The Software Engineering's Way to Artificial Intelligence

Qianxiang Wang

www.huawei.com

October 29, 2017
UIUC
International Workshop on Intelligent Software Engineering (WISE 2017)
co-located with ASE 2017

HUAWEI

Content

1. Understanding the Intelligent Software Engineering
2. Some Explorations for Intelligent Software Engineering
3. The Software Engineering's Way to Artificial Intelligence

1. Understanding the Intelligent Software Engineering(ISE)

An "intelligent" View of ASE history

1983 Kestrel Institute hosted a Workshop culminating in the publication of the Knowledge-Based Software Engineering Assistant (KBSA) report

1986 In 1986 Rome Laboratories initiated the Knowledge-Based Software Assistance (KBSA) conference

The KBSA report called for radically improving software engineering by formalizing the knowledge for all phases of software engineering. It envisioned three stages of knowledge-based assistance: first, recording decisions and activities made during a software engineering project – a software project knowledge base; second, providing automated inference capability for explaining a software system and supporting maintenance; and finally goal-oriented planning to advise strategies and automate tedious tasks. At the time of the KBSA report, AI was enjoying its first practical successes with diagnostic expert systems, based on encapsulating heuristic expert knowledge as rules manipulated by inference engines. Before KBSA, Rome Labs had investigated the possibility of diagnostic expert systems for software maintenance, and concluded that it was not possible at the time due to the lack of formalized knowledge for software. The KBSA report addressed this gap within the scope of a larger vision.

1991 Knowledge-Based Software Engineering (KBSE)

The conference was renamed to the Knowledge-Based Software Engineering conference to signify opening to a general audience, growing by twenty percent over KBSA-5 with representation from nine countries. Barry Boehm's plenary

1997 Automated Software Engineering (ASE)

Memory". At the close, the chairs of the upcoming 1997 conference led a discussion on the future of KBSE, leading to the decision to expand the scope of the conference and changing the name to its present "Automated Software Engineering".

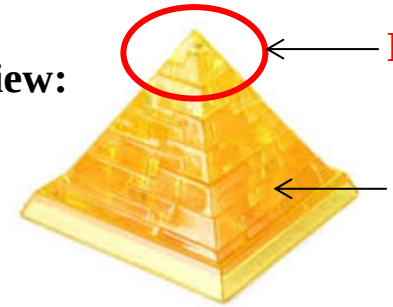
2013 The Second International Workshop on Software Mining

The Second International Workshop on Software Mining aims to bridge research in the data mining community and software engineering community by providing an open and interactive forum for researchers

2017 International Workshop on Intelligent Software Engineering 2017 (WISE 2017)

Automatic Vs Intelligent

Behavioral View:



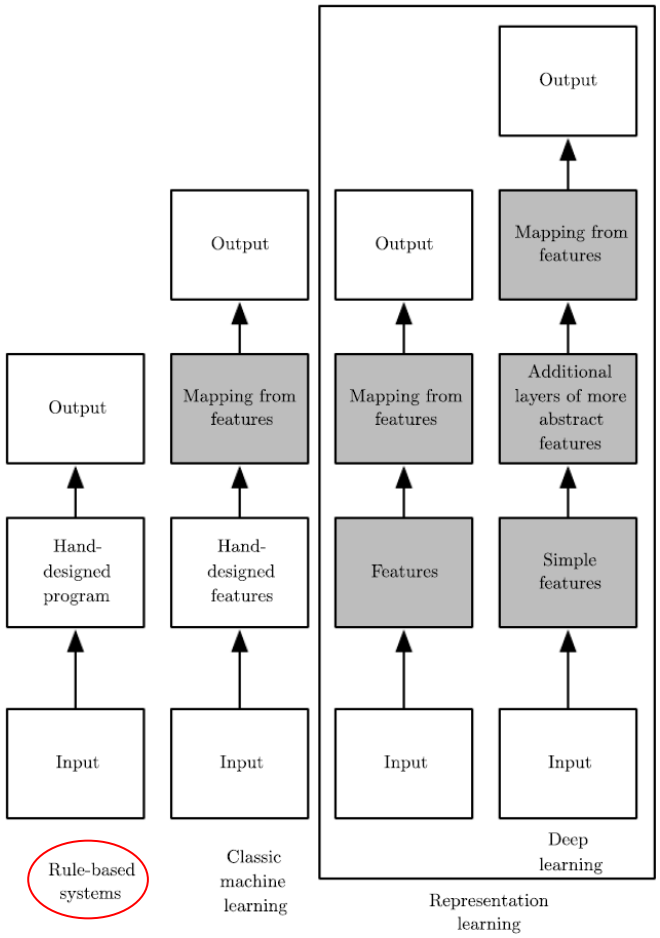
Intelligent Technology

Automatic Technology

Structural View:

- Traditional automatic technology focuses on rules
  Expressed as event sequence(If *, then *), formed some function
  Suitable for problems with certainty: sorting numbers, etc.
  Defect: when function changed, even a little,
  rules needs to be adjusted manually
- Current AI technology (e.g., deep learning) focuses on data
  Expressed as parameters of model, also formed some function
  Parameter can be adjusted automatically by the training data
  Defect : no clear features, difficult to explanation
- Future AI: rule + data ?

AI： symbolic way Vs statistical way



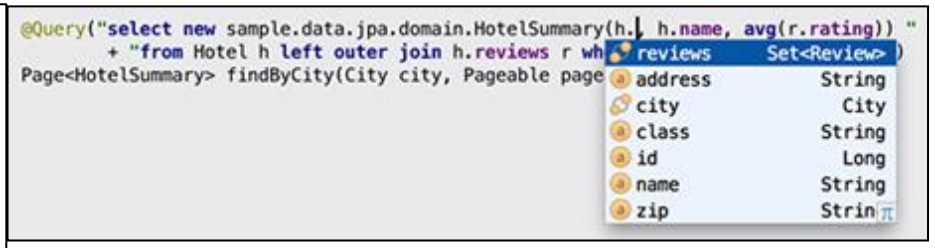
Uncertainty/probability in Software Engineering:
runtime platform, user behaviors, fixing some bug, coding for some task, etc.
Intelligent Software Engineering: solving uncertain SE problems with rules and data for rules

2. Some Explorations for Intelligent Software Engineering: JetBrains for Productivity

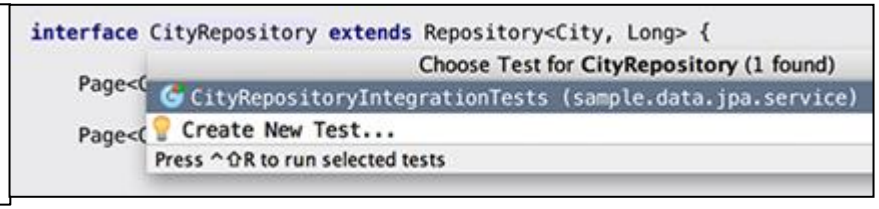
IntelliJ IDEA

1) Smart code completion
While the basic completion suggests names of classes, methods, fields, and keywords within the visibility scope, the smart completion suggests only those types that are expected in the current context.

2) Framework-specific assistance
While IntelliJ IDEA is an IDE for Java, it also understands and provides intelligent coding assistance for a large variety of other languages such as SQL, JavaScript, etc., even when the language expression is injected into a String literal in your Java code.



3) Productivity boosters
The IDE predicts your needs and automates the tedious and repetitive development tasks so you can stay focused on the big picture.



4) Developer ergonomics
In every design and implementation decision that we make, we keep in mind the risk of interrupting the developer's flow and do best to eliminate or minimize it. The IDE follows your context and brings up the corresponding tools automatically.

