International Workshop on Smalltalk Technologies,
Novi Sad, Serbia, 2022

Can We Automatically Generate Class Comments in Pharo?

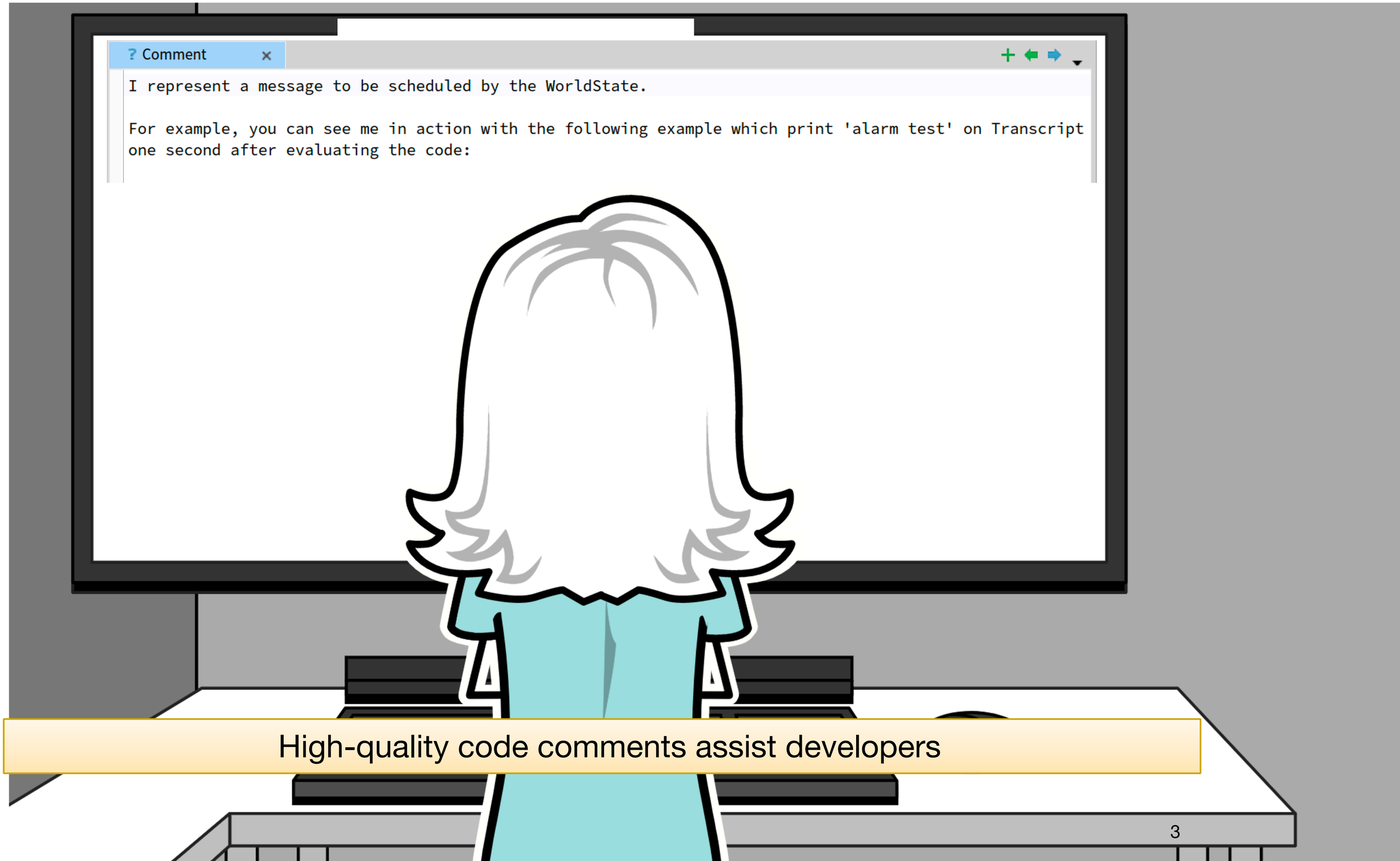
Pooja Rani, Alexandre Bergel,

Lino Hess, Timo Kehrer, Oscar Nierstrasz

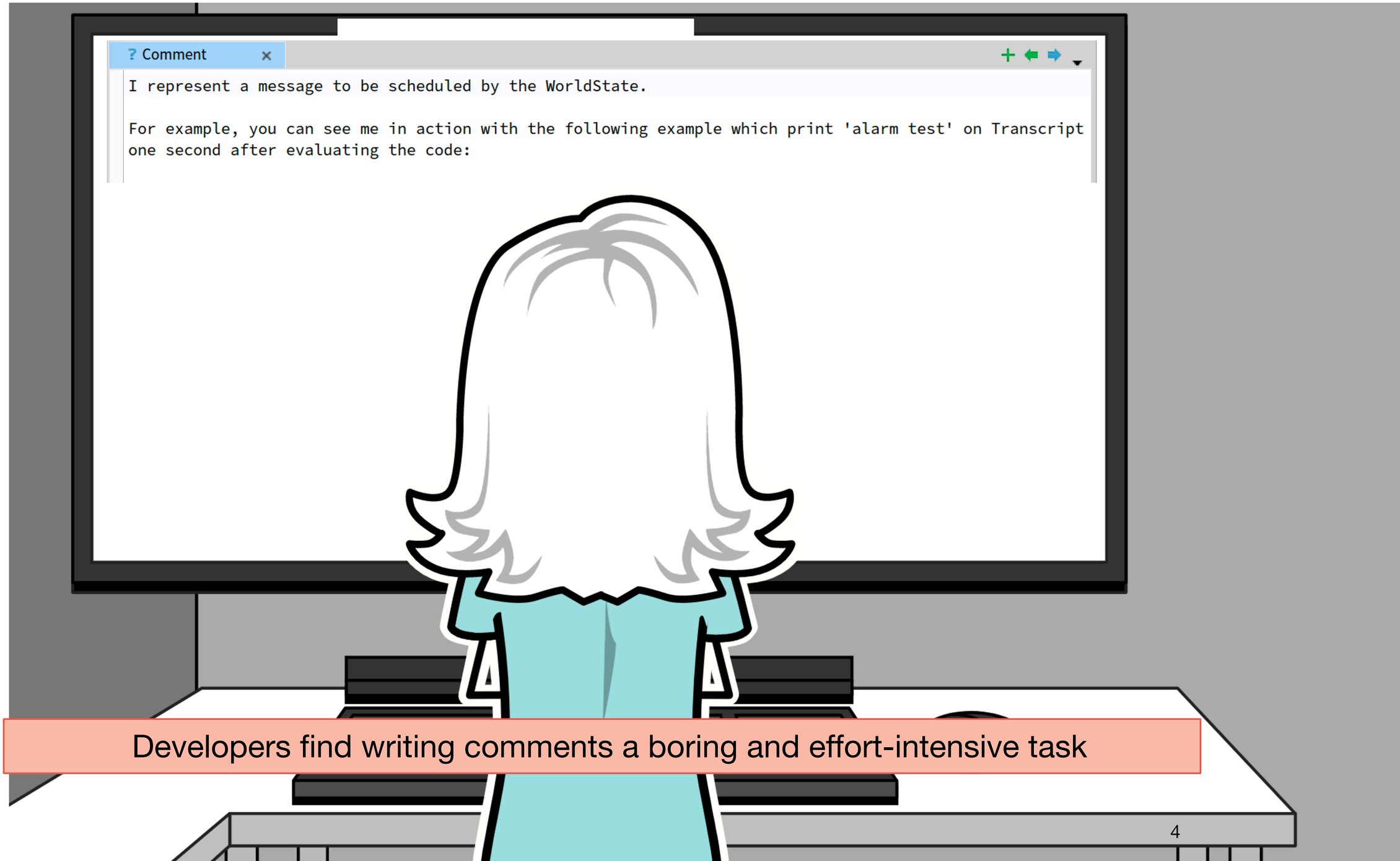
University of Bern, Switzerland

? Comment

```
I represent a message to be scheduled by the WorldState.

For example, you can see me in action with the following example which print 'alarm test' on Transcript
one second after evaluating the code:

Transcript open.
MorphicUIManager currentWorld
        addAlarm: #show:
        withArguments: #('alarm test')
        for: Transcript
        at: (Time millisecondClockValue + 1000).

* Note *
Compared to doing:
[(Delay forMilliseconds: 1000) wait. Transcript show: 'alarm test'] forkAt: Processor activeProcess
priority +1.

the alarm system has several distinctions:
- Runs with the step refresh rate resolution.
- Alarms only run for the active world. (Unless a non-standard scheduler is in use)
- Alarms with the same scheduled time are guaranteed to be executed in the order they were added
```

