Visualizing Swift Projects as Cities

Rafael Nunes, Marcel Rebouças, Francisco Soares-Neto, Fernando Castor
{rngs, mscr, fmssn, castor} @cin.ufpe.br | Federal University of Pernambuco

1. Problem and Motivation

Human's ability to perform software maintenance is compromised as a project gets bigger, older, and more complex. Software visualization can be used to ease software understanding and maintenance. In this work **we leverage software visualization to support Swift developers.**

2. Why Swift?

No visualization tools are available for Swift, a new language with notorious and growing adoption by the developer community. There are such tools as JSCity [1] and CodeCity [2], focusing on Java and JavaScript. Swift is different from these languages in part due to the availability of constructs such as **extensions** and **structs**.

3. City Metaphor applied to Swift

The City Metaphor was introduced by Wettel and Lanza [2]. This metaphor attempts to provide a city atmosphere as a representation of the code, providing to users a sense of locale and easy understanding of the program.

A building (block) represents a unit of development like a class or similar. In a Swift context we can use **Classes**, **Structs**, **Protocols**, **Enumerations** or **Extensions**.

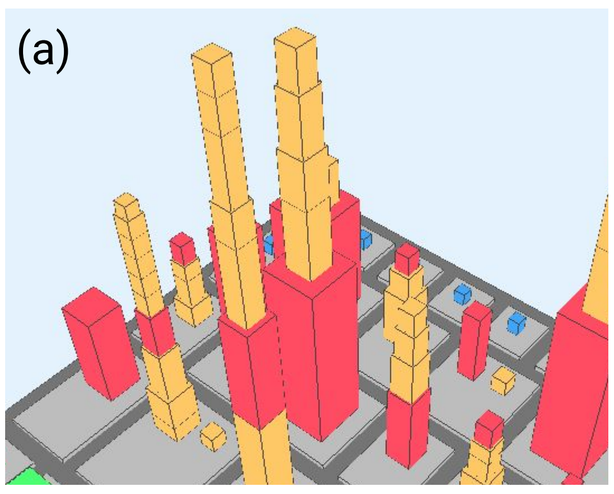
#Methods = width & depth; #LoC = height;

Classes · Extensions · Enums · Structs · Protocols

2-level topology

City level represents the entire project

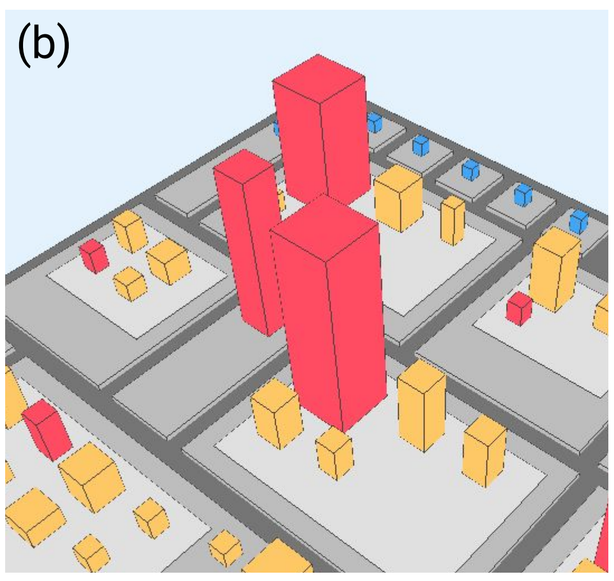
District level, represents each file of the project.



How do we handle extensions?

Stack up the extensions and its type giving rise to so-called skyscrapers (a).

Add **another level** to the topology, grouping together the related types (b).



