M3triCity: Visualizing Evolving Software & Data Cities

Susanna Ardigò, Csaba Nagy, Roberto Minelli, Michele Lanza
REVEAL @ Software Institute – USI, Lugano

ABSTRACT

The city metaphor for visualizing software systems in 3D has been widely explored and has led to many diverse implementations and approaches. Common among all approaches is a focus on the software artifacts, while the aspects pertaining to the data and information (stored both in databases and files) used by a system are seldom taken into account.

We present M3TRICITY, an interactive web application whose goal is to visualize object-oriented software systems, their evolution, and the way they access data and information. We illustrate how it can be used for program comprehension and evolution analysis of data-intensive software systems.

Demo video URL: https://youtu.be/uBMvZFIlWtk

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Software and data visualization, Program comprehension

ACM Reference Format:
Susanna Ardigò, Csaba Nagy, Roberto Minelli, Michele Lanza. 2022. M3triCity: Visualizing Evolving Software & Data Cities. In *44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3510454.3516831

1 INTRODUCTION

Program comprehension is a fundamental activity for software maintenance and evolution. Developers spend considerably more time reading and understanding existing code rather than writing new code [12]. Software visualization is a popular technique to perform program comprehension [19]. Many techniques have been proposed, ranging from simple 2D displays, such as polymetric views [8] and UML diagrams to more complex 3D techniques, even extending into the realm of virtual (VR) and augmented reality (AR) [3, 10, 13].

We present M3TRICITY a web application that visualizes software systems in 3D, focusing on the evolution of systems and how they use and access data [1, 17]. M3TRICITY leverages the city metaphor [4, 6, 15, 16, 20, 21] in the vein of CODECITY [24] and runs on any modern web browser.

ICSE '22 Companion, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9223-5/22/05…$15.00
https://doi.org/10.1145/3510454.3516831

2 M3TRICITY IN A NUTSHELL

In a nutshell, M3TRICITY is a web-based evolution of CODECITY, which popularized the city-based visualization of software systems through the city metaphor [23] (See Figure 1).

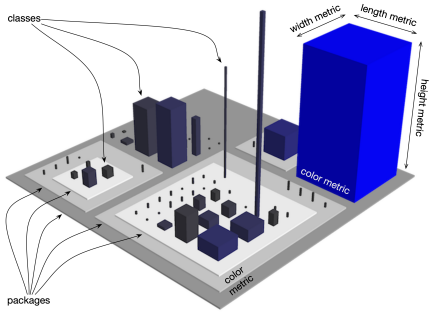


Figure 1: CodeCity and the City Metaphor

In CODECITY every class is visualized as a building with metrics mapped onto the base, height, and color, while packages were visualized as nested districts. M3TRICITY expands on it through a number of features and concepts, namely: (i) it distinguishes between the file types and uses different glyphs (*i.e.*, depictions) for them, as we see in Figure 2.

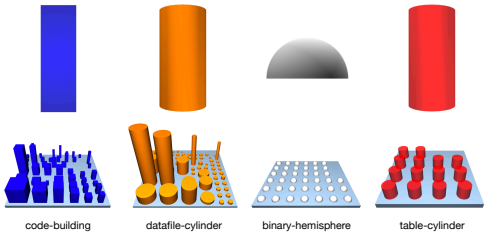


Figure 2: The Glyphs Used by M3triCity

Furthermore, (ii) it takes the evolution of a system into account for the layout of the city structure, as we detail in Section 2.4; (iii) it infers and visualizes the databases used by a system, as we detail in Section 2.2, and (iv) it provides higher accessibility by being publicly available as a web application available at **https://metricity.si.usi.ch/v2**.

2.1 The User Interface of M3triCity

Figure 3 shows the main user interface of M3TRICITY. At the center we see the software city Ⓐ, with folders and files represented as buildings and nested districts. Above the city the sky is used to visualize the (inferred) database(s) and their tables Ⓑ, where connecting lines represent the accesses performed from the source code.

