-Viz also offers some usability features such as

- a searchbar to highlight features
- **a settings window**

HAnS-Viz currently supports the following settings:

- Dark Mode
- Automated fetch to update the views
- Show or hide the LPQ Names of features
- Change the layout of the tangling view



Experiment

- Follow the link or scan the QR code: https://forms.gle/kRT1cZFF9XyuDAPi6
- You do **not** need to log in to a google account to participate
- Follow the instructions to install both plugins and to open the sample project

The questionnaire consists of multiple tasks that will encourage you to use HAnS-Viz.

After each task you have to answer a few questions.

You will always have the opportunity to leave feedback after each task

References

[1] J. Krüger, T. Berger, and T. Leich, ‘Features and how to find them: a survey of manual feature location’, Software Engineering for Variability Intensive Systems, pp. 153–172, 2019.

[2] T. Schwarz, W. Mahmood, and T. Berger, A common notation and tool support for embedded feature annotations. 2020.