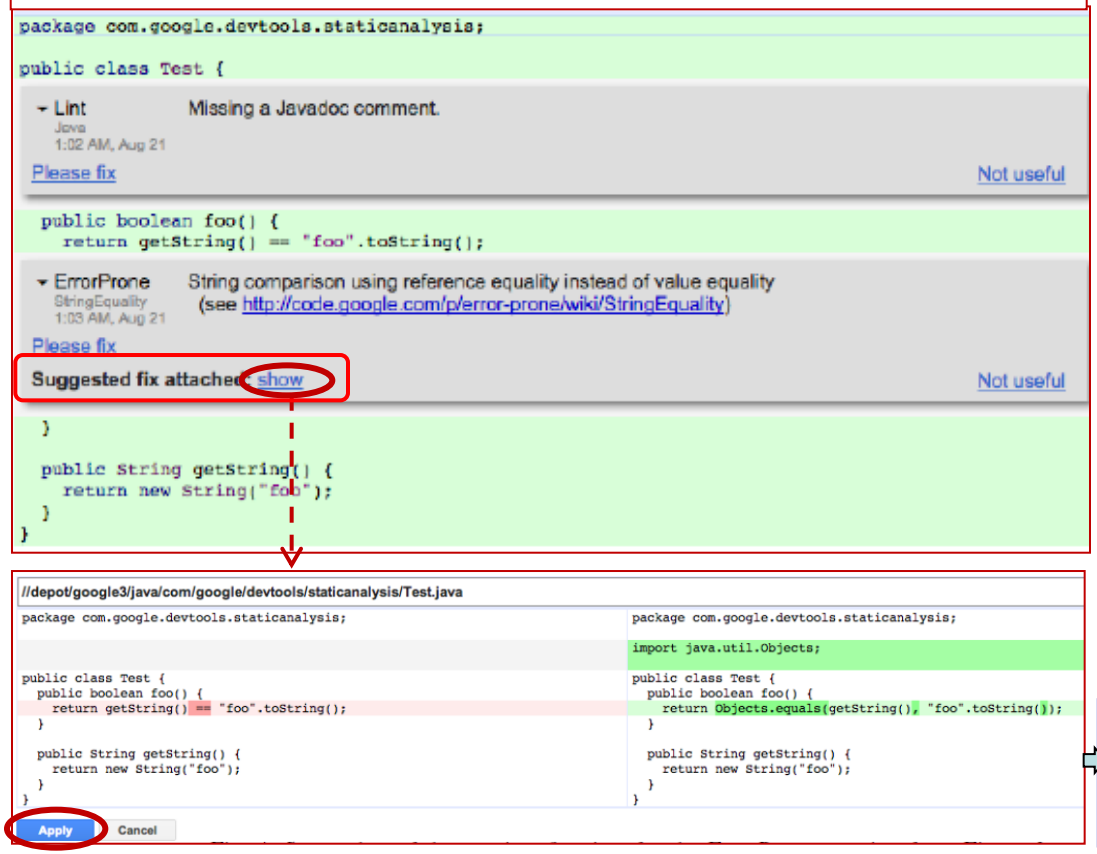
5) Unobtrusive intelligence
The coding assistance in IntelliJ IDEA is not about only the editor: it helps you stay productive when dealing with its other parts as well: e.g. filling a field, searching over a list of elements; accessing a tool window; or toggling for a setting, etc.

rule + data

2. Some Explorations for Intelligent Software Engineering: Google for Quality

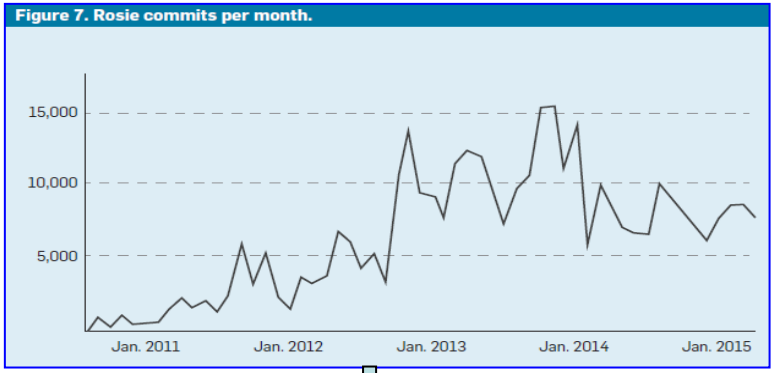
Tricorder

1) A program analysis platform aimed at building a data-driven ecosystem around program analysis
2) Analysis results appear in code review as robot comments (robocomments for short), using the code review tool's commenting system.
3) The analysis writer must show progress toward addressing the issue : false positive higher than 10% threshold. If the rate goes above 25%, we may decide to turn the analyzer off immediately