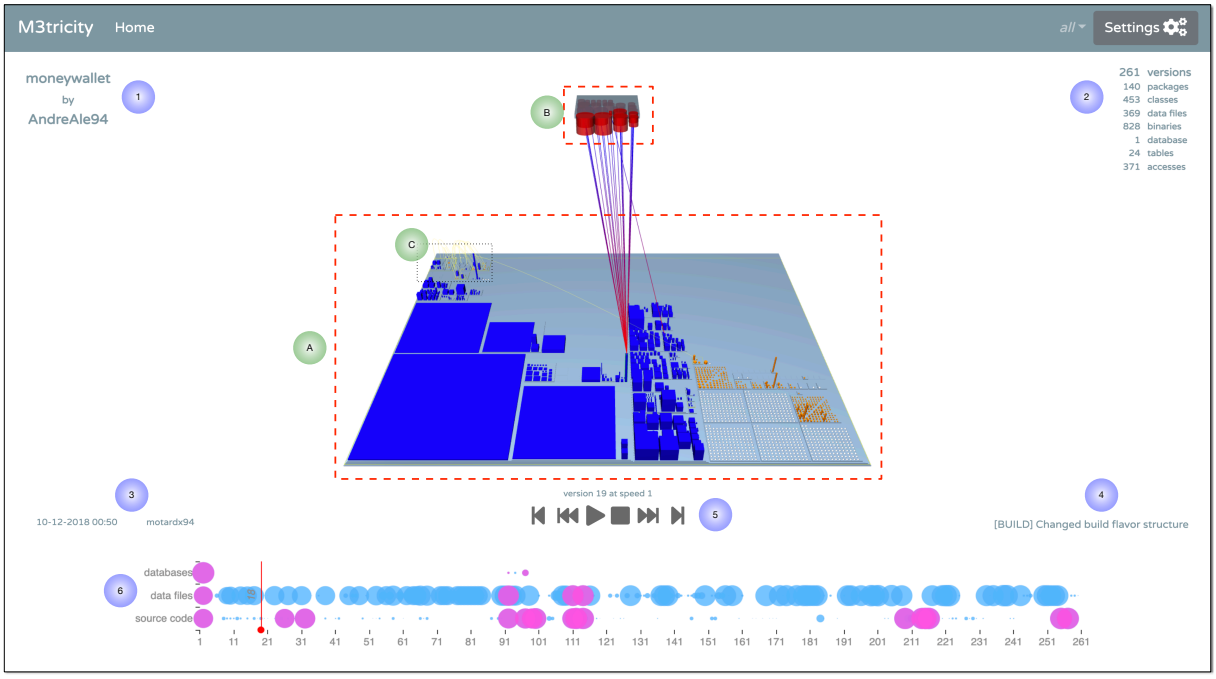


Figure 3: The Main Page of M3triCity

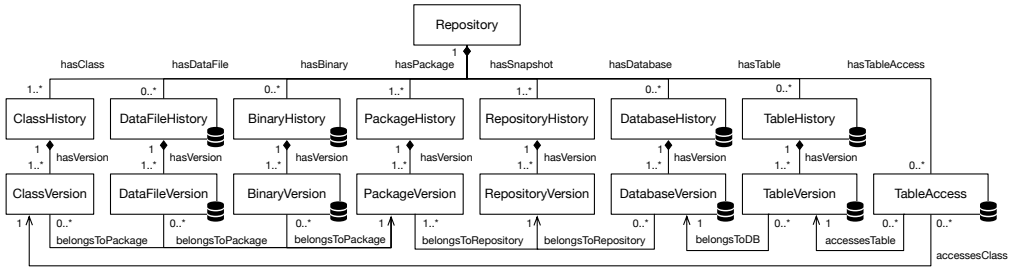


Figure 4: The Evolution Model of M3triCity

Various information about the system being visualized ① is also displayed ②. As M3TRICITY is geared towards evolution comprehension, additional panels provide information about the currently visualized commit ③④. To facilitate moving through time M3TRICITY provides a control panel ⑤ to easily moving forward/backward between different commits as well as fast back/forwarding and pausing. A timeline at the bottom ⑥ provides a global overview of the system evolution with additional details pertaining to the commits (*i.e.*, along the timeline), as well as instantaneous access to a specific place in the commit history. The whole city can be rotated, the user can also change its point of view and zoom in and out. Structural changes (*i.e.*, moving of entities) are depicted using yellow curved arcs (see top left annotation with black dots Ⓒ). When the user clicks on an artifact, it is highlighted both in the main visualization as well as along the timeline, denoting all commits in which the entity was involved. More customizations are also available in the settings panel.

2.2 Modeling Evolving Data-Intensive Systems

Figure 4 depicts the meta-model of M3TRICITY. Evolving software artifacts are modeled using "histories" in the vein of Girba's evolution meta-model [5]. For each artifact history, we model each version including binary files and data files (*e.g.*, JSON, XML). Databases are inferred through SQLINSPECT [14]. For each entity, M3TRICITY computes various metrics, summarized in Table 1.

