Does UML Modeling Associate with Lower Defect Proneness?

Slightly

Adithya Raghuraman
STRUDEL @CMU

Truong Ho-Quang
Chalmers U of Tech

Michel Chaudron
Chalmers U of Tech

Alexander Serebrenik
TU Eindhoven

Bogdan Vasilescu
STRUDEL @CMU
Belief vs evidence

40 percent of major decisions are based not on facts, but on the manager’s gut.


Opinion Formation

(Devanbu et al, ICSE 2016)
We revisit a widely-held belief: Does the use of UML modeling, on average, correlate with higher software quality?
Natural experiment

Mine OSS GitHub projects
Natural experiment

“Control” group

“Treatment” group

Yes

No
Natural experiment

(Robles et al, MSR 2017)

An extensive dataset of UML models in GitHub

Gregorio Robles,† Truong Ho-Quang,* Regina Hebig,** Miguel Angel Fernandez†
†Chalmers — Goteborg University, Goteborg, Sweden
‡Gmail: truongh, hebig, chaudron@gmail.com
§grex@gcsy.urjc.es, mafesan.nsn@gmail.com
¶{truongh, hebig, chaudron}@chalmers.se

Abstract—The Unified Modeling Language (UML) is widely taught in academia and has great acceptance in industry. However, there is not a single dataset of UML diagrams publicly available. This is due to a dataset of UML files together with meta-data of the software projects where the UML file belongs to, that is, two main facts: (i) UML files are stored in various formats and are found in a large number of repositories hosted in GitHub; and (ii) the UML meta-data must be collected manually. We present a systematic approach to collect UML stored in images, .xmi, and .uml files. We have mined GitHub repositories hosted in GitHub, together with their source code and development meta-data, to discover UML diagrams. The result of this effort is a dataset with over 4,650 projects in GitHub.

Keywords: UML, GitHub, modeling, mining software repositories.

I. INTRODUCTION

The Unified Modeling Language (UML) provides the facility for software engineers to specify, construct, visualize and document the artifacts of a software-intensive system and to facilitate communication of ideas [1]. UML is commonly taught in the computer science curriculums worldwide, and the use of UML is generally accepted in industrial software development. However, there is not a single publicly available example of UML is taught in such files. To the knowledge of the authors, the largest UML dataset comprises around 100,000 models obtained by collecting examples from the literature, web searches, and donations. However, that dataset only contains lone-standing diagram. Thus, it cannot be used for studying the software systems and projects associated with these diagrams.

Even though it has been reported the UML is marginally used in Open Source projects [2], the large amount of repositories hosted in GitHub offers the possibility to look for a large number of UML models used in software development projects, together with their source code and development meta-data. This is the reason why we have mined GitHub for UML files. The result of this effort is a dataset with over 4,650 files with UML diagrams. These diagrams comprise several types and formats, and offer a valuable data source for researchers and practitioners. This dataset is of great interest as real-world examples in class, and for further research.

The remainder of this paper is structured as follows. Next, we introduce how we have extracted the data. Section III—dataset; UML; GitHub; modeling; mining software repositories.

II. EXTRACTION METHODOLOGY

The data extraction process comprises the following four steps: (i) retrieval of the tree (file list) from GitHub repositories (Section II-A); (ii) identification (grouping) of potential UML files (Section II-B); (iii) automated examination (and manual evaluation) of the existence of UML notation in the obtained files (Section II-C); and (iv) retrieval of the meta-data from those repositories where a UML file has been identified (Section II-D).

A. Step 1: Mining GitHub

We accept a list of GitHub repositories obtained from GHTorrent [4], which offers a list of over 100,000 non-forked and deleted repositories. Since GHTorrent now distributes CSV files (one file per repository) instead of mysqldump based backups, we use data available in the projects.csv file: the URL of the project and the values of non-forked (true, fork (false), and deleted (false) (we discard those projects that are forks or have been removed/deleted).

For those projects that are not forks and not have been deleted, we retrieve from the GHTorrent API the tree (file list) for the master branch. If the master branch does not exist, then we retrieve the branch that has the most commitments. If the project has not as default, and perform a third request to download its URL. We try to then GitHub adds the .xmi and those projects, given the GitHub API limitation of 5,000 requests/minute, it would take around 14 months to perform the retrieval of data in this first step. As this would have made the data gathering unfeasible, we downloaded the JSON files in parallel with over 20 active GitHub accounts, which were deleted during this process. This reduced the time to approximately one month. For almost 25% of the repositories we obtained an empty JSON file or an error message from the GitHub API, because the repository has been deleted or made private in the time that goes from GHTorrent obtaining its data (which is before February 1st 2016) and our request to the GitHub API cloning attempt (in 2017).

For each of those projects that are forks nor have been deleted, we retrieve from the GHTorrent API the tree (file list) for the master branch. If the master branch does not exist, then we retrieve the branch that has the most commitments. If the project has not as default, and perform a third request to download its URL. We try to then GitHub adds the .xmi and those projects, given the GitHub API limitation of 5,000 requests/minute, it would take around 14 months to perform the retrieval of data in this first step. As this would have made the data gathering unfeasible, we downloaded the JSON files in parallel with over 20 active GitHub accounts, which were deleted during this process. This reduced the time to approximately one month. For almost 25% of the repositories we obtained an empty JSON file or an error message from the GitHub API, because the repository has been deleted or made private in the time that goes from GHTorrent obtaining its data (which is before February 1st 2016) and our request to the GitHub API cloning attempt (in 2017).