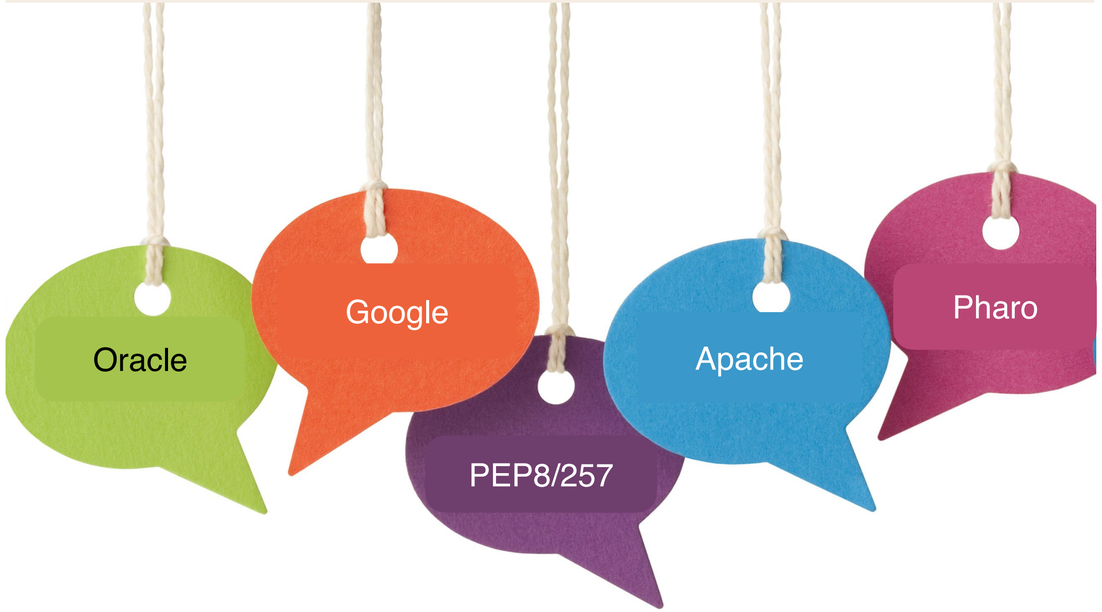


High-quality code comments assist developers

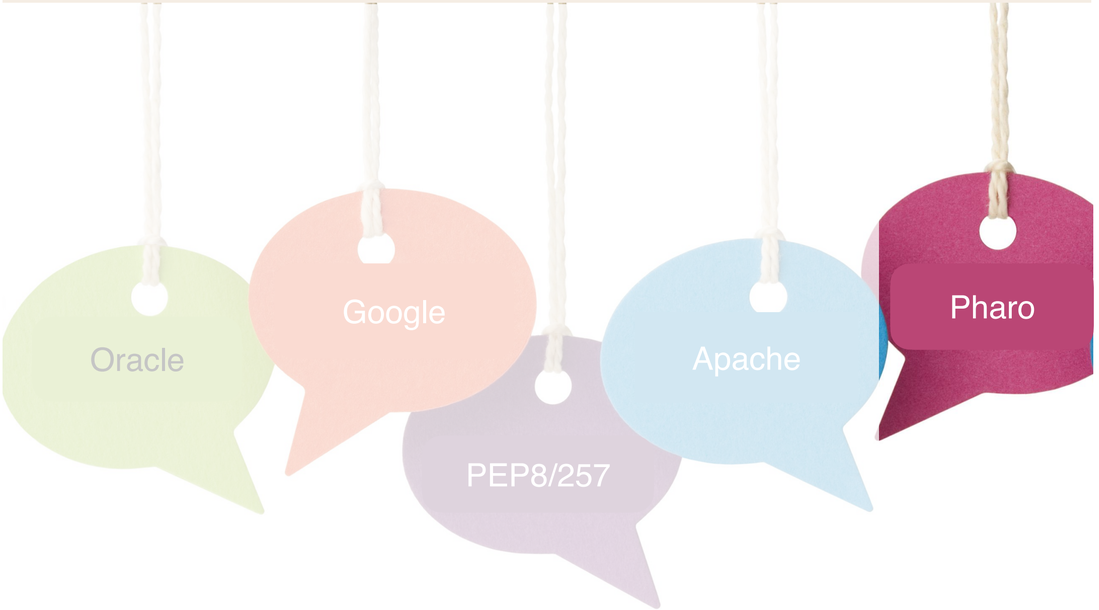


Developers find writing comments a boring and effort-intensive task

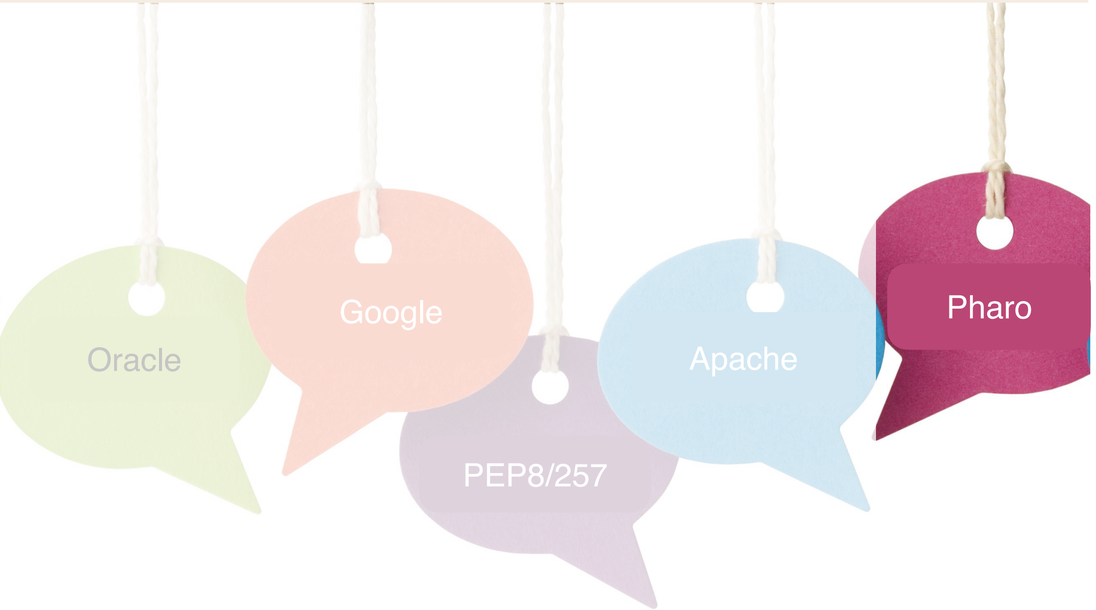
To support developers in writing comments, communities suggest coding guidelines