Entity	Metric Name	Entity	Metric Name
Class	# Instance Variables # For Loops # Methods # Lines of Code	Data File	# Entities # Entity Types Max # Properties per Entity Max Nesting Level
Table	# Columns # Table Accesses	Binary	Size

Table 1: The Metrics Supported by M3triCity

2.3 Architecture

Figure 5 shows the architecture of M3TRICITY.

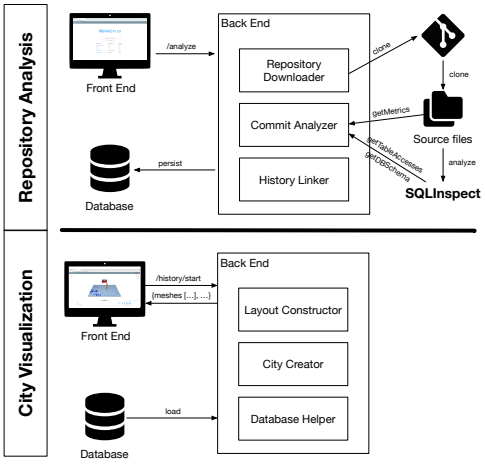


Figure 5: The Architecture of M3triCity

The frontend of M3TRICITY is implemented in TypeScript.[1] It uses Vue.js[2] for the user interface. The 3D visualization is created using Babylon.js.[3] The backend is a Spring Boot [4] application implemented using Java 11[5] and Gradle 6.[6]

To start the analysis, the user indicates the URL of a repository and, as an optional parameter, the database type. The frontend contacts the backend through the public REST API endpoint *analyze*. The execution starts in the module *Repository Downloader* which contacts git to clone the repository. The *Commit Analyzer* module iterates through all files of each snapshot of the given repository in chronological order, classifies them, analyzes them and then extracts the metrics. The project uses SQLINSPECT [14] to reverse engineer the schema of the database and the interactions with the source code. The histories of the entities are created by linking the versions. At last, all information is persisted in a MongoDB database.