Rosie(Robot Maid)

1) Maintain the health of the codebase, by some code-cleanup changes.
2) With Rosie, developers create a large patch. Rosie then takes care of splitting the large patch into smaller patches, testing them independently, sending them out for code review, and committing them automatically once they pass tests and a code review.
3) Two related refactoring tools:
Refaster（Java）
ClangMR（C++）



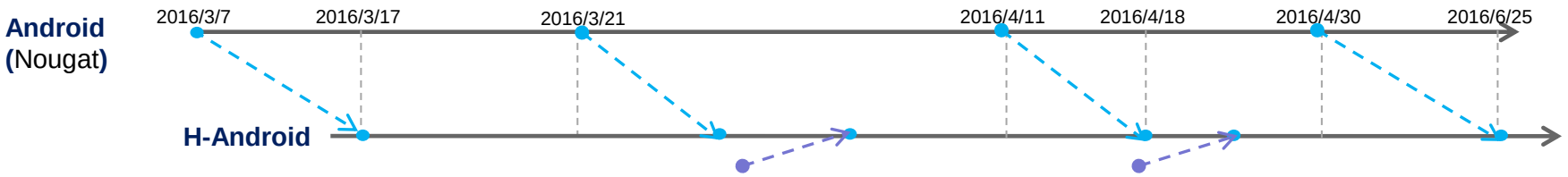
Why Google stores billions of lines of code in a single repository, Rachel Potvin, Josh Levenberg, Communications of the ACM, Volume 59 Issue 7, July 2016

Tricorder: Building a Program Analysis Ecosystem, Caitlin Sadowski, Jeffrey van Gogh, Ciera Jaspan, Emma Soederberg, Collin Winter, *International Conference on Software Engineering (ICSE)* (2015)

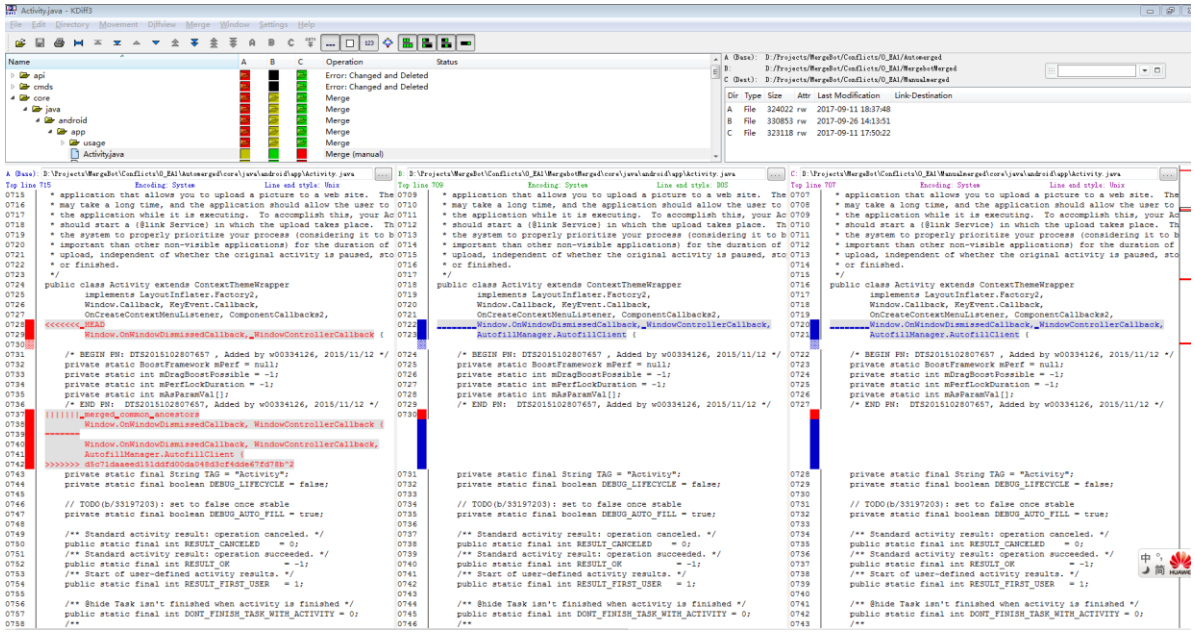
rule + data

2. Some Explorations for Intelligent Software Engineering: HUAWEI for Cooperation

Issue: Different teams need to cooperate for the same task, and may change the same code chunk.