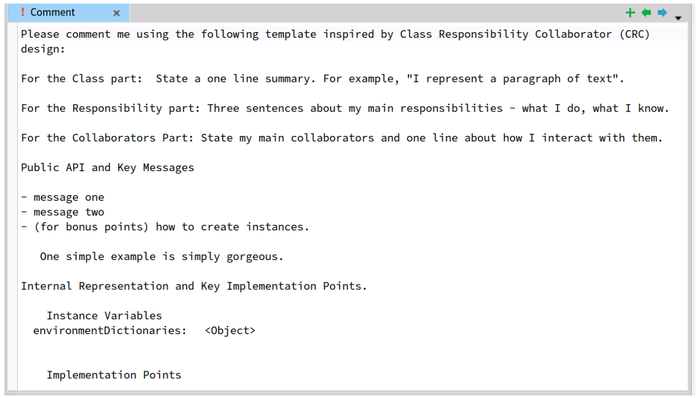


To support developers in writing comments, communities suggest coding guidelines



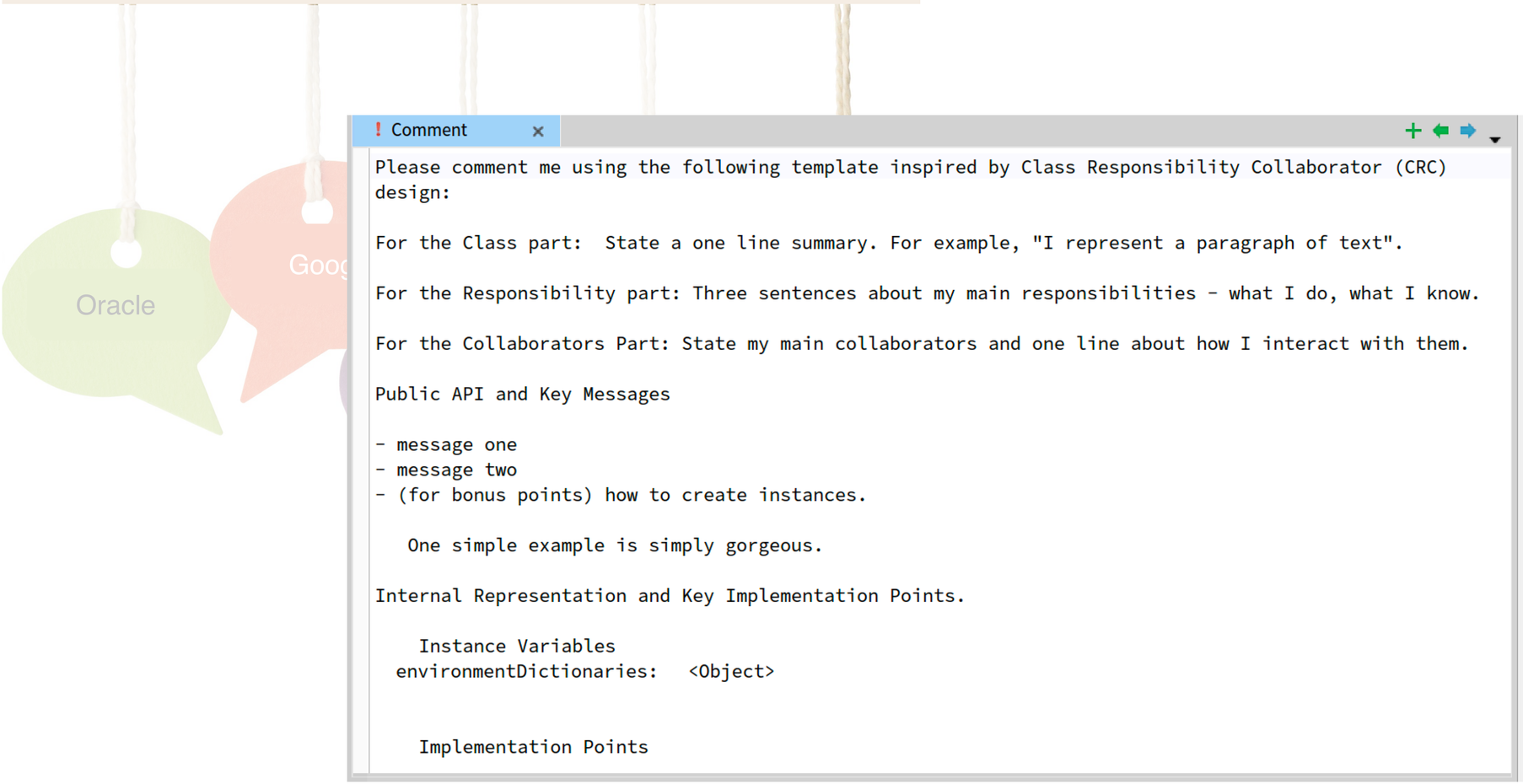
To support developers in writing comments, communities suggest coding guidelines



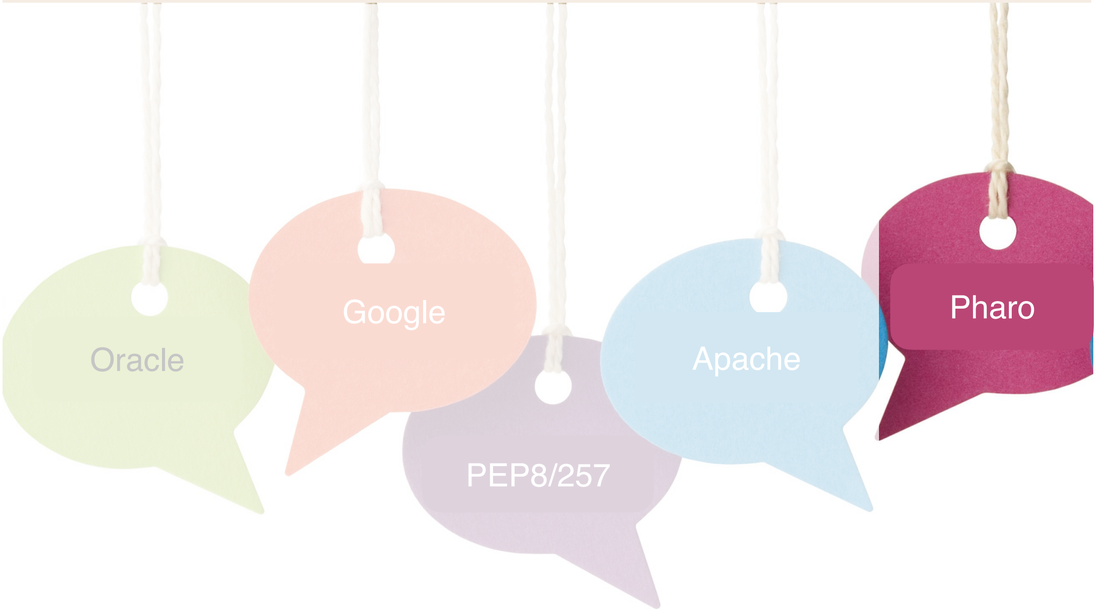


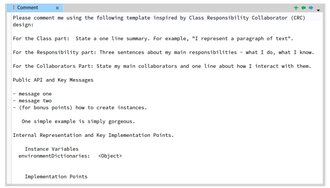
```
Comment
Please comment me using the following template inspired by Class Responsibility Collaborator (CRC)
design:

For the Class part:  State a one line summary. For example, "I represent a paragraph of text".

For the Responsibility part: Three sentences about my main responsibilities - what I do, what I know.

For the Collaborators Part: State my main collaborators and one line about how I interact with them.

Public API and Key Messages

- message one
- message two
- (for bonus points) how to create instances.

   One simple example is simply gorgeous.

Internal Representation and Key Implementation Points.

    Instance Variables
  environmentDictionaries:   <Object>


    Implementation Points
```