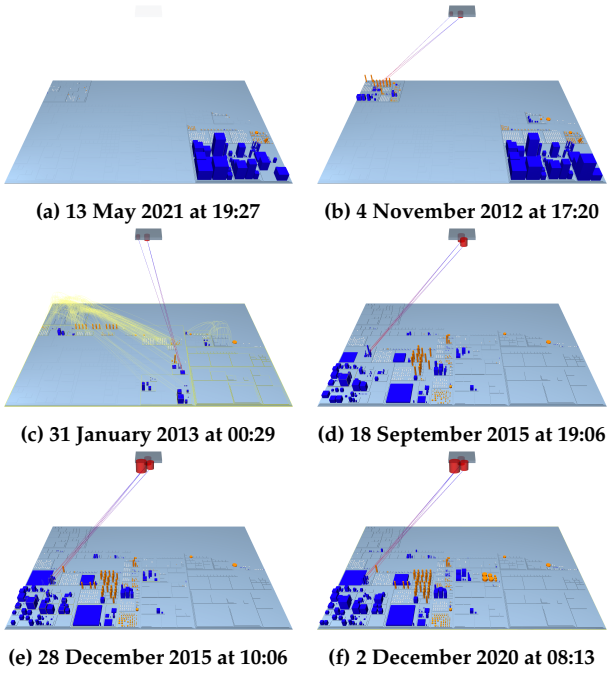
2.4 Usage

The user starts the visualization of a city by selecting a processed repository. The repository-related information is loaded from the database with the *Database Helper* component. The computation of the city layout is handled by the *Layout Constructor* component which iterates through all the entities of the city. M3TRICITY considers evolution as a first-class citizen, implementing a history-resistant layout, which allocates a dedicated position to each artifact throughout the lifetime [17]. The *City Creator* module creates the meshes information that are then sent to the frontend which renders them as 3D meshes.

[1]See **https://www.typescriptlang.org**
[2]See **https://vuejs.org**
[3]See **https://www.babylonjs.com**
[4]See **https://spring.io/projects/spring-boot**
[5]See **https://docs.oracle.com/en/java/javase/11/docs/api/**
[6]See **https://docs.gradle.org/6.3/release-notes.html**

3 SOFTWARE CITY TALES

We illustrate how M3TRICITY can be used to comprehend the evolution of a system, by using the GNUCASH-ANDROID[7] Android companion app of the GNUCASH accounting program as an example. GNUCASH-ANDROID allows recording transactions on-the-go to import the data into GNUCASH later. The main branch of the project is composed of 1,730 commits by 46 contributors. Figure 6 shows six M3TRICITY snapshots in the overall evolution.



(a) 13 May 2021 at 19:27 **(b) 4 November 2012 at 17:20**

(c) 31 January 2013 at 00:29 **(d) 18 September 2015 at 19:06**

(e) 28 December 2015 at 10:06 **(f) 2 December 2020 at 08:13**

Figure 6: The Evolution of GnuCash-Android

May 13 2012: GnuCash-Android is born (Figure 6a). On May 13, 2012, Ngewi Fet creates the project repository with 83 Java classes, 85 data files, and 243 binaries. The source code is mainly located in the `com_actionbarsherlock` package, nicely divided into sub-packages. The `res` folder contains three districts of images and several districts of data files. Separately, the module **GnucashMobile** has one Java class, four data files, and some binaries.

Nov 4 2012: The database is created (Figure 6b). The developers added a database which is being used by a part of the system that has been added in the meantime. Test classes are growing as well. Data files have been added, mostly related to text constants that need to be displayed.

Jan 31 2013: GnuCash-Android undergoes a major restructuring (Figure 6c). Almost 450 files are deleted, and 220 are moved with the renaming of the folder `GnucashMobile` to `app`. The database-related classes are still present but the accesses to the tables are removed.

[7]See **https://github.com/codinguser/gnucash-android**

Sep 18 2015: Controlled evolution (Figure 6d). Fast forwarding 2 years, the system keeps evolving: the test suite is expanded, the developers started working on a user interface module. The database co-evolves with the system, with the addition and quick deletion of tables.

Dec 28 2015: Extending the tests (Figure 6e). The system evolves mostly with new tests.

Dec 2 2020: Fast forward (Figure 6f). Five years later the system has grown considerably, with some classes reaching considerable size in terms of variables and methods. Data file districts have been added, complementing systematic database accesses on a well-organized DB featuring the three major tables `transactions`, `splits`, and `scheduled_actions`.

