how people describe software problems

Andrew J. Ko
Brad A. Myers
Duen Horng Chau

Human-Computer Interaction Institute
Carnegie Mellon University

Last session of the day, so I'll try to keep this short.

This project started with my interest in debugging and mainly with the question, how does software get fixed?
in most large projects, it starts with someone describing a problem in a bug report. These might be users or developers who encountered the problem...

Once the report is written...
people then analyze the reports

sort
search
compare
categorize
assign
triage
prioritize
organize
...

Then, someone takes all of these reports and analyzes them, sorts them, searches, and so on.

All of the labor involved in these tasks takes a great deal of time, and so there's a great potential for automation, or at least semi-automation that helps people get through these tasks more quickly.
can we automate these analyses?

recent approaches have been hindered by the difficulties in processing natural language

Usability discussions in bug reports [Nichols 2005]
Social networks in bug reports [Sandusky 2005]
Analyzing Mozilla reporting process [Antoniol 2004]
Clustering reports by failure [Podgurski 2003]
Tracking features by report [Fischer 2003]
Triaging reports by text classification [Anvik 2006]

yes, but those that have tried have been hindered by the limitations of processing natural language.

all of these use various types of analyses of text in order to try to relate reports to themselves and code.

One alternative is to first understand how people describe software problems. What do these descriptions actually contain?
what do problem descriptions contain?

smooth scrolling inside iframe is not smooth
AccessInterceptor incorrectly matches paths
Export dialog box
Cannot disable JavaScript with new UI
Flag allowLinking should be documented

For example, consider these set of descriptions.

smooth scrolling inside iframe is not smooth
accessInterceptor incorrectly matches paths

how can we categorize these descriptions?
what do problem descriptions contain?

- smooth scrolling inside iframe is not smooth
- AccessInterceptor incorrectly matches paths
- Export dialog box
- Cannot disable JavaScript with new UI
- Flag allowLinking should be documented

what are the major components of each description?
what do problem descriptions contain?

- smooth scrolling inside iframe is not smooth
- AccessInterceptor incorrectly matches paths
- Export dialog box
- Cannot disable JavaScript with new UI
- Flag allowLinking should be documented

how do people use English to form these descriptions?
what do problem descriptions contain?

smooth scrolling inside iframe is not smooth
AccessInterceptor incorrectly matches paths
Export dialog box
Cannot disable JavaScript with new UI
Flag allowLinking should be documented

of all of the questions we could ask, we focused on two in particular
we had two linguistic questions

how does the use of various parts of speech in software problem descriptions compare to common usage of English?

what roles do these parts of speech play in identifying problems?

(read the two questions)

by doing this, we hoped to identify some new ways that might analyze bug reports

we might also find some guidelines for redesigning bug report forms.

here's what we did.
we studied five project’s bug reports

- linux kernel: 5,916 reports
- apache: 1,234 reports
- firefox: 37,952 reports
- openoffice: 38,325 reports
- eclipse: 90,424 reports
  total: 187,851 reports

we downloaded bug reports from five open source projects, totaling nearly 200,000 reports.

these were acquired in January of 2006.

from these reports, we extracted the report titles...
we focused on report titles

find toolbar unexpectedly pops up
“firewire device not found”
blue screen with saa7134 tv tuner
saving a webpage as “Complete” messes with some tags
Line backgrounds and ‘highlight current line’
Performance: slow object effects on solaris 9

these were essentially summaries of the problem or request that the report described.

for example...find toolbar unexpectedly pops up,“firewire device not found”

as you can see, some quoted error messages, others indicated qualities like “performance”.

we performed TWO types of analyses on these titles.
we automatically tagged parts of speech using the most accurate *part of speech tagger* available [Toutanova 2003]

```
noun       det noun       prep adjective
saving a webpage as “Complete”

verb       prep       adjective noun
messes with some tags
```

we automatically tagged different parts of speech using a part of speech tagger

it wasn’t entirely accurate, because the tagger was trained on newspaper articles and not brief sentence fragments like these.

this allowed us to count occurrences of various types.

they also supported our manual exploration of the data.
we manually classified samples of titles generating descriptive categories, such as explicit and implicit indications of correctness

Port specified in doc is incorrect
tables do not import correctly from MS word
file Downloads size wrong

Export to PDF has no options
Apache using 100% cpu
PNG icons are not transparent

we wrote a querying tool that helped us sample and explore the data set, ultimately helping us manually classified titles based on trends that we found.

