GoCity: Code City for Go

Rodrigo Brito, Aline Brito, Gleison Brito, Marco Tulio Valente
ASERG Group, Department of Computer Science (DCC), Federal University of Minas Gerais, Brazil
rodrigobrito@ufmg.br, {alinebrito, gleison.brito, mtov}@dcc.ufmg.br

Abstract—Go is a statically typed and compiled language, which has been widely used to develop robust and popular projects. As other systems, these projects change over time. Developers commonly modify the source code to improve the quality or to implement new features. In this context, they use tools and approaches to support software maintenance tasks. However, there is a lack of tools to support Go developers in this process. To address these challenges, we introduce GoCity, a web-based implementation of the CodeCity program visualization metaphor for Go. The tool extracts source code metrics to create a software visualization in an automated way. We also report usage scenarios of GoCity involving three popular Go projects. Finally, we report the feedback of 12 developers about the GoCity of their projects.

Index Terms—Software Visualization, Software Evolution, Mining Software Repositories

I. INTRODUCTION

Go is a statically typed and compiled programming language created by Google in 2007, and public available since 2009. The language was created to supply demands of large-scale and complex systems. Recently, it was pointed as one of the most loved languages. Besides that, there are approximately 20K Go projects hosted on GitHub. Among these Go projects, there are popular and large software systems like DOCKER (a platform to software containerization), and KUBERNETES (a system to manage containers).

As other software systems, changes often occur in Go projects to include new features, improve the source code quality, and fix bug [1], [2]. As a consequence, an important part of a software evolution encompasses maintenance and code understanding tasks [3]. In this context, the literature presents several approaches and tools to help developers when evolving software systems [4]–[8]. Specifically, these approaches focus in important questions that often arise during software analysis and evolution: (a) Is my project well modularized? (b) Are there design problems in my system? (c) What are the most complex and large elements in my project?

However, we still have a limited number of techniques and tools to support Go developers in these tasks. To address these problems, we propose GoCity, a web tool to visualize Go projects as 3D cities. Our tool is based in the code city metaphor, a popular approach to support software analysis and evolution proposed by Wettel and Lanza [4]. GoCity analyses projects hosted on GitHub in an automatic way, providing an on-the-fly visualization based in metrics from the source code. For example, a struct in a Go project represents a building in the city, and the number of methods in the struct defines the building height. Our ultimate goal is to help Go developers in maintenance tasks, showing relevant information about code design and architecture.

Although it is still a research prototype tool, GoCity was widely used and explored by Go developers. For example, the tool had 23K unique users from different countries on November, 2018. Besides that, the project has around 1.3K stars on GitHub, in the same time frame. As a complementary evaluation, we sent emails to 60 owners of popular Go projects, asking for feedback about GoCity. More specifically, we first created the city visualization of these projects in our tool. Then, we sent an email with the visualization link to the developers, asking for their insights about code design, architecture, and maintenance tasks. We received 12 answers, which represents a response rate of 20%.

Structure of the paper. Section II describes the city metaphor to represent the source code. Section III shows the architecture of GoCity. Section IV presents usage scenarios with three real-world Go projects. Section V details the users feedback about our tool. Section VI presents related work. Finally, we conclude the paper in Section VII.

II. CODE CITY FOR GO

GoCity is based in CodeCity [4], [5], which is a popular metaphor to represent software systems as cities. However, CodeCity was initially implemented only to C++, Java, and Smalltalk. In the original implementation, the source code elements are represented as urban structures, and code metrics are used to characterize them. With GoCity, we adapted the metaphor to support Go features. For example, there are no classes in Go, but the language includes a similar structure named struct, which is a type that can contain attributes and methods. Besides that, Go allows declaration of methods and attributes outside a struct. In this way, GoCity represents a file as a building, and the structs are sub-building positioned on building’s top.

Figure 1 shows an overview of the proposed visualization. Similar to CodeCity, the building’s width and height represent the number of attributes and methods in a struct, respectively. Buildings representing files are defined by the number of methods and attributes declared outside a struct. The color refers to the element category; particularly, structs are denoted by blue color, the files are represented by gray color, and the
directories are red. In addition, the color intensity indicates the number of lines (i.e., darker buildings are large, in terms of lines of code).

![Code City for Go](image)

Fig. 1. Code City for Go

III. ARCHITECTURE

As presented in Figure 2, the high level architecture of GoCity includes two modules: processing and front-end.

![GoCity Architecture](image)

Fig. 2. GoCity Architecture

**Processing Module.** This module produces an instance of the project as a city metaphor. First, in Git Service, the project is fetched from its repository. Then, Parser Service extracts the metadata (e.g., number of methods, fields, and lines). The hierarchy of the city and the location of buildings are defined by Position Service, aiming a better use of space. This module also includes a cache service, which is based on a Least Recently Used (LRU) algorithm. As consequence, our tool stores temporarily the processed project, avoiding unnecessary requests and performing a better CPU usage.