To support developers in writing comments, communities suggest coding guidelines



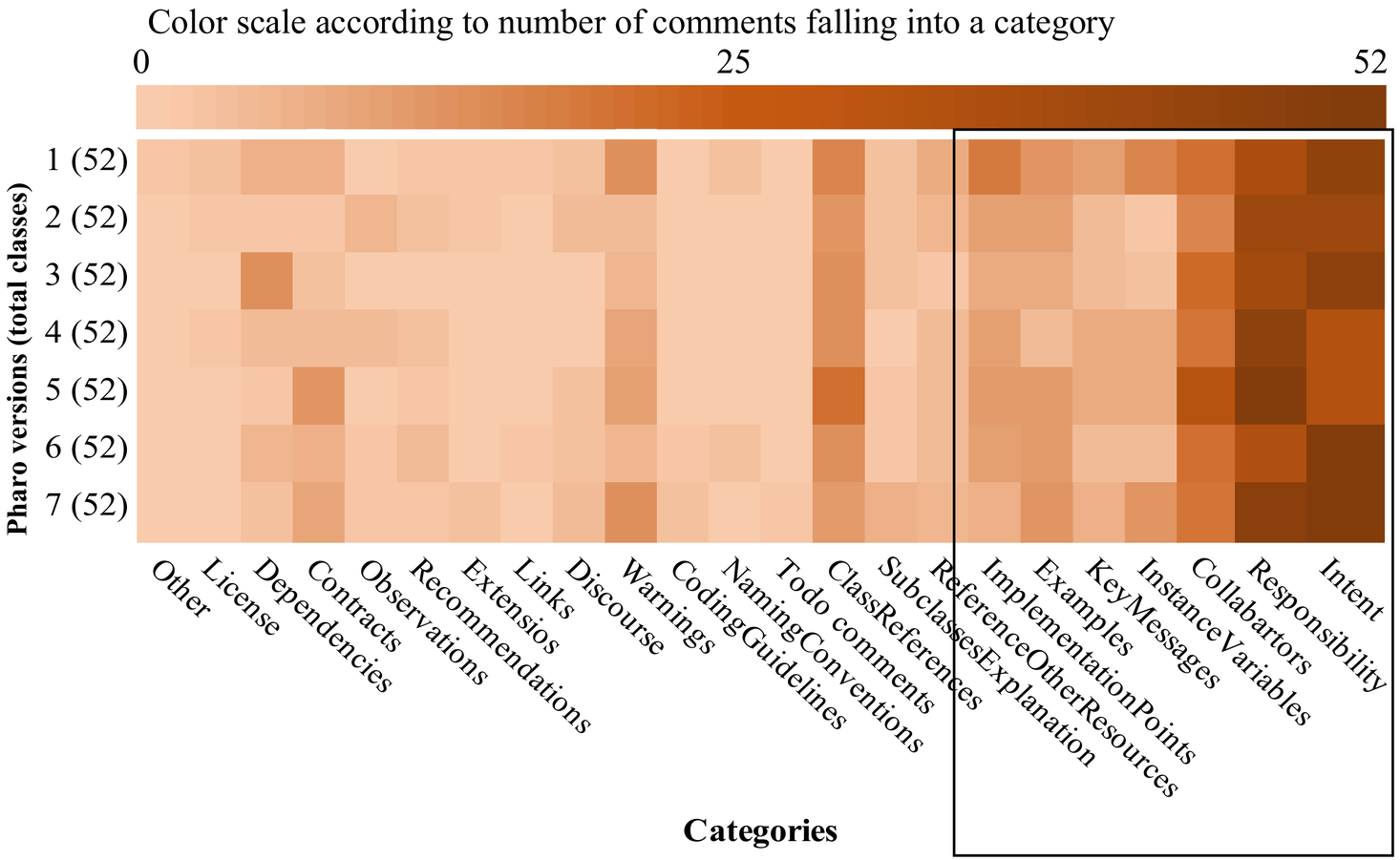
To support developers in writing comments, communities suggest coding guidelines

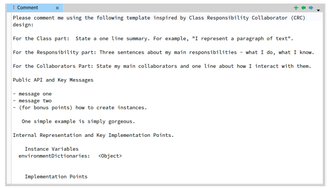




Developers follow **template inspired information types** more often

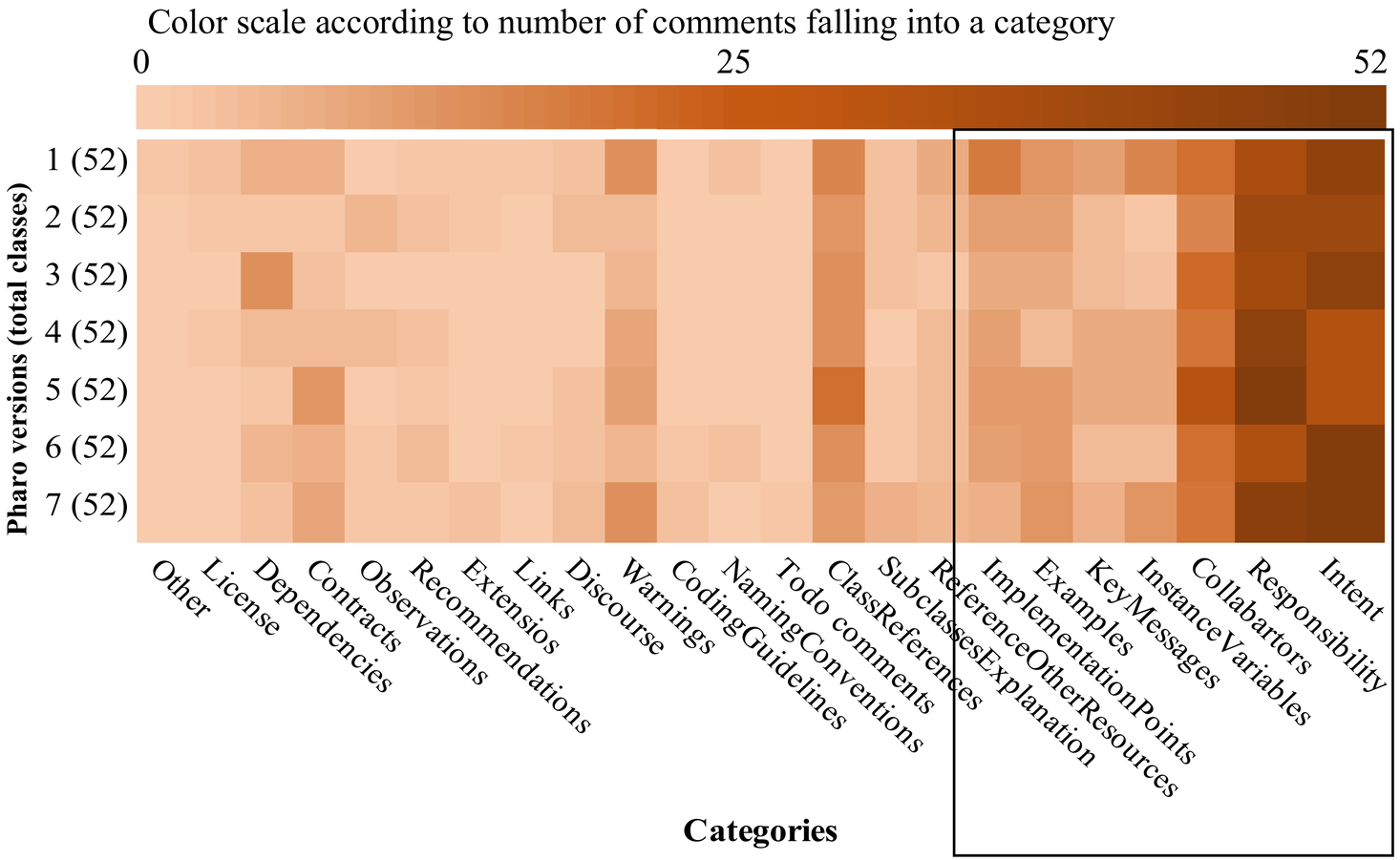
- Rani et al., Empirical Software Engineering, 2021

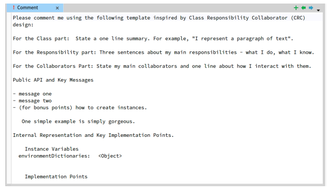




Developers follow **template inspired information types** more often

- Rani et al., Empirical Software Engineering, 2021





Can we automatically generate these information types?