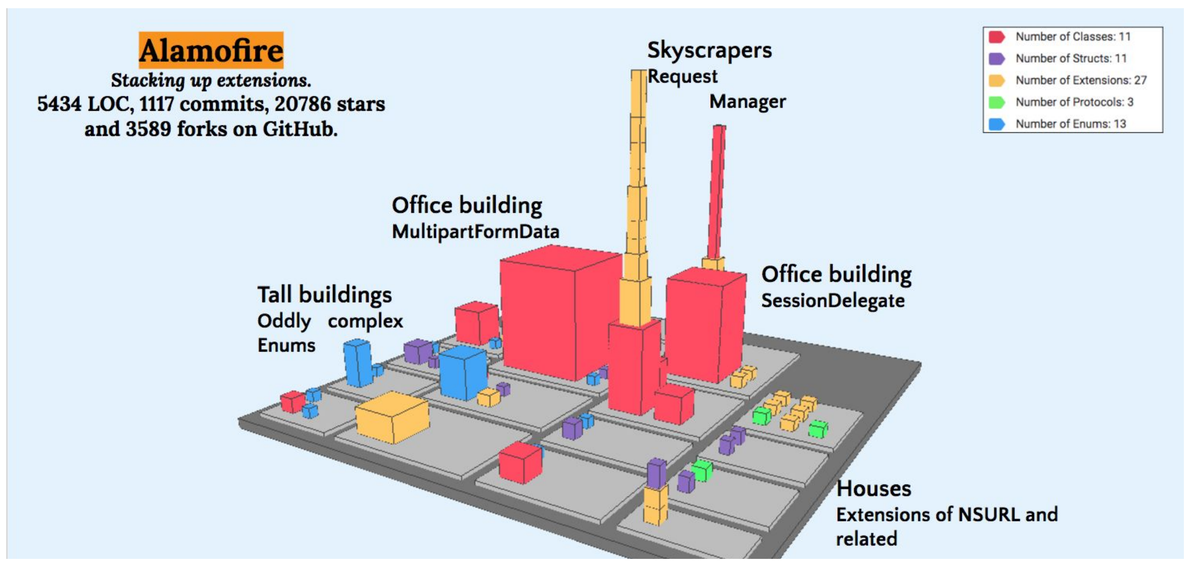
4. Tool Support

1. Custom toolchain to get output from the **AST**.
2. Then parse that output to generate a **metrics** file.
3. The metrics file is used as input for the **visualizer**.

Custom Toolchain → Parser → SwiftCity

http://swiftcity.github.io/

5. Case studies

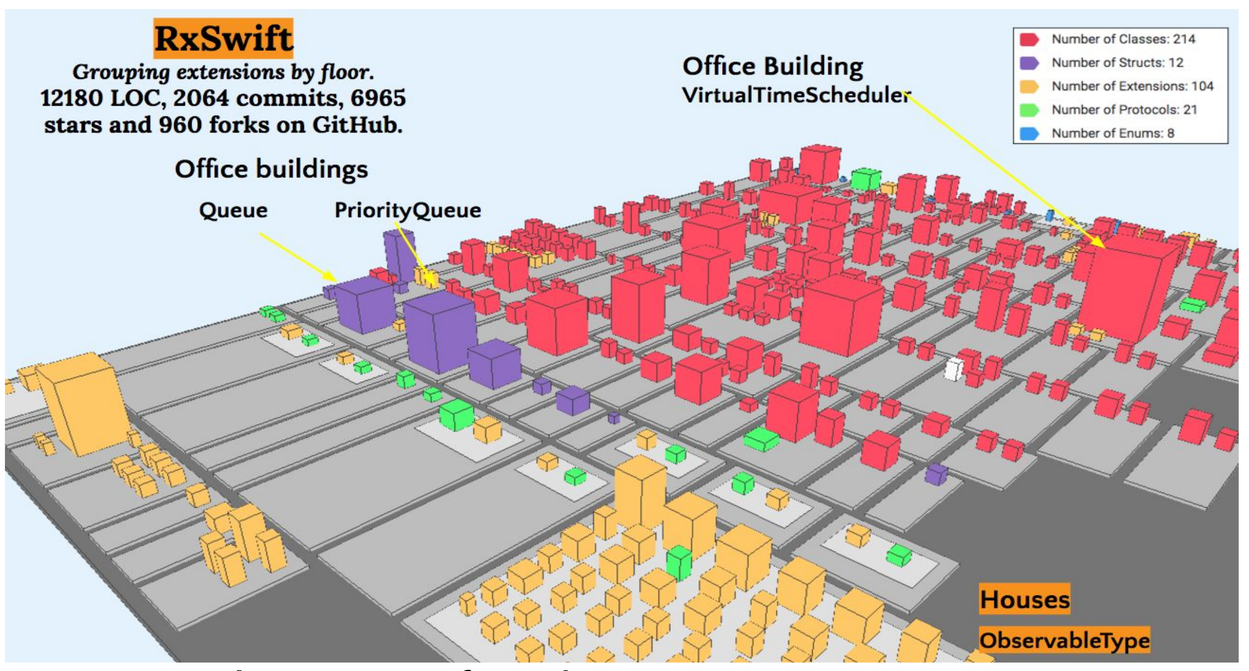


Alamofire: API for HTTP networking.

- Extensions as a pattern, but there are some **huge classes** in the city.
- Two strangely large blue towers (enums). This is unusual for enums. Therefore, they could be subject to inspection.

RxSwift: reactive programming framework.

- Few extensions that extend classes;
- More extensions to native Swift types, shown as ungrouped yellow blocks in the city;
- **53** extensions to a protocol (*ObservableType*);
- Structs are not as often used as in Alamofire.



Some simple patterns found:

1. **Yellow buildings throughout the city:** Extensions widely used in projects, often for decoupling and modularization. Some are **more complex than the extended type**.
2. **Variety in the colors of the blocks:** The systems design really makes use of all the standard Swift building blocks.

6. Contributions and Future Work

Contributions of the study are:

- Mapping elements of the Swift language to the city metaphor. A starting point for future works.
- Tools for developers to analyse their systems written in Swift, extract knowledge about them and generate insights about their improvement.

Future works to improve the visualization system and the completeness of the translated metaphor are:

- Dealing with variables and functions out of a type scope or new spatial relationships, for example.
- Implement other features like tag, filtering, search and others.

7. References

[1] M. Viana, E. Moraes, G. Barbosa, A. Hora, and M. T. Valente, "JSCity: Visualização de sistemas JavaScript em 3D," in III Workshop de Visualização, Evolução e Manutenção de Software (VEM), 2015, pp. 73–80.

[2] R. Wettel and M. Lanza, "Visualizing software systems as cities," ser. VISSOFT 2007, 2007, pp. 92–99.