For example, if we were looking for categories of phrasings of incorrectness, we might categorize these three as explicit statements of correctness and these other three as implicit statements of correctness.

overall, our sample used a small subset of English
Our sample

<table>
<thead>
<tr>
<th>Type</th>
<th>English</th>
<th>Our Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>common nouns</td>
<td>85,738</td>
<td>4000</td>
</tr>
<tr>
<td>proper nouns</td>
<td>?</td>
<td>78,181</td>
</tr>
<tr>
<td>adjectives</td>
<td>42,869</td>
<td>17,034</td>
</tr>
<tr>
<td>verbs</td>
<td>24,496</td>
<td>14,241</td>
</tr>
<tr>
<td>conj., prep., etc.</td>
<td>18,373</td>
<td>1,751</td>
</tr>
<tr>
<td>unique words</td>
<td>171,476</td>
<td>123,417</td>
</tr>
</tbody>
</table>

(likely includes misspellings and misclassifications)

There were very few common nouns,

But there was a wide variety of proper nouns (mostly names of components and other identifiers)

About a quarter of the adjectives in English were used and over half of the verbs, and only a tenth of the conjunctions and prepositions

Names are very important and significant proportion of the content in the report titles.
Does it follow Zipf’s law, which states that the frequency of a word is inversely proportional to its rank?

Generally, yes, but the decline in frequency is slow. It goes from 2.5% to 2%.

The interesting thing is that the frequent words varied from normal English usage.
we looked at the top 25 words in our data and the top 25 words in more common sources of English and took the set difference.

what we see is is ten words that were uniquely common to the bug report titles.

errors, of course, comes to the top, but also words that have to do with what programs should and should not be doing; and the situations when they should be doing it.

we also see a few common data types (file, pages, and text)

now let’s investigate each part of speech a bit further
we look at nouns using regular expressions, finding several categories of all of the nouns that were used in all of the titles (other than misspellings)

54% represented names, acronyms and version numbers...
we also looked at noun phrases, combination of words that identified some set of entities.

we categorized all of the noun phrases in a sample of 1000 titles and found several large categories.

26% of noun phrases identified the visual manifestation of some program functionality. For example...

other smaller categories are described in the paper.
how many names for one thing?

the find dialog in Eclipse was called null pointer exceptions were called

find dialog null pointer exception
find NullPointerException
“Find...” NPE
find command null pointer
finding null pointer Ex
find command to find null pointer error
find/replace npexception
null exception
find/replace nil exception

we also looked at was the variety in naming an entity

for example, the very same dialog in eclipse was identified in numerous ways

the same was true for null pointer exceptions thrown in java applications. Even when there is an official name for some entity, people did not use it.
common verbs were computational

25 verbs (and their conjugations) accounted for 20% of the verbs used

add  use  create  hang  display
open set load install appear
build select get click move
update show save try crash
find remove run change freeze

now moving to verbs

the top 25 verbs and their conjugations accounted for 20% of the verbs used. These were words like add, open, update, find, use, select. Things that commonly happen in modern UIs

the thing to notice about all of these is how they all refer to things that the user is doing or things that the computer is doing.

and this was despite the fact that these descriptions were about some applications that didn’t have graphical user interfaces, such as the Linux kernal and Apache server
Based on 100 descriptions, verbs showed variation in intent.

<table>
<thead>
<tr>
<th>Type</th>
<th>Percentage</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declarative</td>
<td>69%</td>
<td>UI <em>is</em> non-responsive</td>
</tr>
<tr>
<td>None</td>
<td>23%</td>
<td>Compile failure in driver/scsi</td>
</tr>
<tr>
<td>Imperative</td>
<td>5%</td>
<td>Add descriptions for errors</td>
</tr>
<tr>
<td>Subjunctive</td>
<td>3%</td>
<td>Introduce param <em>should</em> work for locals</td>
</tr>
</tbody>
</table>

Certain verbs helped to show the intent of the writer. We took a random sample of 100 of these and categorized them by intent.

Most of the titles were phrased as statements of fact in a declarative way, or with no verb at all.