Rebecca Wirfs-Brock

Objects are responsible members of an object community

Object role stereotypes in 1992 in a Smalltalk Report article

Rebecca Wirfs-Brock

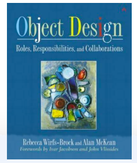
Objects are responsible members of an object community

Object role stereotypes in 1992 in a Smalltalk Report article

Rebecca Wirfs-Brock

Objects are responsible members of an object community

Object role stereotypes in 1992 in a Smalltalk Report article

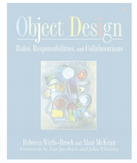


- Controller
- Coordinator
- Structure
- Interface
- Information holder

Objects are responsible me
object community

Object role stereotypes in 1992 in a Sn ak Report article

Stereotype-based approach



- Controller
- Coordinator
- Structure
- Interface
- Information holder

Previous works

Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno[1], Jairo Aponte[2], Giriprasad Sridhara[3], Andrian Marcus[1], Lori Pollock[4], K. Vijay-Shanker[4]

[1]Wayne State University
Detroit, MI, USA
{lmorenoc, amarcus}@wayne.edu

[2]Universidad Nacional de Colombia
Bogotá, Colombia
jhapontem@unal.edu.co

[3]IBM Research India
Bangalore, India
gisridha@in.ibm.com

[4]University of Delaware
Newark, DE, USA
{pollock, vijay}@cis.udel.edu

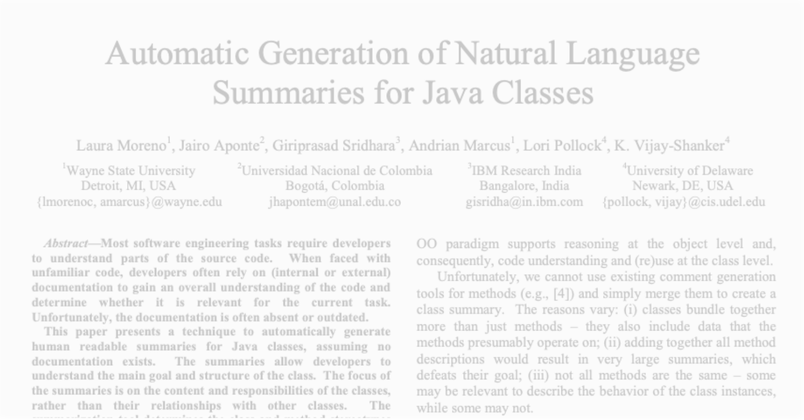
***Abstract*—Most software engineering tasks require developers to understand parts of the source code. When faced with unfamiliar code, developers often rely on (internal or external) documentation to gain an overall understanding of the code and determine whether it is relevant for the current task. Unfortunately, the documentation is often absent or outdated.**

This paper presents a technique to automatically generate human readable summaries for Java classes, assuming no documentation exists. The summaries allow developers to understand the main goal and structure of the class. The focus of the summaries is on the content and responsibilities of the classes, rather than their relationships with other classes. The

OO paradigm supports reasoning at the object level and, consequently, code understanding and (re)use at the class level.

Unfortunately, we cannot use existing comment generation tools for methods (e.g., [4]) and simply merge them to create a class summary. The reasons vary: (i) classes bundle together more than just methods – they also include data that the methods presumably operate on; (ii) adding together all method descriptions would result in very large summaries, which defeats their goal; (iii) not all methods are the same – some may be relevant to describe the behavior of the class instances, while some may not.

Previous works



Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno[1], Jairo Aponte[2], Giriprasad Sridhara[3], Andrian Marcus[1], Lori Pollock[4], K. Vijay-Shanker[4]

[1]Wayne State University
Detroit, MI, USA
{lmorenoc, amarcus}@wayne.edu

[2]Universidad Nacional de Colombia
Bogotá, Colombia
jhapontem@unal.edu.co

[3]IBM Research India
Bangalore, India
gisridha@in.ibm.com

[4]University of Delaware
Newark, DE, USA
{pollock, vijay}@cis.udel.edu

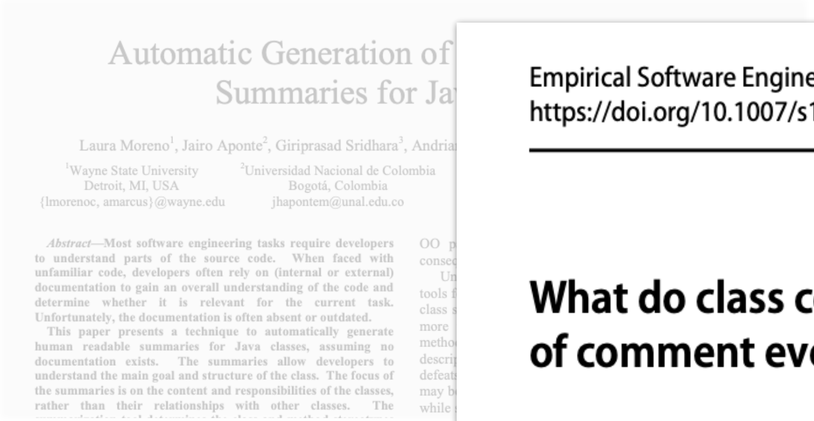
***Abstract*—Most software engineering tasks require developers to understand parts of the source code. When faced with unfamiliar code, developers often rely on (internal or external) documentation to gain an overall understanding of the code and determine whether it is relevant for the current task. Unfortunately, the documentation is often absent or outdated.**

This paper presents a technique to automatically generate human readable summaries for Java classes, assuming no documentation exists. The summaries allow developers to understand the main goal and structure of the class. The focus of the summaries is on the content and responsibilities of the classes, rather than their relationships with other classes. The

OO paradigm supports reasoning at the object level and, consequently, code understanding and (re)use at the class level.

Unfortunately, we cannot use existing comment generation tools for methods (e.g., [4]) and simply merge them to create a class summary. The reasons vary: (i) classes bundle together more than just methods – they also include data that the methods presumably operate on; (ii) adding together all method descriptions would result in very large summaries, which defeats their goal; (iii) not all methods are the same – some may be relevant to describe the behavior of the class instances, while some may not.

Previous works


Automatic Generation of
Summaries for Ja

Laura Moreno[1], Jairo Aponte[2], Giriprasad Sridhara[3], Andria

[1]Wayne State University
Detroit, MI, USA
{lmorenoc, amarcus}@wayne.edu

[2]Universidad Nacional de Colombia
Bogotá, Colombia
jhapontem@unal.edu.co

Abstract—Most software engineering tasks require developers to understand parts of the source code. When faced with unfamiliar code, developers often rely on (internal or external) documentation to gain an overall understanding of the code and determine whether it is relevant for the current task. Unfortunately, the documentation is often absent or outdated.

