Good morning, thank you for being here. My name is Qimu Zheng, I am from Peking University. Today, I am going to talk about **data quality issues**. - We are going to introduce a method to identify and correct problematic software activity data.
So, let's get started.

————

First of all, let's agree that software activity data, such as code/issue reports, are important, for both developers and software engineering researchers.
For developers, these data are used to assess software quality, manage the development process and many other tasks.
While for researchers, these data are widely used to answer all sorts of empirical research questions in software engineering.

————-

Recently, thanks to the rapid development of open source world, more and more software activity data become available to us. We have more code, more issues reports, more communication in mailing list and even more social media data related to software development.

————-

This is, of course, great news for our research.
In empirical software engineering research, we gather the available activity data. We build on top of them our statistical or machine learning models. Finally we reach our conclusions by analysing or testing the models we built.

—————

With this approach, we manage to answer various kinds of questions.
We measure the developers' productivity.
We predict which bug report is duplicate before it reaches developers.

We predict the time needed to resolve an issue.

Basically, we try to answer the questions that can be quantitatively investigated with software activity data.

———————

However, this process is not free of problems in that the software activity data may be low-quality. If there are some data quality issues with the activity data, and we built our model on top of these data, the reliability of the research result will be jeopardised.

———-

And, we **do** find some real issues.

————

In software activity data, the task completion time is important.

For researchers, it is often used to measure productivity or predict how long it takes to fix a bug and so forth.

For practical development, it is used to manage the progress of development.

In research, the timestamp that an issue is marked as fixed in issue tracking system is often used to represent the task completion time.

so, let's make the same assumption, and take a look at how well the developers do their job.

We conduct an experiment on the issue tracking data from Mozilla community, and we count the number of bugs fixed by each developer on each day.

———

And then, we are surprised.

Here, the x-axis represents the dates, the y-axis represents the number of bugs, and each point represents the number of bugs fixed by one individual on one single day. While the mean is 1.92, less than 2 bugs a day, some developers are recorded to fix more than 50 bugs on one day.

Fixing more than 50 bugs on one day, that’s quite a lot.
The data quality issues of this piece of data should not be easily ignored.

———-

We turn to check out the research on data quality. but not much can be found.

We then tried to find the research mentioning data quality, still, not much can be found. Instead, we found 3 papers showing that data quality consideration is a minority practice in software engineering research.

———

So, it seems to us that researchers love data, yet few care about their quality.

———-

So we try to fix the problem:

We propose a method to identify and correct problematic data

We then applied the method to the task completion time data

and we discuss the possibility of generalising our method

———-

But before all that, i want to share two observations about software activity data, based on which we propose our method for identifying and correcting problematic data.

————

The first one is capacity constraints.

Physical constraints tend to bound what could be accomplished by any individual or group over a fixed period of time. For example, a developer may be able to finish a limited number of tasks in any given time interval. The existence of such constraints can help us identify problematic data that violate them. The specific bound, can be estimated with the distribution of data.

The second one is data redundancies.

Activity data tend to be highly redundant with several types of event that could be used to measure the quantity of interest. For example, before closing an issue in ITS, we may find a code commit to resolve the issue, code review discussion through the mailing list, or even official announcement on Twitter. This redundancy, as in the information theory, can help us to correct problematic data.

———————-

Now we will introduce our method, method for identifying and correcting problematic data.

—————————

The idea is to exploit capacity constraints and data redundancies for identifying and correcting problematic data. We introduce the method in the context of fixing the task completion time data. the method includes the following steps:

* first, gather events from available sources and link them by tasks and by individuals
* second, choose the primary event type to represent the desired task completion times.
* third, choose a set of redundant event types that approximate the desired task completion times to provide redundancy
* after that, obtain event times for each task and each event type
* and then, Use the distribution of data to identify problematic values. we define the method isProblematic which return the likelihood that the observed value is incorrect.
* and For each task, obtain values of isProblematic(t) for each event type.
* finally, choose from the observations via the following rule: if primary type is good, we use the primary type. otherwise, we choose the redundant type with lowest likelihood of being incorrect.

