Automatic Extraction of Developer Expertise

Cédric Teyton  Marc Palyart  Jean-Rémy Falleri  Floréal Morandat  Xavier Blanc

SoftwareEngineering@LaBRI, Bordeaux, France
Software developers & Experts

• Software development involve teams of developers who have their own set of skills and knowledge

• Experts are developers with a strong knowledge in a particular domain or technology:
  – A programming language
    - R
    - Groovy
    - Scala
  – A library or framework
    - jQuery
    - JUnit
Experts are needed...

• ...In a software project
  - Who master the core module?
  - Who touched the most Ruby scripts last month?

• ...In open-source communities
  - Who can write unit tests using mock objects?

• ... In the "crowd"
  - Who can help me with this technology?
  - Who can review my code to fix an issue?

Challenge: How to identify experts?
What is difficult with finding experts? (1)

Express what we exactly look for, the skill. Ex:

1) Know the **Scala** programming language. What does it mean "to know a language"?
   - Have **Scala** written on your personal resume?
   - Know all the language constructs?
   - Edit 1,000+ of files?

2) Know the **JUnit** library:
   - Be aware of all methods from the API?
   - Create 100+ of @Test methods?
   - Use a particular subset API?
What is difficult with finding experts? (2)

Collect and maintain developer's skills and expertise:

1. Self-evaluation is too subjective and time-consuming ("Which level do you have w.r.t. this skill?")

2. How to keep information up to date? What if a developer tame a new skill?

Developers must not intervene: full automation is required!
What implies full automation?

- Developers are evaluated from their actions
  - Thomas the Apostle: "I only believe what I observe"

- Expertise = skill experience = |skill expressions|

**Hypothesis**: Experience is correlated with expertise

Idea: Assign level of experience to developers w.r.t. a skill
Goals & Challenges: Expert Identification

1) Skills specification
   - Manual operation
   - Simple is better: define a language
   - Flexible, must support a diversity of skills
   - Users only have to care about this part

2) Skills extraction
   - Automatic operation
   - Input: data sources + skills
   - Output: reporting, scores, ...

Specification files

Data sources

Tooling

Reporting, Exploitation
Our Solution (1) : X Tic

Extract skills from Version Control Systems

Searched skills at Atomic File Change granularity

Delta computation : compare with previous version

Isolate contributions
Our Solution (2) : X Tic

Extract contribution by Delta Analysis

1. Compute tree differencing using a language-specific parser (ex : JDT for Java)
2. Produce a annotated representation of the file where the new elements are enhanced

Example

```
//Version v0
class Foo {
    @Test
    void testFoo() {
        assertTrue(true);
    }
}
```

```
<CompilationUnit>
  <TypeDeclaration>
    <SimpleName label="Foo"/>
    <MethodDeclaration added="1">
      <MarkerAnnotation>
        <SimpleName label="Test" added="1">
          <MarkerAnnotation>
            <PrimitiveType label="void" added="1"/>
            <SimpleName label="testFoo" added="1"/>
            <Block added="1">
              ...
            </Block>
          </MarkerAnnotation>
        </SimpleName label="Test" added="1"/>
      </MarkerAnnotation>
    </MethodDeclaration>
  </TypeDeclaration>
</CompilationUnit>
```
Our Solution (3) : XTic

Specification : a Domain-specific language

1) Defines a skill as a set of patterns that can constrain an atomic file change on:
   - The Kind of the AFC (added, edited, removed, renamed)
   - The Path of the file (src/core/.../*.scala)
   - The Content of the file
   - The Abstract Syntax Tree representation of the file

2) Constraints can also concern the parent version
3) Currently based on the XML language
Our Solution (4) : XTic Example

Skill : "add a JUnit Method"

Kind
Path
Content
Tree

Filters act as a chain. When one fails, the search stops.

XPath Expression
New elements are annotated @Added

We are currently limited to XPath capabilities in expressiveness
XTic : A prototype

Repositories URL

Specs. files

Input

Model Extraction

Skills Extraction

Harmony Tool [1]
History, revisions, ...

Browse atomic file changes

XTic : Deployment (1)

Performance : Does XTic Scale ?

→ Stress test on **16** Java open source projects from GitHub
  - [350 - 1300] revisions, [2 - 177] Java KLOC
→ Defined **skills** at a low level of granularity
  - add JUnit method, use Generics, add local variable

→ Average **execution** time : 7 min
→ Most of the time spent on **checking out** files and computing **diffs**

Yes, XTic Scales !
XTic : Deployment (2)

Does XTic produce valid results?
- Experiment with an industrial partner
- 1 repository, 13 developers, 4 years of work, ~4500 rev.

Both rated each developer as Low, Medium, High for skill. 13*8 = 104 values

<table>
<thead>
<tr>
<th></th>
<th>Agree</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>48</td>
<td>56</td>
<td>21</td>
</tr>
<tr>
<td>XTic,</td>
<td>67</td>
<td>37</td>
<td>6</td>
</tr>
<tr>
<td>XTic,</td>
<td>59</td>
<td>45</td>
<td>14</td>
</tr>
</tbody>
</table>

Yes, results are realistic!
XTic : Conclusion

- **XTic**, an approach to extract developers expertise
- Alleviate the burden of work for project managers as well as researchers
- Only requires specifications of skills

Try it! [https://se.labri.fr/xtic/](https://se.labri.fr/xtic/)
Future work

• Specify skills at higher levels than atomic file changes:
  → revision ? (i.e., set of AFC)
  → history ? (i.e., set of revisions)

• Extract skills from other software repositories (bug trackers, mailing lists, ...)

• Merge multiple identities across repositories
Thank you for your attention.

Software Engineering @ LaBRI, Bordeaux, France