4,650 projects
http://oss.models-db.com
Natural experiment

“Treatment” group

- 50 projects
  - C++, C#, Java
  - 2009+
  - 10+ stars
  - 30+ issues
  - Issues in English

Mine the issue tracker
Natural experiment

“Control” group

93 projects
Same filters, sampled using GHTorrent

“Treatment” group

50 projects
- C++, C#, Java
- 2009+
- 10+ stars
- 30+ issues
- Issues in English
Natural experiment

“Control” group

Separate bugs from features etc.

Naive Bayes classifier 89% accuracy

“Treatment” group
Natural experiment

“Control” group

“Treatment” group

Compare defect rates
Projects w/ UML: ~35% fewer bugs reported than projects w/ UML (R^2 = 58%)

Num bug issues

Num commits +
Num stars +
Has license -

Has UML -

Plus controls for:
- Proj age
- Num contribs
- Test suite ratio
- Comment ratio
- Has CI
- Language
Projects w/ UML: ~35% fewer bugs reported than projects w/t UML ($R^2 = 58\%$)

Num bug issues

- Num commits (57% of variance explained)
- Num stars
- Has license

Plus controls for:
- Proj age
- Num contribs
- Test suite ratio
- Comment ratio
- Has CI
- Language

< 0.5% of GitHub repos have UML

Only 50 UML projects in this model

Has UML (2% of variance explained)
Does UML Modeling Associate with Lower Defect Proneness?: A Preliminary Empirical Investigation

Adithya Raghuraman
Carnegie Mellon Univ.
adithya@cmu.edu

Truong Ho-Quang,
Michel R. V. Chaudron
Chalmers University Chalmers
{truong, chaudron}@chalmers.se

Abstract—This benefits of modeling the design to improve the quality and maintainability of the software being developed have been extensively discussed and recognized. Yet, the empirical evidence on the impact of UML modeling on software defects is sparse. To fill this gap, we investigate through an empirical study of the relationship between UML modeling and software defect proneness in a large sample of open-source GitHub projects. Using statistical modeling and controlling for confounding factors, we show that projects containing UML models had statistically significantly fewer defects in software issues mined from their issue trackers than projects without traces of UML models. The benefits of using UML in software design, UML software quality, open-source software.

I. INTRODUCTION
Software design is widely accepted as a fundamental step in developing high-quality software [1]. By making designs developers go through a process of abstraction, reasoning about the architecture, and clarifying the structure of the system, it is estimated that software designs help to avoid many defects and improve system maintainability [2]. The benefits of software designs are also well documented; architectural decisions that developers make become well documented, reducing information loss and potential misinterpretation during system implementations and facilitating communication among team members and the onboarding of new developers [2]. Both commercial [2] and open-source software developers [2] alike recognize these potential benefits.

Among modeling languages, the Unified Modeling Language (UML) is often viewed as the de facto standard for describing the system’s design using diagrams [3]. In practice, UML is often used in a less formalized manner (not adhering strictly to the standard) [4]. Also, UML is used selectively, focusing on important, critical or novel parts. Still, despite many benefits of having UML as software development outcomes, the empirical evidence on the matter is scarce. Notable exceptions include a study by Attriboulis et al. [7], showing through two controlled experiments involving students that, after a learning curve, the availability of UML models may increase the student confidence and the design quality of subsequent code changes. There is also work by Fernández-Saiz et al. [10] that suggests an overall positive influence of UML modeling towards software maintenance. Finally, we note an empirical study by Nugroho and Chaudron [2] of an industrial Java system, showing that classes for which UML-modelled classes, on average, have a lower defect density that those that were not.

In this paper we study the intuitive and widely held belief that the use of UML-modeling on average, should correlate with higher software quality. To this end, we statistically analyze empirical data obtained from 143 open-source GitHub projects. Many hypotheses about the benefits of UML models on specific software maintenance outcomes have been proposed [7]. However, more generally, one can expect that the more practice of UML modeling as part of software development indicates a high team- and process maturity and discipline that, in turn, should lead to higher quality code.

In search of evidence [8] to substantiate this belief, we start from a publicly available data set of open-source soft-ware projects not known to use UML models. 1) Gather data from the GitHub hosting repository of both sets of projects (using and not using UML models), estimating that defect rates for a given project repository is the mean defect rate for the project repository as a whole. 2) Use this data to train a classification model estimating the likelihood of a project repository using UML modeling, on defect proneness, while controlling for confounding factors. In this manner, a small statistically significant effect of using UML models on defect proneness, i.e., projects with UML models tend to have fewer defects.

II. METHODOLOGY
We designed a quasi-experiment to compare the defect proneness between two groups of open-source GitHub projects: a treatment group of projects using UML models, and a control group of projects not known to use UML models. The data set was provided by GitHub and comprises 143 projects sampled randomly using GHTorrent [9]. We describe our data collection and analysis process next.

A. Data
As part of a previous study [11], Robles et al. [9] released a data set of 4,650 non-anonymized GitHub projects, defined as having at least six months of activity between their first and most recent commits and at least two contributors, that use UML models, as identified by a manually-annotated automated repository mining technique. As an independent factor of defect proneness involves mining the projects' source-repositories.

<0.5% of GitHub repos have UML
Only 50 UML projects in this model
Has UML (2% of variance explained)
Has UML (2% of variance explained)
Num commits (57% of variance explained)
Num stars
Has license
Plus controls for:
• Proj age
• Num commits
• Test suite ratio
• Comment ratio
• Has CI
• Language
<0.5% of GitHub repos have UML
Only 50 UML projects in this model
Has UML (2% of variance explained)
Has UML (2% of variance explained)
Num commits (57% of variance explained)
Num stars
Has license
Plus controls for:
• Proj age
• Num commits
• Test suite ratio
• Comment ratio
• Has CI
• Language

Projects w/ UML: ~35% fewer bugs reported than projects w/o UML (R² = 58%)