—————————

To simplify it a little bit, we need to:

* gather data

*	choose primary event type, something like a default choice

*	choose redundant event types, which serve as approximation

*	then we identify problematic data

*	and we correct the problematic data

———-

If we make it event shorter, we can summarise it as five elements:

*	data gathering

*	primary event type

*	redundant event types

*	problematic data identification

*	problematic data correction

———-

OK, so that is our method, which now seems a little too abstract. now, let’s apply it to our task-completion time data.

____

So the first step - data gathering.

Two datasets are used in this paper. The issue tracking data provided by Mozilla in January 2013, and all available code commit data in Mozilla community by February 2014.

____

Next, the primary type. We follow the convention and choose the bug-fix time recorded in issue tracking system as the primary type.

Then we need to determine the redundant event types, the question is what event types can properly approximate the task completion time.
We figure out this by understanding of error mechanisms.

We sample 200 issues that are fixed in top 20 most productive person-days.
We manually check the comments in these issues, and we find three possible error mechanisms listed here.

More importantly, we discovered that, in all of these cases, a better proxy for task completion time appears to be the date of the last comment or the last code commit associated with issues. These two event types, can serve as our redundant event types.

Next step, problematic data identification.

We assume that each point comes from a Poisson distribution where the parameter lambda is determined by the task difficulty, so our observation represents a mixture of Poisson distribution. We further assume that the lambas are weighted by gamma distribution.

Finally, the resulting distribution is a 0-truncated negative binomial distribution. We fit this distribution to our data, and found that under this model, the probability of observing 10 or greater is less than .01%. We therefore take 10 as a bound for capacity constraints. That is, if an individual fixes more than 10 bugs a day, these records are considered to be problematic.

———

Finally, problematic data correction.

As we discussed before, available options for redundant event types include last comment time and last code commit time. Since last code commit time will be used for evaluation later, we use last comment time for correction.
That is, if the fixing time recorded in issue tracking system is problematic, we replace it with last comment time, otherwise we use the recorded fixing time.

————

Then comes the question, does it matter?

———-

We answer the question from two perspective: data accuracy and impacts on research

——

First, Data accuracy.

We found that 16% the issues are fixed with a link pointing to some commits in version control system.

The timestamp of the commit is more likely to reflect the completion of the fix activities as there are no subsequent modifications to the code. We evaluate the accuracy of corrected data by comparing the timestamp with or without proposed correction to the timestamp in version control system.

specifically, we define two error metrics to measure the data accuracy, and result shows that the corrected values are about 50% more accurate than the uncorrected ones.

so the correction of data indeed makes a difference in terms of data accuracy.

———

And second, impacts on research.

In existing research, the task completion time is used to calculate the time until fix, the time it takes to fix a bug.

Prior studies have used various issue report attributes for predicting time until fix， such as summary, severity, priority, product ,description and so forth.

We use a variety of predictors reported in the literature to model the time until fix and reports how the model changes after correction.

———-

This is our model.

______

The result shows that after data correction, we have a better fit, and four predictors switch from being significant or insignificant, which indicates that the correction of data makes a substantial difference when modelling time until fix.

______

Finally, generalisation.

____

If we take a closer look at our method, we fill find that the keystone is to use capacity constraints to identify error, and use data redundancies to correct them, just like what we do in information theory. With error identification and correction, we can improve the quality of data.
This idea, of course, is not restricted to the task completion time.

______-

We conduct similar experiment on two other event types, issue report event and code commit event.

These two types of events are both non-trivial, we are expecting a relative low number of issue reports and code commits finished by each individual on each day.

It turns out that we can find some exceptionally productive individuals based on the two data types, which indicates that we should take a close look at the quality of these data before using them.

________-

next, let’s talk about limitation.

____