# Poster: Exploring Uncertainty in GitHub OSS Projects

— When and How Do Developers Face Uncertainty? —

Naoyasu Ubayashi Hokuto Muraoka Daiki Muramoto Yasutaka Kamei Ryosuke Sato
Kyushu University
Fukuoka, Japan
{ubayashi,muraoka,muramoto,kamei,sato}@posl.ait.kyushu-u.ac.jp

## ABSTRACT

Recently, many developers begin to notice that uncertainty is a crucial problem in software development. Unfortunately, no one knows how often uncertainty appears or what kinds of uncertainty exist in actual projects, because there are no empirical studies on uncertainty. To deal with this problem, we conduct a large-scale empirical study analyzing commit messages and revision histories of 1,444 OSS projects selected from the GitHub repositories.

## CCS CONCEPTS

• **Software and its engineering** → *Software creation and management*;

## KEYWORDS

Uncertainty, Empirical Study, OSS, Commit Message Analysis

**ACM Reference Format:**
Naoyasu Ubayashi Hokuto Muraoka Daiki Muramoto Yasutaka Kamei Ryosuke Sato. 2018. Poster: Exploring Uncertainty in GitHub OSS Projects: — When and How Do Developers Face Uncertainty? —. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194966

## 1 INTRODUCTION

Nowadays most developers, regardless of open source software (OSS) or industry projects, admit that uncertainty is a real problem in software development. Indeed, uncertainty has attracted a growing interest among researchers. Research themes spread over uncertainty of goal modeling, UML modeling, model transformations, and testing. Unfortunately, no one knows how often uncertainty appears or what kinds of uncertainty exist in actual projects, because there are no empirical studies on uncertainty. We assume that uncertainty is a crucial issue in software development. However, state-of-the-art research does not provide an evidence for guaranteeing this assumption. Moreover, it is unclear what kinds of issues exist in actual projects.

We conduct a large-scale empirical study analyzing commit messages and revision histories of GitHub [1] OSS projects to confirm whether or not our assumption is really true. Our study addresses the following three research questions:

**Research Questions and Motivation**

**RQ1:** What kinds of uncertainty appear?
Most people might consider that developers suffer from many kinds of uncertainty: program defects whose reason is uncertain, strange program behavior, unexpected value of a program variable, and program code whose implementation policy is doubtful. We provide an evidence showing that these subjective observations are not wrong.

**RQ2:** How do committers deal with uncertainty?
In RQ1, we understand what kinds of uncertainty appear in GitHub projects. However, it is not yet clear what kinds of actions committers take when facing uncertainties.

**RQ3:** When and Why does uncertainty appear?
In RQ1 and RQ2, we analyze uncertainty by inspecting commit messages. However, as another aspect, we should analyze uncertainty from the historical and causal viewpoints. In RQ3, we address the following three sub research questions: RQ3-1) How often do uncertainty commits appear?; RQ3-2) How many time are code snippets modified until an uncertainty commit eventually appear?; and RQ3-3) Why are uncertainty commits raised? It is important to answer the reasons why uncertainty appears. If we can know the reasons, we are able to provide tool support for dealing with uncertainty.

## 2 EMPIRICAL STUDY SETUP

In this section, we explain data set and an approach to identifying uncertainties.

### 2.1 Data Set

The GitHub repositories as of February 2017 are used in this empirical study. 1,444 projects are randomly selected from the GitHub repositories and used as the data set. This data set contains various project data ranging from large ones (e.g., Linux) to small ones. We use commit messages in our investigation, because text data included in commit messages can be a clue to analyze the tendency of uncertainty in GitHub.

### 2.2 Identifying Uncertainty

In this study, we assume that a committer deals with some kinds of uncertain concerns when he or she modifies the code and commits with the message containing the keywords related to uncertainty. We call these commits and keywords *uncertainty commits* and *uncertainty keywords*, respectively. Of course, a commit message containing uncertainty keywords does not always indicate an actual

[1] https://github.com/github


*ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden*

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194966

**Table 1: Uncertainty Keywords**

| Synonyms for "Uncertainty" (Oxford American Writer's Thesaurus) |
|---|
| debatable, undetermined, unsure, unpredictable, unforeseeable, incalculable, risky, chancy, dicey, informal, iffy, vague, ambiguous, unknown, unascertainable, obscure, arcane, changeable, irregular, unreliable, unsettled, erratic, fluctuating, doubtful, dubious, undecided, irresolute, vacillating, unclear, ambivalent, hesitant, tentative, faltering, unconfident, may, might, probably, fuzzy |

