Audrey

On Demand Run time Information

Rathesan Iyadurai
6. November 2018

Supervised by Haidar Osman, Boris Spasojević and Oscar Nierstrasz
public PatientCase parse(String line) throws ParseException {
    PatientCase pc = new PatientCase();

    List<String> vars = FIELD_SPLITTER.splitToList(line);

    if (vars.size() != EXPECTED_COLUMNS) {
        throw new ParseException();
    }

    pc.setAge(vars[2]);

    return pc;
}
public PatientCase parse(String line) throws ParseException {
    PatientCase pc = new PatientCase();

    List<String> vars = FIELD_SPLITTER.splitToList(line);

    if (vars.size() != EXPECTED_COLUMNS) {
        throw new ParseException();
    }

    pc.setAge(vars[2]);

    return pc;
}
public PatientCase parse(String line) throws ParseException {
    PatientCase pc = new PatientCase();

    List<String> vars = FIELD_SPLITTER.splitToList(line);

    if (vars.size() != EXPECTED_COLUMNS) {
        throw new ParseException();
    }

    pc.setAge(vars[2]);

    return pc;
}
Questions Programmers Ask During Software Evolution Tasks

Jonathan Sillito, Gail C. Murphy and Kris De Volder
Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada
{sillito,murphy,kdvolder}@cs.ubc.ca
“What are the values of these arguments at runtime?”

Questions Programmers Ask During Software Evolution Tasks

Jonathan Sillito, Gail C. Murphy and Kris De Volder
Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada
{sillito,murphy, kdvolder}@cs.ubc.ca
“What are the values of these arguments at runtime?”

“How does this data structure look at runtime?”
“What are the values of these arguments at runtime?”

“How does this data structure look at runtime?”

“Which run time data leads to throw a particular exception?”
“Can we address these issues with a system that provides run time examples on demand?”

“What are the values of these arguments at runtime?”

“How does this data structure look at runtime?”

“Which run time data leads to throw a particular exception?”
Audrey
On Demand Run time Information in
PharO
Audrey Components
Audrey Components

Instrumentation
Audrey Components

Instrumentation

Storage
Audrey Components

Instrumentation

Storage

Visualization
Problem: Overhead

30 - 50% Instrumentation overhead in Pharo!
Problem: Overhead

30 - 50% Instrumentation overhead in Pharo!

Solution?
Problem: Overhead

30 - 50% Instrumentation overhead in Pharo!

Solution?

Partial instrumentation with temporal sharding
Problem: Overhead

30 - 50% Instrumentation overhead in Pharo!

Solution?

Partial instrumentation with temporal sharding

<table>
<thead>
<tr>
<th>50%</th>
<th>50%</th>
</tr>
</thead>
</table>

% of time spent in your codebase
Problem: Serialization

How do you store the example data so that it is presentable for program comprehension later on?
Problem: Serialization

How do you store the example data so that it is presentable for program comprehension later on?

“haidar|boris”

42

“haidar|boris”

42
Problem: Serialization

How do you store the example data so that it is presentable for program comprehension later on?

```
"haidar|boris"
42
```

arbitrary objects / values

some fancy JSON?

```
"haidar|boris"
42
```
Problem: IDE Infrastructure

How to communicate with the IDE?
Problem: IDE Infrastructure

How to communicate with the IDE?

Solution?

Language Server Protocol

The Language Server Protocol (LSP) defines the protocol used between an editor or IDE and a language server that provides language features like auto complete, go to definition, find all references etc.

https://microsoft.github.io/language-server-protocol/
Problem: Usability
Problem: Usability

Solution?
Usability tests during change tasks
The Perfect Audrey
The Perfect Audrey

Minimal overhead
Instrumentation
The Perfect Audrey

Minimal overhead Instrumentation

Efficient, language-agnostic representation
The Perfect Audrey

Minimal overhead Instrumentation

Efficient, language-agnostic representation

Useful visualizations in multiple IDEs
The Perfect Audrey

Minimal overhead Instrumentation  Efficient, language-agnostic representation  Useful visualizations in multiple IDEs
does not exist (at the end of this project)
Towards a language-agnostic solution
Why GraalVM?

https://www.graalvm.org/docs/why-graal/

Create Language-agnostic Tools

GraalVM provides a framework for creating language-agnostic tools like debuggers, profilers, or other instrumentations. GraalVM provides a standardized way to express and run program code enabling cross-language research and the development of tools that are developed once and then can be applied to any language.
So far...

An argument inspector in Pharo. Why?
- To get a feel for the problem
- Quick prototyping

Next up...

Audrey on Truffle. Why?
- Language-agnostic tooling