This paper presents a technique to automatically generate human readable summaries for Java classes, assuming no documentation exists. The summaries allow developers to understand the main goal and structure of the class. The focus of the summaries is on the content and responsibilities of the classes, rather than their relationships with other classes. The


Empirical Software Engineering
https://doi.org/10.1007/s10664-021-09981-5

Check for updates

What do class comments tell us? An investigation of comment evolution and practices in Pharo Smalltalk

Pooja Rani[1] · Sebastiano Panichella[2] · Manuel Leuenberger[1] · Mohammad Ghafari[3] · Oscar Nierstrasz[1]

Previous works

Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno[1], Jairo Aponte[2], Giriprasad Sridhara[3], Andrian Marcus[1], Lori Pollock[4], K. Vijay-Shanker[4]

[1]Wayne State University
Detroit, MI, USA
{lmorenoc, amarcus}@wayne.edu

[2]Universidad Nacional de Colombia
Bogotá, Colombia
jhapontem@unal.edu.co

[3]IBM Research India
Bangalore, India
gisridha@in.ibm.com

[4]University of Delaware
Newark, DE, USA
{pollock, vijay}@cis.udel.edu

***Abstract*—Most software engineering tasks require developers to understand parts of the source code. When faced with unfamiliar code, developers often rely on (internal or external) documentation to gain an overall understanding of the code and determine whether it is relevant for the current task. Unfortunately, the documentation is often absent or outdated.**

This paper presents a technique to automatically generate human readable summaries for Java classes, assuming no documentation exists. The summaries allow developers to understand the main goal and structure of the class. The focus of the summaries is on the content and responsibilities of the classes, rather than their relationships with other classes. The

OO paradigm supports reasoning at the object level and, consequently, code understanding and (re)use at the class level.

Unfortunately, we cannot use existing comment generation tools for methods (e.g., [4]) and simply merge them to create a class summary. The reasons vary: (i) classes bundle together more than just methods – they also include data that the methods presumably operate on; (ii) adding together all method descriptions would result in very large summaries, which defeats their goal; (iii) not all methods are the same – some may be relevant to describe the behavior of the class instances, while some may not.

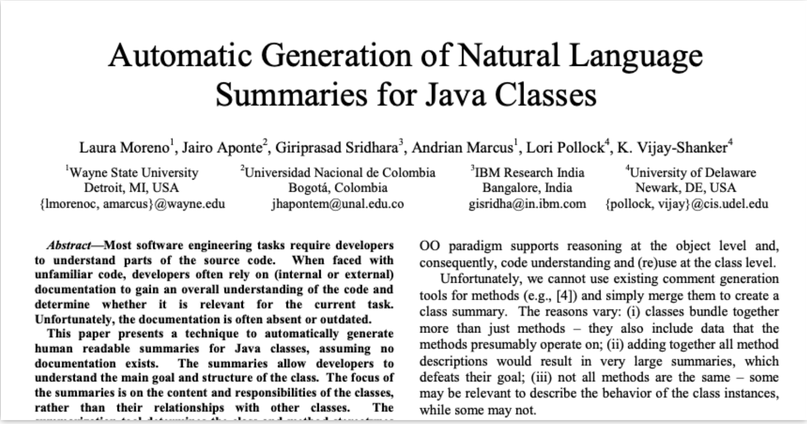
Empirical Software Engineering
https://doi.org/10.1007/s10664-021-09981-5

Check for updates

What do class comments tell us? An investigation of comment evolution and practices in Pharo Smalltalk

Pooja Rani[1] · Sebastiano Panichella[2] · Manuel Leuenberger[1] · Mohammad Ghafari[3] · Oscar Nierstrasz[1]

Previous works



Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno[1], Jairo Aponte[2], Giriprasad Sridhara[3], Andrian Marcus[1], Lori Pollock[4], K. Vijay-Shanker[4]

[1]Wayne State University Detroit, MI, USA {lmorenoc, amarcus}@wayne.edu
[2]Universidad Nacional de Colombia Bogotá, Colombia jhapontem@unal.edu.co
[3]IBM Research India Bangalore, India gisridha@in.ibm.com
[4]University of Delaware Newark, DE, USA {pollock, vijay}@cis.udel.edu

***Abstract*—Most software engineering tasks require developers to understand parts of the source code. When faced with unfamiliar code, developers often rely on (internal or external) documentation to gain an overall understanding of the code and determine whether it is relevant for the current task. Unfortunately, the documentation is often absent or outdated.**

This paper presents a technique to automatically generate human readable summaries for Java classes, assuming no documentation exists. The summaries allow developers to understand the main goal and structure of the class. The focus of the summaries is on the content and responsibilities of the classes, rather than their relationships with other classes. The

OO paradigm supports reasoning at the object level and, consequently, code understanding and (re)use at the class level.

Unfortunately, we cannot use existing comment generation tools for methods (e.g., [4]) and simply merge them to create a class summary. The reasons vary: (i) classes bundle together more than just methods – they also include data that the methods presumably operate on; (ii) adding together all method descriptions would result in very large summaries, which defeats their goal; (iii) not all methods are the same – some may be relevant to describe the behavior of the class instances, while some may not.

Empirical Software Engineering
https://doi.org/10.1007/s10664-021-09981-5

Check for updates

What do class comments tell us? An investigation of comment evolution and practices in Pharo Smalltalk

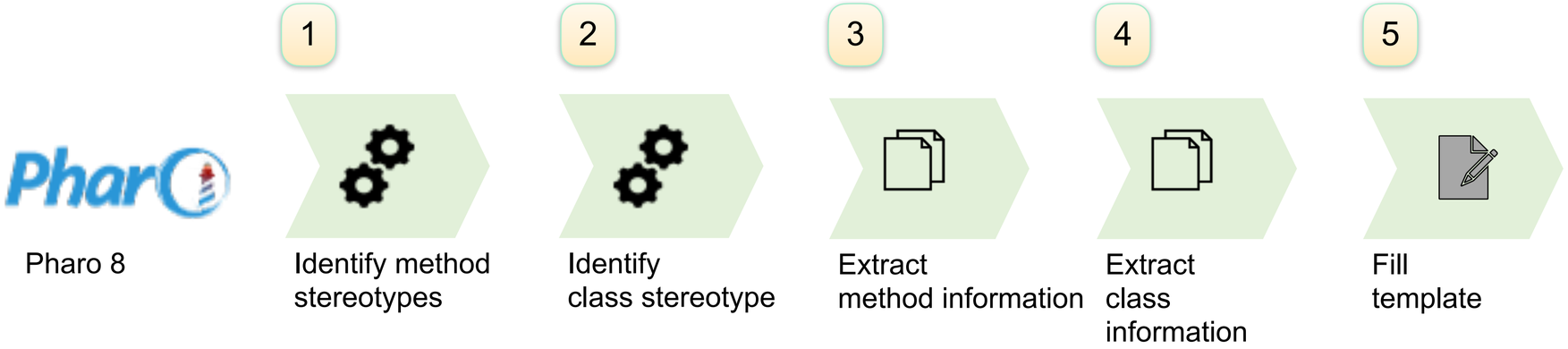
Pooja Rani[1] · Sebastiano Panichella[2] · Manuel Leuenberger[1] · Mohammad Ghafari[3] · Oscar Nierstrasz[1]

Methodology

Dataset

Can We Automatically Generate Class Comments in Pharo?