**Front-end Module.** This module represents the web-based interface of GoCity. Its receives as input the URL of the analyzed project. Validation Service reports the errors in the data input, as well as, server errors (e.g., connection failures). 3D Processor provides the output, i.e., a visualization of the Go source code as a 3D city, which is implemented using Babylon.js[^5] and React.js[^6]. In addition, the interface is interactive, e.g., Navigation Service provides features to rotate and zoom the image.

[^5]: https://www.babylonjs.com
[^6]: https://reactjs.org

IV. USAGE EXAMPLES

In this section, we present two examples of usage of GoCity. In the first example, we inspected a file from GO-HUGOIO/HUGO[^7] (a framework to construct websites). We locate the file on GitHub, and we also describe the code metrics. Then, in a second example, we analyze the code design of two popular Go projects, LABSTACK/ECHO[^8] and GORILLA/MUX[^9], which are used to build web applications.

A. Inspecting a Source Code File

GoCity provides an option to visualize source code metrics of a given code element. When hovering the mouse over the element, a window is displayed with the number of lines, number of methods, and number of attributes of the building structure. For example, in Figure 3, we inspect the `struct Page` of GO-HUGOIO/HUGO project. This `struct` contains 1,542 lines, 96 methods, and 62 attributes. In addition, our tool provides a button to retrieve the code on GitHub. The functionality is provided for directories, files and structs. Figure 4 shows an example, when the `struct Page` is opened on GitHub, highlighting the file declaration.

[^7]: https://github.com/gohugoio/hugo
[^8]: https://github.com/labstack/echo
[^9]: https://github.com/gorilla/mux
B. High-level Design Visualization

Figure 5 shows the code city visualization of LABSTACK/ECHO, a popular project to create web applications in Go. The project has more than 12K GitHub stars and 53 source code files. The largest gray structure represents the main module, which contains the application core. The structures in the bottom right corner refer to a collection of utilities and files from router module. Finally, the tallest building in blue (on top) represents the context system, a key component of the project, which controls router variables, parameters, and response values.

Figure 6 shows a second example, the city of GORILLA/MUX, a popular web router for Go. The project has 14 source code files and more than 7K GitHub stars. The city has a single district, since the project only contains one folder. The large blue building refers to the core application, including methods to perform router operations. The largest buildings (gray) and the highest building (black) represent test files. As we can see, the project groups the tests in few and large files. For example, the black building represents a test file with approximately 2K lines of code.

V. USER FEEDBACK

We also surveyed Go developers to collect their feedback about our tool. This section details the survey design and the survey answers.

A. Survey Design

Selection of the Go projects. First, we selected the top-100 most popular Go projects on GitHub (on November, 2018), ordered by number of stars, a relevant metric to filter popular repositories [9]. This list does not include projects that are forks or organizations, as well as projects whose owners do not have a public email. Next, we manually inspected the projects to guarantee they are software systems. For example, we removed AVELINO/AWESOME-GO because this repository just contains a list of popular Go systems. These filtering steps resulted in 64 projects. Then, we removed projects maintained by the same owner. In other words, we only selected one repository per owner to avoid sending multiple emails to the same person. The final list consists in 60 systems, including well-known Go projects, such as ASTAXIE/BEEGO (a framework to Web development), and JUNEGUNN/FZF (a fuzzy finder to command line).

Contacting the developers. We sent an email to each owner from the 60 selected Go projects, asking for their feedback about GoCity. Figure 7 shows the email template. In the emails, we added a link to the code city visualization provided by our tool. Then, we asked two questions. With the first question, our interest is to investigate whether GoCity helps to reveal information about the system design. With the second question, we intend to reveal the use of GoCity on software maintenance process. We received 12 answers, corresponding a response ratio of 20%.

Analyzing the answers. We use thematic analysis to interpret the developers responses [10], which consists in a technique

Hi [developer name],

I am a CS student at [university name], [country name]. I am working in an implementation of the Code City metaphor for Go.

For example, [repository/project] code city is at:

[link project as 3D city]

I would like to ask you two questions about this visualization:

1. Does it provide interesting insights about [repository/project] design or architecture? If yes, please describe.

2. How can it help on the maintenance and evolution of [repository/project]?

Fig. 7. Mail sent to Go developers

[10]https://github.com/avelino/awesome-go
[12]https://github.com/junegunn/fzf
to identify and record themes (i.e., patterns) in texts. The process includes the following steps: (1) reading the developers responses, (2) identifying preliminary codes for each response, (3) searching for themes among the defined codes, (4) merging similar themes, (5) defining the final themes. Steps 1 to 4 were performed independently by two authors of this paper. Then, the final themes were discussed and defined in a meeting. We use labels D01 to D12 to refer to developers.

B. Survey

1) Does GoCity provide interesting insights about software design and architecture? 11 developers answered this question. According to 8 developers (73%), our tool is useful to provide information about the project design and architecture. In addition, they detailed these answers. As presented in Figure 8, we grouped the answers in two categories: CODE MODULARIZATION and DESIGN PROBLEMS. We describe and give some examples of each group in the followings paragraphs.

