THE ACADEMIC COLLEGE of TEL AVIV-YAFFO

Workshop: Software Engineering

Project Proposal Document

Project Information

Project Name:
OOP Analyzer

Student Names:
1. Yonatan Brooker

2. Yoav Sraya

Project Track:
Technological

The Background:

Object-oriented programming (OOP) is a type of programming based on the idea of objects that can contain both data and code. Objects have fields (also called attributes or properties) that contain data and procedures (known as methods) that contain code. With OOP, computer programs are built by using objects that interact with each other.

OOP became popular in the 1980s with the rise of C++, and it became one of the most widely used programming paradigms with the emergence of languages like Java and C# in the mid-1990s and early 2000s. Despite its popularity, OOP can be challenging for programmers to understand due to its abstract concepts, such as class hierarchies, inheritance, and polymorphism, especially in large projects.

To make the code more readable and realistic, programmers began to see how OOP could be represented as a network of strings that connect all the objects into one logic diagram. In the late 1990s, this became the Unified Modeling Language (UML). The UML allows code architects to design a program more efficiently and pass the concept to programmers. However, UML is intended for design and not code maintainability. Usually, the UML becomes very complicated and not intuitive.

So, What is the problem?

In modern software development, OOP projects can become large and complex, creating significant challenges for experienced developers and newcomers. As projects grow in size and complexity, it becomes increasingly difficult to maintain a clear understanding of the underlying design principles, class relationships, and design patterns. This complexity becomes a barrier for new programmers trying to enter the project, and for existing developers who need to navigate and extend the codebase. Consequently, the lack of clarity and structure in OOP projects hinders productivity, increases the likelihood of errors, and impedes the overall maintainability and scalability of the software system.

And what is the solution?

The idea is an object-oriented programming mapping and analysis tool for c# projects that helps developers to visualize their codebase. This tool will display the project's architecture, including the relationships between classes, modules, and components in an interactive map. Developers can drill down to specific files, classes, methods, and dependencies to understand the codebase's structure and organization.

The map will have a smart search where you can filter and search for specific objects, and the map will take you directly to the place where the object in the map

The tool also uses an artificial intelligence (AI) engine via API to identify areas for improvement and provide suggestions for better Structuring, architecting, and make sure the project is holding in most important rules of OOP: maintainability, and encapsulation. It can recommend design pattern refactoring.

the code will be pulled live from IDE/GitHub and present changes on time.

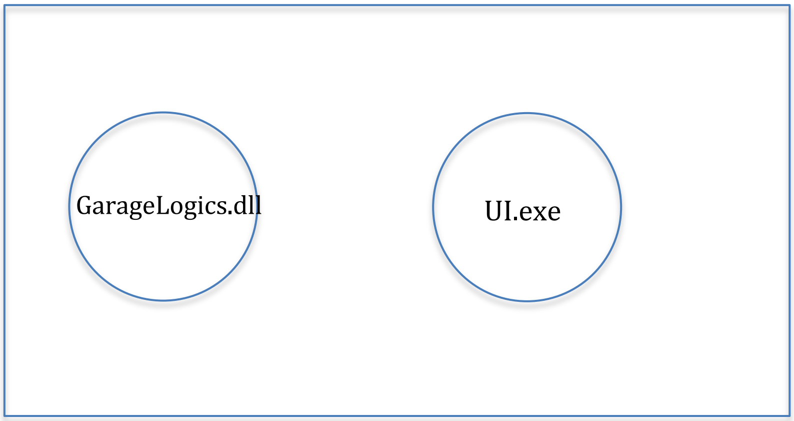
Example will provide in the end of the Proposal with example of rules that the map will have to keep order and clean view

Example:

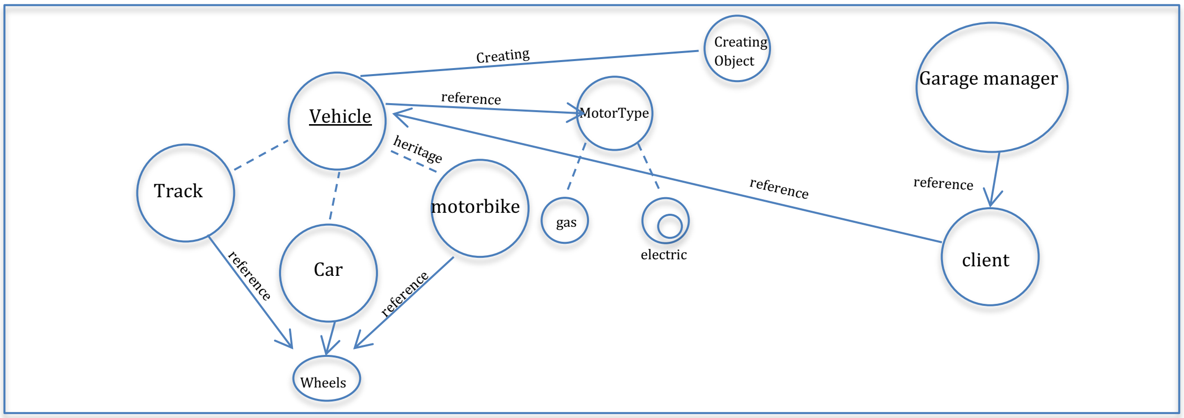
Program of garage manager.

The design will change, its only example, the map design is one of the important thing in our App

First, the map will present the project files:

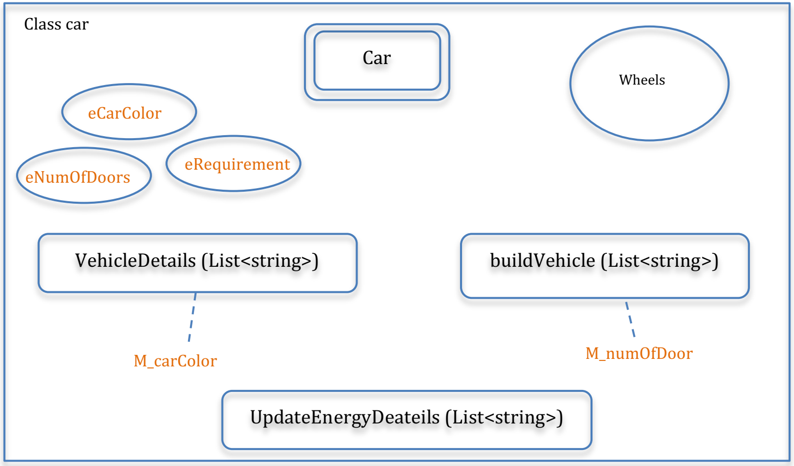


Second, when the user clicks on one of the circles or zoom in with his mouse the map will zoom in into the file for example: GrageLogics.dll



** The shape size will change in parallel to the class size
** The Diagram will always keep order by attribution (successor always below his ancestor)
** the circles can be move by the user

Third, when the user click on one of the class for example: Car


Class car
Car
Wheels
eCarColor
eNumOfDoors
eRequirement
VehicleDetails (List<string>)
buildVehicle (List<string>)
M_carColor
M_numOfDoor
UpdateEnergyDeateils (List<string>)

** constructor will always be on the Top and remark differently

** All types group by there type (for example : all the Enum together)

** if there is nested class inside the user can zoom in to see the class inside

We gave the parsing to the Chat GTP AI engine and that what he have to say:

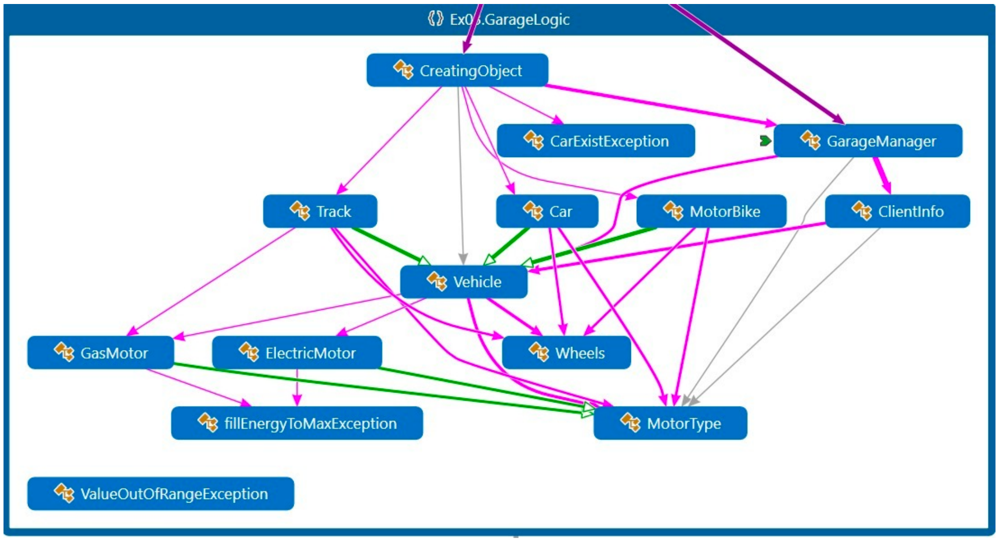
Others do it, but..