4 RELATED WORK

Since the seminal works of Reiss [18] and Young & Munro [25], many approaches to visualize software systems in 3D have been explored. The cities metaphor has been widely used and led to diverse implementations, such as the SOFTWARE WORLD by Knight *et al.* [6], the visualization of communicating architectures by Panas *et al.* [15], VERSO by Langelier *et al.* [7], CODECITY by Wettel *et al.* [23, 24], EVO-STREETS by Steinbrückner & Lewerentz [20], CODEMETROPOLIS by Balogh & Beszedes [2], and VR CITY by Vincur *et al.* [22].

Only a few approaches considered presenting data(bases) together with the source code, mostly using the city metaphor. Meurice and Cleve presented DAHLIA to visualize database schema evolution [11], which uses the city metaphor where buildings in the city represent database tables. Zirkelbach and Hasselbring presented RACCOON [26], which uses the 3D city metaphor to show the structure of a database based on entity-relationship diagrams. Marinescu presented a metamodel containing object-oriented entities, relational entities and object-relational interactions [9]. M3TRICITY does not separate source code and data(bases), like existing approaches, but it shows them with their interactions in the same city.

5 CONCLUSION

M3TRICITY extends the original city metaphor by considering an important aspect that has been ignored up to now: the data.

Our tool visualizes object-oriented software systems, their evolution, and the way they access data and information. M3TRICITY expands the original city metaphor by adding a number of features and concepts: using different glyphs to distinguish between the various file types, taking software evolution into account to layout the city, inferring and visualizing the databases used by a system, and providing higher accessibility by being publicly available as a web application.

To demonstrate the usefulness of our approach, we illustrate how M3TRICITY can be used to comprehend the evolution of a data-intensive system: GNUCASH-ANDROID.

ACKNOWLEDGMENTS

We acknowledge the financial support of the Swiss National Science Foundation and the Fonds de la Recherche Scientifique for the project "INSTINCT" (SNF Project No. 190113).