Methodology



1

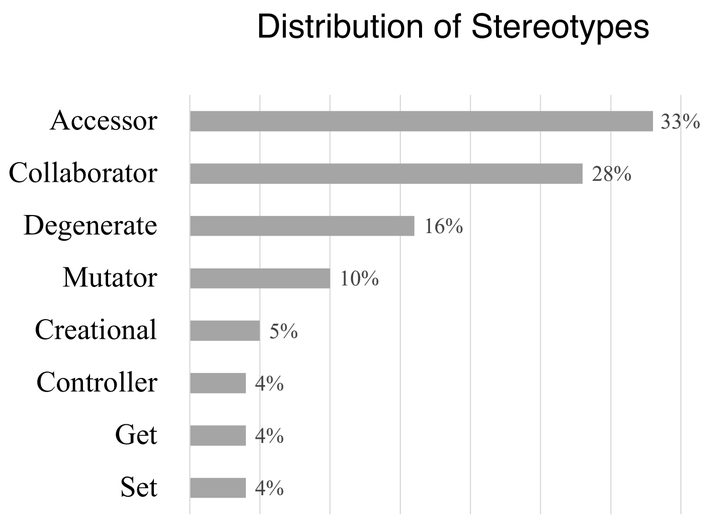
Method Stereotypes

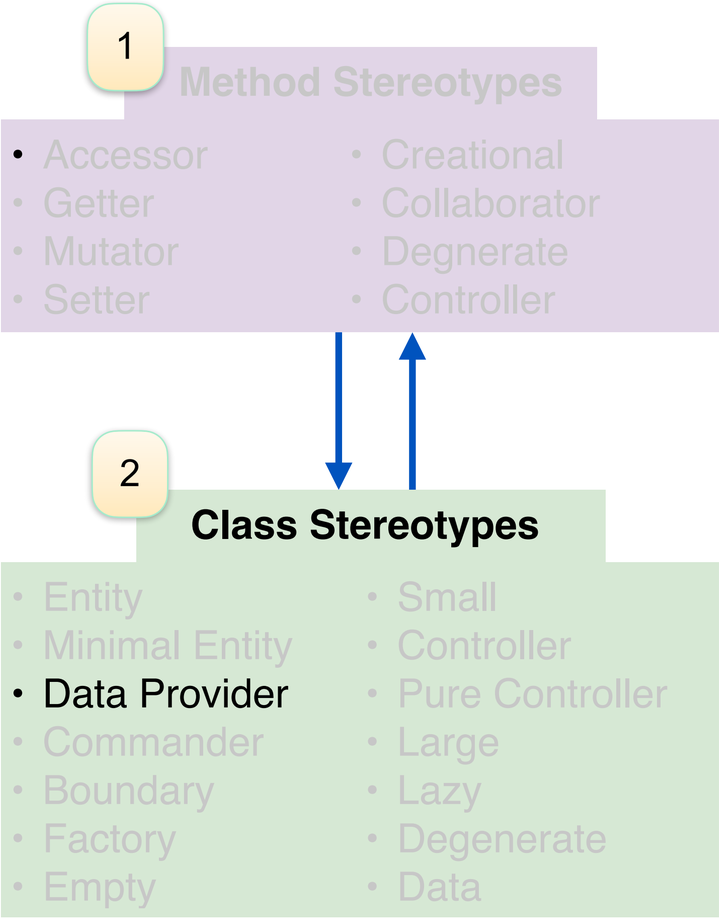
- Accessor
- Getter
- Mutator
- Setter
- Creational
- Collaborator
- Degnerate
- Controller