There are various software available for mapping OOP code platforms, such as SourceTrail, SourceTrace, and Understand, that help developers visualize codebases and structures. However, none of them provide the option to zoom in on the project types and classes. Instead, they give a 2D diagram that is usually not interactive.

In addition, there are analysis tools like SonarQube, ReSharper, and CodeClimate that offer static code analysis capabilities to identify code smells, vulnerabilities, and maintainability issues. These tools extract code reports or mark the code itself (if the tool is inside the IDE). Moreover, there are AI engines that developers can use to ask for code improvements.

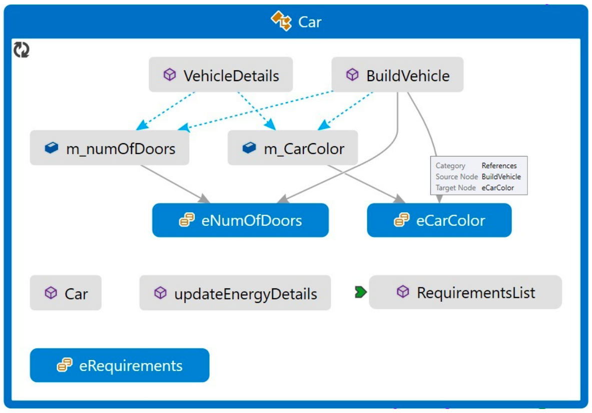
We aim to create a product that provides a more interactive experience around the code structures and architecture and offers specific suggestions to improve the code structure.

Lets give an example for a bad – complicated map:



As you can see the map doesn't order the objects, most of the item is placed randomly. It's hard to understand the connection between the object by just looking at the map.

Inside the class it's hard to see the differences between methods, data members, Enums and more..



Who is this tool for?

The App is intended for OOP programmers and Companies that want to make orders in their OOP projects and let new workers enter the project much more quickly. This tool is for those who don't want to lose it when they are trying every time to understand how a specific object was created and from whom it is heritage. It's for programmers who just want to be in control of their code structure and growing projects. To be sure, there are rules for object-oriented concepts and design patterns.

What is the program Flow:

1. Import the project:
   the user will have to connect his code to the app using GitHub.
   to user need to login to his GitHub user and select a public repository he wishes to analyze.

2. Parse the c# project:
   the app will parse the c# code into textual data about the classes, files, and methods.
   With the data we can create the map and connect with the AI engine.

3. Create code Mapping:
   the app takes the parsing data from the user code and create interactive code map using graph liberty in JS.

4. Analyze the parsing:
   the app will provide data to an AI engine to look for structural mistakes/ suggest improvements (or everything the AI can provide – it will research during the work)

5. presenting the interactive map and the AI suggestion.

6. On demand:
   keep refreshing the map to be almost live with the code changes using the GitHub Rest API.

What is the application design?

The app is a web application and will be separated to back and front:

1. Front-End: For the front-end, we will React framework to manage the application's UI and integrate with one of the graph liberties in js. integration with openAI API to analyze the parsing data and upload suggestions to the user.

2. Back end: For the back end, we will use server-side technology to handle the communication with GitHub API in node JS. And parse the .cs file with the c# parsers “Roslyn”.

3. Deployment: For deployment, we are using the AWS platform to host our application and make it accessible over the internet.