- **Git-merge is widely used to merge code from different teams, and report lots of merge confliction.**
- **We developed tools to reduce the merge confliction which is based on lots of rules by mining plenty of manual merged code, and can solve about 50% conflictions reported by git-merge.**



Git-merge reported confliction | Merged code from MergeBot | Manual Merged code

Other works involved:

1) Inferring code change intention, so as to help to make merge decision easily.

2) Detect interface migration, and update involved APKs.

3) Detect code that aim to lead to confliction, which should not permitted to commit .

4) Diagnose runtime problems(for example, bad response time) caused by upgrading, try to locate the defect code and repair it.

rule + data

2. Some Explorations for Intelligent Software Engineering: Academic Researches

Interesting works that are not widely used by programmers so far

1) Series work on Program Synthesis (three dimentions), Sumit Gulwani
- The user intent can be expressed in various forms including logical specification, examples, traces, natural language, partial programs, or even related programs.
- The search space should strike a good balance between expressiveness and efficiency
- The search technique can be based on enumerative search, deduction, constraint solving, statistical techniques, or some combination of these.

2) DEEPCODER: LEARNING TO WRITE PROGRAMS, Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, Daniel Tarlow, ICLR(International Conference on Learning Representations) 2017.

3) Google DeepMind: Neural Programmer-Interpreters, Scott Reed, Nando de Freitas, ICLR 2016.

.....

CCF TCSE(Technical Committee of Software Engineering, China Computer Federation)

1) CodeNet, which is inspired by ImageNet, “an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Near 1 million nodes so far”.

2) CodeNet Challenge, which is similar with ImageNet Challenge
This year, code defect (vulnerability) detection, 8 contestants have entered the final stage
Will be held on November 3, 2017

rule + data

3. The Software Engineering's Way to Artificial Intelligence

Current hot AI technology (Neural Network) is good at Perceptual Intelligence, e.g., picture recognition, voice recognition.
The ingredients inside the target is loosely coupled, with few structure information.

But the code is the expression of precise logic, well structured, and difficult to be modeled by Neural Network:
"Are Deep Neural Networks the Best Choice for Modeling Source Code", Prem Devanbu, ICSE 2017.

Two main challenges for software engineering, compared with other fields that Neural Network are well applied:

1. Challenge about judgement
   Picture recognition, voice recognition, chess, … , all the recognition results are easy to judgement.
   How to judge the correctness of the code snippet? (program halt issue is undecidable)
2. Challenge about understanding
   understanding the code: meaning of the code, meaning of code changing
   understanding the document: by code

If someone thinks that the deep Learning has not been well applied in software engineering, please refer the following paper:
"Will software engineering ever be engineering?" Michael Davis, Communications of the ACM, Volume 54 Issue 11, November 2011.
"*Whether or not software engineers do any engineering, engineers increasingly engage in activities that look like software engineering*"

3. The Software Engineering's Way to Artificial Intelligence

Three phases of intelligence:

- Computational intelligence, defined by Turing Machine
- Perceptual Intelligence, defined by Neural Network
- Cognitive intelligence, defined by what?

Conjecture: will software engineers give the definition of cognitive intelligence?

Intelligent features of software engineering should simulate human's cognitive process

- Mining rules from well prepared data
- Confirming the candidate rules
- Adjusting rules according to new data

It should be: rule/knowledge centric: simple or complex
context sensitive: surrounding, environment
human-machine cooperative: user feedback, developer confirm

Open issues:

1) How to get the primary rules from data with a general way?
   Specific data mining technology for ISE?
2) How to adjust rules with new data?
3) Paradigms for 1) and 2)

Thanks!