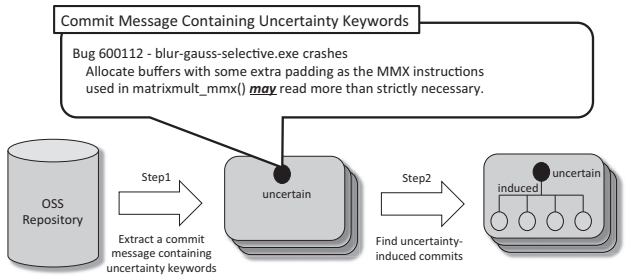


**Figure 1: Modified SZZ Algorithm for Uncertainty**

uncertainty commit. On the other hand, uncertain concerns might be dealt with even when a committer writes a commit message that does not contain uncertainty keywords. In general, it is not easy to identify uncertainty automatically and correctly, because we have to understand the deep semantics of the target commits that can be affected by not only code modifications but also requirements or design documents. In some cases, these documents may not be stored in project repositories. Taking these situations into account, it is practically reasonable to admit our assumption as a first step approximation of uncertainty identification. In our study, not only automated analysis based on text mining but also manual inspection is performed to avoid mis-judging of uncertainty commits.

**Dictionary for Uncertainty:** The uncertainty keywords shown in Table 1 are used as a dictionary for determining whether a commit deals with uncertainty or not. These keywords, synonyms for "Uncertainty", are extracted from Oxford American Writer's Thesaurus.

**Historical Uncertainty Identification:** An uncertainty commit can be affected by a series of previous code modifications. We need to detect not only an uncertainty commit itself but also commits affecting the uncertainty commit in order to answer RQ3. We adopt the SZZ algorithm [3] to automatically identify the changes that eventually caused uncertainty commits as illustrated in Figure 1. We slightly changed the original SZZ algorithm, because it does not find uncertainty-causing changes but bug-introducing changes. The original algorithm links each bug fix to the program code change introducing the original bug by combining information from the version archive such as Git with the issue tracking system such as Bugzilla. Our algorithm consists of two steps. First, it identifies the change related to uncertainty. Our algorithm searches uncertainty keywords shown in Table 1. The second step identifies when the uncertainty is potentially introduced or when the code snippets eventually affecting the uncertainty commit are induced. We use the diff command to locate the lines that were changed by the uncertainty commit. Then, we use the annotate command to trace back to the last revision that changed those patched lines.

## 3 STUDY RESULTS

We could obtain the following answers to three research questions.

**RQ1: What kinds of uncertainty appear?**
Uncertain program behavior (ratio is 36.05 %), uncertain variable / value / name (14.97 %), and uncertain program defects (11.56 %) are major kinds of uncertainty.

**RQ2: How do committers deal with uncertainty?**
Committers take one of the following actions: considering about uncertainty (47.06 %), resolving uncertainty (38.24 %), and escaping / ignoring uncertainty (14.71 %). In some situations, committers tend to take an action for not resolving but escaping or ignoring uncertainty.

**RQ3: When and Why does uncertainty appear?**
Uncertainty exists in the ratio of 1.44% (average) and developers cannot ignore its existence (RQ3-1). Uncertainty commits appear shortly after uncertainty-causing commits (median: two changes by two committers). However, there are cases in which the code is repeatedly modified many times (RQ3-2). Most of the uncertainties are categorized into *Known Unknowns* [1]. As a reason of uncertainty, the ratio of future requirements is close to half (RQ3-3).

## 4 LESSONS LEARNED

Our findings show that developers cannot ignore the existence of uncertainty. The answers to the research questions give us an opportunity for discussing how to support developers facing uncertainty. It is effective to provide the uncertainty-aware software development environment reflecting our findings.

Embracing uncertainty in software development is one of the crucial research topics in software engineering. Garlan, D. discusses the future of software engineering from the viewpoint of uncertainty [2]. He claims that *software engineering is founded on a computational myth that no longer fully serves its purpose: that the computational environment is predictable and in principle fully specifiable, and that the systems that compute in those environments can in principle be engineered so that they are trouble-free.* He argues that we must embrace uncertainty within the engineering discipline of software engineering.

Our empirical study is the first attempt to explore uncertainty in actual software development. As clarified in this paper, uncertainty exists everywhere in a certain percentage.

**Acknowledgments:** This work was supported by JSPS KAKENHI Grant Number JP 26240007.

## REFERENCES

[1] Elbaum, S. and Rosenblum, D. S., Known Unknowns: Testing in the Presence of Uncertainty, In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*, pp.833-836, 2014.
[2] Garlan, D., Software Engineering in an Uncertain World, In *Proceedings of FSE/SDP Workshop on Future of Software Engineering Research (FoSER 2010)*, pp.125-128, 2010.
[3] Śliwerski, J., Zimmermann, T., and Zeller, A.: When Do Changes Induce Fixes?, *ACM Sigsoft Software Engineering Notes*, Vol. 30, No. 4, pp.1-5, 2005.