![Fig. 8. Insights provided by GoCity](image)

**Code modularization.** In four answers (50%), the developers answered that GoCity helps to provide insights about the modules, complexity of the project, and distribution of the source code in the files. As example, we have the following answer:

*I really find the visualization useful to understand where most complexity is and how well decoupled things are.* (D02)

**Design problems.** This category includes four answers (50%), where developers report that GoCity can help to reveal possible design or architectural problems. As example of this category, we have the following answer:

*If I see high and large blue buildings it probably means that the struct is not respecting the single responsibility principle or that the type do a lot of things.* (D06)

2) Does GoCity help on software maintenance and evolution? 10 developers answered this question, and six developers (60%) agreed that GoCity can help in software maintenance activities. Figure 9 reports the summary of developers answers. We grouped the benefits pointed in the answers in two categories: IDENTIFYING REFACTORIZATION CANDIDATES and CODE COMPREHENSION. We describe and give some examples of each category in the followings paragraphs.

![Fig. 9. How GoCity can help on the software maintenance and evolution](image)

**Code Comprehension.** In four cases (67%), the developers said that the tool can help developers to get to a high-level understanding of the code base, and the modules distribution in the system. As example, we have:

*This seems like a great tool for getting a high-level understanding of a foreign code base. It can probably accelerate the on-boarding process for new contributors to a complex project.* (D02)

**Identifying Refactoring Candidates.** In two answers (33%), the developers said that GoCity can help developers to identify possible candidates to refactoring (e.g., split module, move elements). As example, we have this answer:

*Identifying unbalanced bits of the code in terms of file size and or struct size is useful to point out areas which might need refactoring.* (D12)

C. Threats to Validity

We send emails to owners of popular Go projects, asking for feedback about GoCity. However, we report insights provided by our tool considering only 12 answers. Besides that, the results cannot be generalized to other systems, since we only send emails to owners of popular and non-organizational projects hosted on GitHub. However, we select relevant systems among the top-100 most popular Go projects, and the answers were analyzed by two authors of the paper.

VI. RELATED WORK

The literature presents several approaches to support developers on maintaining and evolving software systems [4], [6], [8], [11]–[19]. Among these studies, there are techniques based in software visualization, which reveals relevant information about the source code [4], [6], [13]–[15], [20], [21]. Alnabhan et. al [14] present an approach based in a 2D software visualization. They represent the metrics from the source code using geometric forms. For example, a rectangle represents a class, and the number of lines defines the height of an object. The tool also provides information about the element relationships and method signatures. Scarsbrook et. al [8] present MetropolJS, a tool to analyze large JavaScript projects, based in a treemap layout.

Wettel and Lanza [4], [15] proposed a city metaphor to represent large systems as a 3D visualization. CodeCity implements the proposed technique, and it support the languages Java, C++ and Smalltalk [4]. In another study, the authors evaluate the relevance of the city metaphor by means of a controlled experiment [22]. They reveal that CodeCity can decrease the time to conclude software maintenance and analysis tasks.
In this context, there are other tools based in the code city metaphor. For example, Merino et al. [23] introduce CityVR, which uses virtual reality to represent the visualizations. In the JavaScript ecosystem, Viana et al [6] introduce JSCity, a tool to visualize JavaScript projects, which is also inspired in the code city metaphor. The authors report an experiment using the tool to build visualizations for 40 systems.

Recently, Nunes et al [7] implemented the city metaphor to Swift projects. The authors adapted the city metaphor to perform a visual representation of modular structures from Swift. The tool includes a web interface, where users can visualize and extract different information from the project (e.g., lines of code and classes).

However, these tools do not include support to Go, a popular and modern programming language to implement large and complex systems. In this paper, we describe a new tool named GoCity, which is also based on the code city metaphor. We adapted the metaphor to support Go features. To our knowledge, we are the first to survey developers, in order to collect their feedback and opinions about the code city of popular open source software projects.

VII. CONCLUSION

Several tools and approaches help developers to support software analysis, evolution, and maintenance. In this context, visualization techniques based on code cities are gaining momentum to support developers in software maintenance tasks. In this paper, we presented GoCity, an implementation of the code city metaphor to visualize Go projects as cities, which is public available at https://go-city.github.io. The tool provides a web interface to end users to interact and visualize the source code in 3D structures.

To evaluate the tool, we sent emails to 60 owners of popular Go projects hosted on GitHub. We received 12 answers. By analyzing their feedback, we show that GoCity can be used to provide insights about design problems and modularization. We also report that the tool can help on software maintenance tasks, since six developers pointed that GoCity is useful to identify refactoring candidates, and on code comprehension tasks.

As an additional contribution, we made the source code of GoCity publicly available on GitHub. Further studies can consider other emerging ecosystems (e.g., Rust and Elixir). We also plan to provide a new feature to visualize the software evolution over time.

ACKNOWLEDGMENT

We thank the 12 GitHub developers who shared their feedback about GoCity with us. This research is supported by FAPEMIG, CNPq, and CAPES.

REFERENCES