REFERENCES

[1] Susanna Ardigò, Csaba Nagy, Roberto Minelli, and Michele Lanza. 2021. Visualizing Data in Software Cities. In *Working Conference on Software Visualization, VISSOFT 2021*. IEEE, 145–149.
[2] Gergő Balogh and Arpad Beszedes. 2013. CodeMetropolis - code visualisation in MineCraft. In *Proc. 13th Int. Working Conf. Source Code Analysis and Manipulation*. IEEE Computer Society, 136–141.
[3] Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring. 2015. Exploring software cities in virtual reality. In *3rd IEEE Working Conference on Software Visualization, VISSOFT*. IEEE Computer Society, 130–134.
[4] Florian Fittkau, Jan Waller, Christian Wulf, and Wilhelm Hasselbring. 2013. Live trace visualization for comprehending large software landscapes: The ExplorViz approach. In *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE Computer Society, 1–4.
[5] Tudor Adrian Girba. 2005. *Modeling history to understand software evolution*. Ph.D. Dissertation. University of Bern.
[6] Claire Knight and Malcolm Munro. 2000. Virtual but visible software. In *Proc. 17th Int. Conf. Information Visualization*. IEEE, 198–205.
[7] Guillaume Langelier, Houari Sahraoui, and Pierre Poulin. 2005. Visualization-based Analysis of Quality for Large-scale Software Systems. In *Proc. 20th Int. Conf. Automated Software Engineering*. ACM, 214–223.
[8] Michele Lanza and Stéphane Ducasse. 2003. Polymetric Views - A Lightweight Visual Approach to Reverse Engineering. *IEEE Trans. Software Eng.* 29, 9 (2003), 782–795.
[9] Cristina Marinescu. 2019. Applications of Automated Model's Extraction in Enterprise Systems. In *Proc. 14th Int. Conf. Software Technologies (ICSOFT 2019)*. SCITEPRESS, 254–261.
[10] Leonel Merino, Alexandre Bergel, and Oscar Nierstrasz. 2018. Overcoming Issues of 3D Software Visualization through Immersive Augmented Reality. In *Working Conf. on Software Visualization, VISSOFT*. IEEE, 54–64.
[11] Loup Meurice and Anthony Cleve. 2016. DAHLIA 2.0: A Visual Analyzer of Database Usage in Dynamic and Heterogeneous Systems. In *Proc. 2016 Working Conf. Software Visualization (VISSOFT)*. IEEE, 76–80.
[12] Roberto Minelli, Andrea Mocci, and Michele Lanza. 2015. I know what you did last summer-an investigation of how developers spend their time. In *23rd International Conference on Program Comprehension*. IEEE, 25–35.
[13] David Moreno-Lumbreras, Roberto Minelli, Andrea Villaverde, Jesús M González-Barahona, and Michele Lanza. 2021. CodeCity: On-Screen or in Virtual Reality?. In *VISSOFT 2021*. IEEE, 12–22.
[14] Csaba Nagy and Anthony Cleve. 2018. SQLInspect: a static analyzer to inspect database usage in Java applications. In *40th International Conference on Software Engineering: Companion Proceedings, ICSE*. ACM, 93–96.
[15] Thomas Panas, R. Berrigan, and John Grundy. 2003. A 3D metaphor for software production visualization. In *Proc. 7th Int. Conf. Information Visualization*. IEEE Computer Society, 314 – 319.
[16] Thomas Panas, Thomas Epperly, Daniel J. Quinlan, Andreas Sæbjørnsen, and Richard W. Vuduc. 2007. Communicating Software Architecture using a Unified Single-View Visualization. In *Proc. 12th Int. Conf. Engineering Complex Computer Systems*. IEEE Computer Society, 217–228.
[17] Federico Pfahler, Roberto Minelli, Csaba Nagy, and Michele Lanza. 2020. Visualizing Evolving Software Cities. In *Proceedings of VISSOFT 2020 (8th Working Conference on Software Visualization)*. IEEE CS Press, 22–26.
[18] Steven P. Reiss. 1995. An Engine for the 3D Visualization of Program Information. *J. Visual Languages & Computing* 6, 3 (1995), 299–323.
[19] John T. Stasko, Marc H. Brown, and Blaine A. Price. 1997. *Software Visualization*. MIT Press.
[20] Frank Steinbrückner and Claus Lewerentz. 2010. Representing development history in software cities. In *Proceedings of the ACM 2010 Symposium on Software Visualization*. ACM, 193–202.
[21] Yuriy Tymchuk, Andrea Mocci, and Michele Lanza. 2015. ViDI: The Visual Design Inspector. In *37th IEEE/ACM International Conference on Software Engineering, ICSE*. IEEE Computer Society, 653–656.
[22] Juraj Vincur, Pavol Návrat, and Ivan Polásek. 2017. VR City: Software Analysis in Virtual Reality Environment. In *Proc. Int. Conf. Software Quality, Reliability and Security Companion*. IEEE Computer Society, 509–516.
[23] Richard Wettel and Michele Lanza. 2007. Visualizing Software Systems as Cities. In *Proc. 4th Int. Workshop on Visualizing Software for Understanding and Analysis*. IEEE Computer Society, 92–99.
[24] Richard Wettel and Michele Lanza. 2008. CodeCity: 3D Visualization of Large-Scale Software. In *Proceedings of ICSE 2008 (30th International Conference on Software Engineering)*. ACM Press, 921–922.
[25] Peter Young and Malcolm Munro. 1998. Visualizing Software in Virtual Reality. In *6th International Workshop on Program Comprehension (IWPC '98), June 24-26, 1998, Ischia, Italy*. IEEE Computer Society, 19–26.
[26] Christian Zirkelbach and Wilhelm Hasselbring. 2019. *Live Visualization of Database Behavior for Large Software Landscapes: The RACCOON Approach*. Technical Report. Department of Computer Science, Kiel University.