About 5% of titles were phrased as an imperative “add this or fix this”, and 3% in as a desire way, as in “this should do this…”

Moving on to adjectives and adverbs…
adverbs and adjectives had two roles

classify the problem described
method completion doesn’t always trigger
dialog too big

improper warning for varargs argument

classify the subject of the report
unable to search for deeply overridden method

-crash when loading multiple documents
same file appears twice

these two parts of speech had two major roles.
the first was to aid in characterizing the problem. for example, the problem is that it doesn’t always trigger or that the dialog is too big...

the second was to characterize the subject of the report. For example, the subjects here include deeply overridden methods, multiple documents, or same files

when we look at conjunctions and prepositions...
conjunctions and prepositions

provided the top level structure of a sentence

after a few months of use,

downloading or saving of files or pictures from web pages results in

slowing of mozilla

to the point of

computer lockup for up to 30 seconds or more.

one of the first things that becomes obvious is the that they helped structure the descriptions

for example, here is the title that had the most number of conjunctions and prepositions, broken up by some of its top level structure.

(read the title)

each of these conjunctions and prepositions helped divide up the sentence into meaningful phrases

conjunctions and prepositions also had two specific roles
conjunctions and prepositions helped attribute properties to an entity
doesn't save files with long names
panic when copying file from cdrom
helped describe the context of a problem
image shrinks when protecting sheet
freezes while opening this document

the first was

1) to help attribute properties to some entity

for example, just like with adjectives and adverbs, it indicate a
subset of a certain type. files with long names, file from

2) conjunctions and prepositions also helped describe the
context of a problem.

for example, problem occurs when protecting sheet or while
opening document.

(pause)
better forms

there were four components to a description

toolbar tooltips take too long to appear when hovering

entity/behavior toolbar tooltips
quality usability
problem slow
context when hovering

for one, they help us design better report forms.

if we look at all of the roles and the content in the titles, there are five major components

1) what was exhibiting a failure
2) what was the quality the failure regarded (usability, performance, etc.)
3) what was the particular failure of that quality
4) what was the context of the failure

if we designed report forms around these four components, we would be able to get a more structured form that matches the way people describe these problems that matches the structure that people use to describe software problems.
titles were not grammatical, so natural language “chunkers” are not effective. Conjunctions and prepositions indicate the structure of a bug report title. Our prototype parser achieves 89% accuracy at identifying the top level node of titles.

is to parse the titles.

because the titles aren’t grammatical, modern natural language parsers aren’t effective.

but we did find that conjunctions and prepositions help indicate the structure.

so we looked at the usage of each conjunction and preposition and created a parser based on a list of ranked rules. and it achieved 89% accuracy at identifying the top level node of titles.
each title indicates a quality

each title referred to some quality attribute like *usability*, *performance*, or *consistency*

can we identify words that correspond to these qualities?

can we use these to classify reports by quality, helping to triage and prioritize?

because each report referred to some quality attribute. sometimes this was implicit, sometimes this was explicit.

tools should be able to identify these words and use them to classify reports by the quality attribute, helping project leaders to triage reports based on the project’s goals.

for example, if we could identify all of the adjectives and adverbs that are related to usability, we could filter a major subset of a project’s reports
some limitations include...

- did not study “closed” source projects, which may have different report standards
- we do not know if report authors were users or developers
- the data excludes some “in the moment” problems that developers did not report

there were several limitations of our study.

for example, we didn’t study closed source projects. these might have stricter standards on the quality of the reports, or specific requirements about the structure of the descriptions.

we don’t know who the reporters of our data set were. they could have all been developers. end users might describe problems in a substantively different way.

even if they were developers, our data likely excludes many “in the moment” problems that developers did not write reports about. these may have different from the descriptions that we studied.
major results include...

bug report titles use a *small* subset of English and a *large* set of proper names

major components of report titles include the *subject*, *quality*, *problem*, and *context*

people identify the subject of the report by referring to *output*, *user actions* and *devices*

just to summarize the results...

I hope that these results and the others will help inform the design of automated analyses of bug reports.

If you’d like to download the reports and our querying tool, you can find them at this URL.
questions?

http://www.cs.cmu.edu/~marmalade/reports.html

this work was supported by the National Science Foundation under NSF grant IIS-0329090, by the EUSES consortium under NSF grant ITR CCR-0324770 and by an NDSEG fellowship.

thank you for your time and I'd be happy to answer any questions.