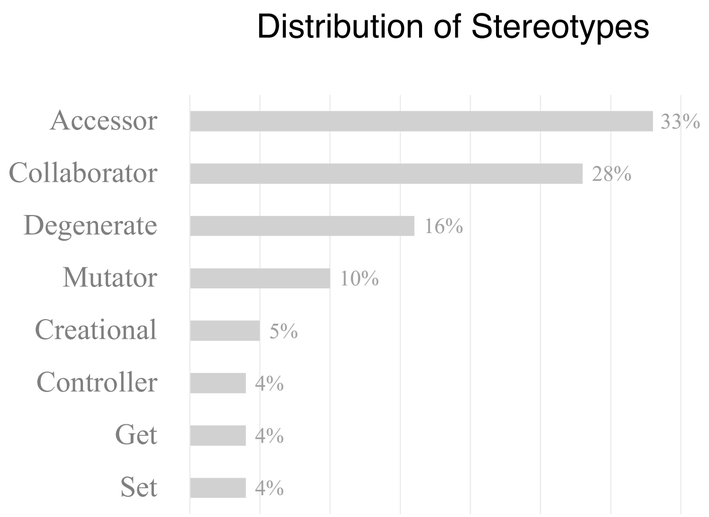
1

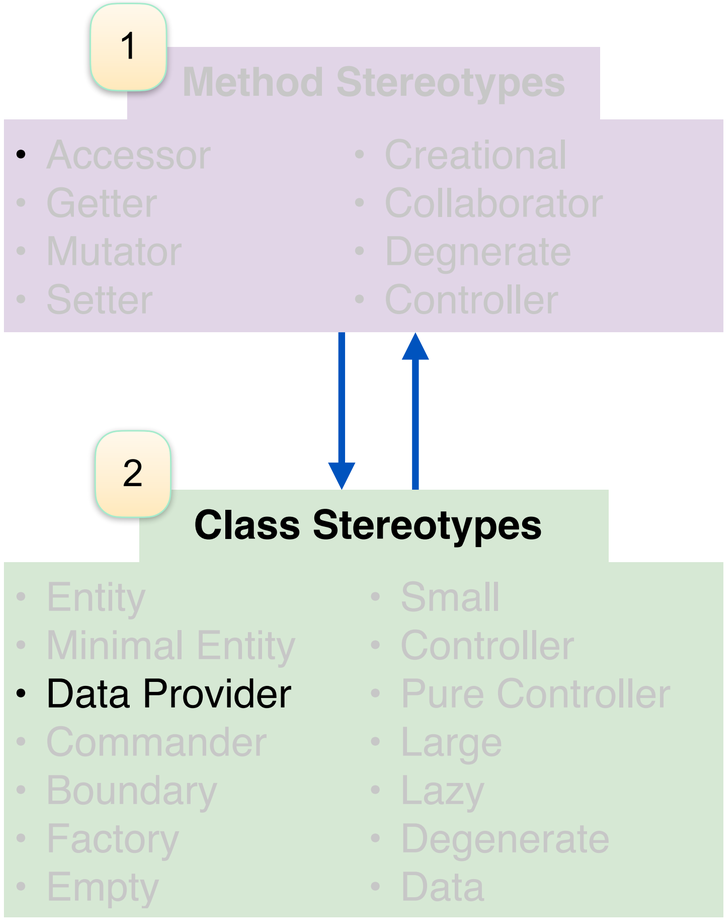
Method Stereotypes

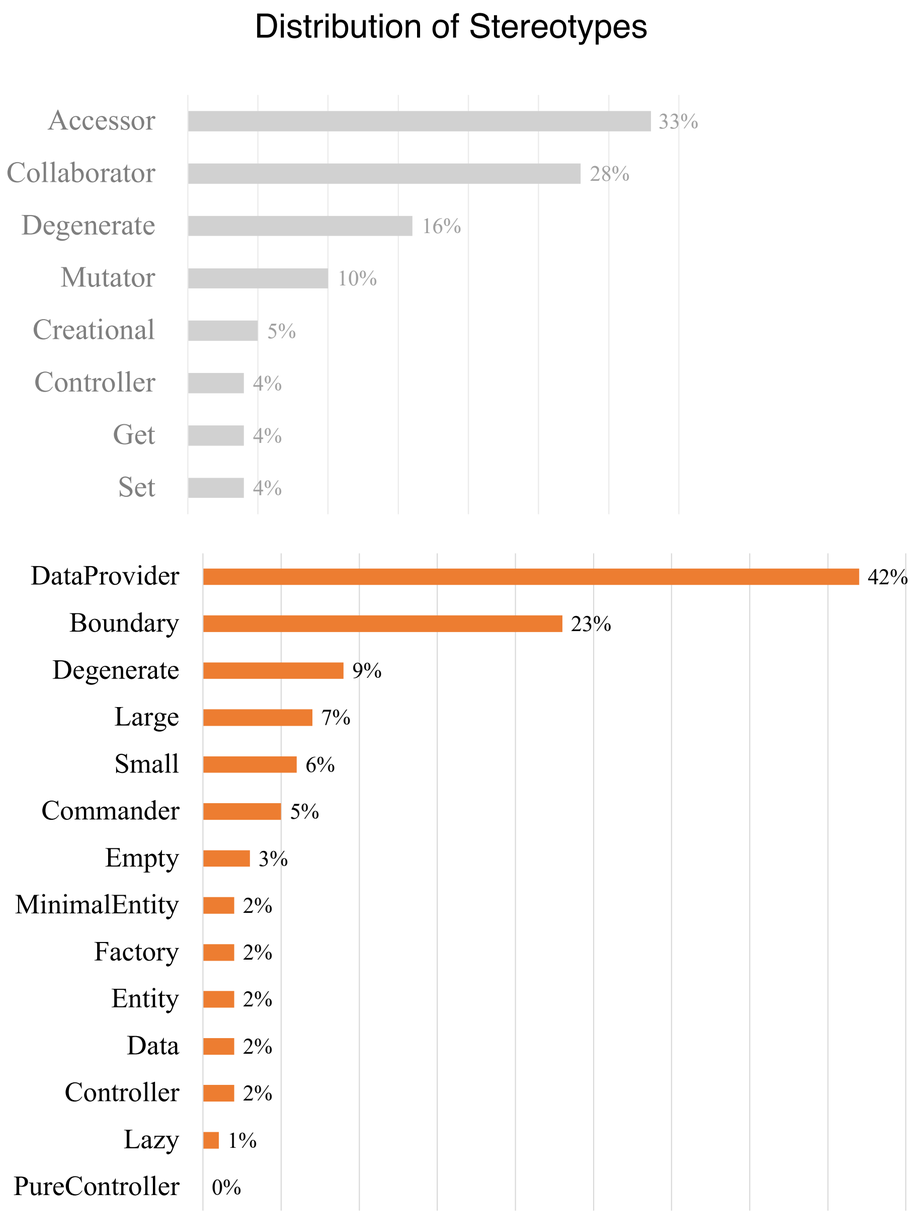
- Accessor
- Getter
- Mutator
- Setter
- Creational
- Collaborator
- Degnerate
- Controller

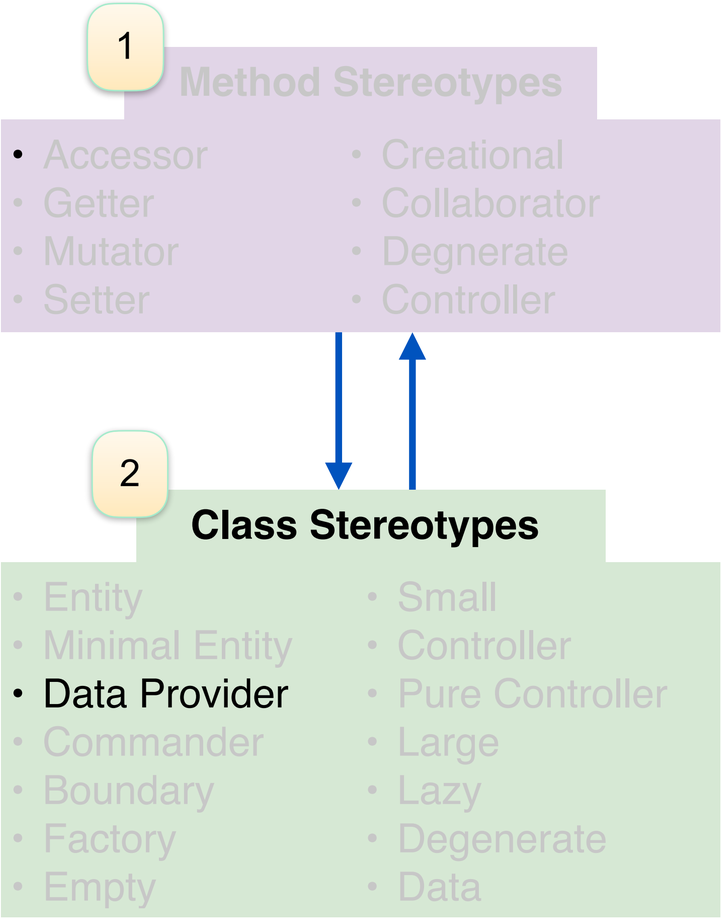



1
Method Stereotypes
• Accessor
• Getter
• Mutator
• Setter
• Creational
• Collaborator
• Degnerate
• Controller
2
Class Stereotypes
• Entity
• Minimal Entity
• Data Provider
• Commander
• Boundary
• Factory
• Empty
• Small
• Controller
• Pure Controller
• Large
• Lazy
• Degenerate
• Data


Distribution of Stereotypes
Accessor 33%
Collaborator 28%
Degenerate 16%
Mutator 10%
Creational 5%
Controller 4%
Get 4%
Set 4%


1
Method Stereotypes
• Accessor
• Getter
• Mutator
• Setter
• Creational
• Collaborator
• Degnerate
• Controller
2
Class Stereotypes
• Entity
• Minimal Entity
• Data Provider
• Commander
• Boundary
• Factory
• Empty
• Small
• Controller
• Pure Controller
• Large
• Lazy
• Degenerate
• Data


Distribution of Stereotypes
Accessor 33%
Collaborator 28%
Degenerate 16%
Mutator 10%
Creational 5%
Controller 4%
Get 4%
Set 4%
DataProvider 42%
Boundary 23%
Degenerate 9%
Large 7%
Small 6%
Commander 5%
Empty 3%
MinimalEntity 2%
Factory 2%
Entity 2%
Data 2%
Controller 2%
Lazy 1%
PureController 0%


1
Method Stereotypes
• Accessor
• Getter
• Mutator
• Setter
• Creational
• Collaborator
• Degnerate
• Controller
2
Class Stereotypes
• Entity
• Minimal Entity
• Data Provider
• Commander
• Boundary
• Factory
• Empty
• Small
• Controller
• Pure Controller
• Large
• Lazy
• Degenerate
• Data

eotypes

reational
ollaborator
egnerate
ontroller

otypes

Small
Controller
Pure Controller
Large
Lazy
Degenerate
Data

eotypes

reational
ollaborator
egnerate
ontroller

otypes

Small
Controller
Pure Controller
Large
Lazy
Degenerate
Data

3

Method Information

- External used methods
- Internally used methods

4

Class Information

- Class name
- Class stereotype description
- Classes used by the class
- Dependent classes
- Relevant keywords

Classname: RSShape

I have class stereotype:
  – DataProvider

I encapsulate data. I consist mostly of accessor methods.

I am using the classes:
– RSObjectWithProperty
– RSShapeAddedEvent
– Color

I am used by classes:
– RSCanvas
– RSComposite
– RSCustomCPController
– RSTContainer

I have relevant public methods which are ordered by their usage:
Externally:
– models
– width
– height
– extent
– parent

Internally:
– shape
– extent
– canvas
– model
– encompassingRectangle

My instance variables are:
– paint
– path
– border
– parent
– isFixed
– encompassingRectangle
– model

My defining keywords are:
border, is, shape, with, paint, color, parent, rectangle, encompassing, has

eotypes

reational
ollaborator
egnerate
ontroller

otypes

Small
Controller
Pure Controller
Large
Lazy
Degenerate
Data

3

Method Information

- External used methods
- Internally used methods

4

Class Information

- Class name
- Class stereotype description
- Classes used by the class
- Dependent classes
- Relevant keywords

I am using the classes:
– RSObjectWithProperty
– RSShapeAddedEvent
– Color

I am used by classes:
– RSCanvas
– RSComposite
– RSCustomCPController
– RSTContainer

I have relevant public methods which are ordered by their usage:
Externally:
– models
– width
– height
– extent
– parent

Internally:
– shape
– extent
– canvas
– model
– encompassingRectangle

My instance variables are:
– paint
– path
– border
– parent

eotypes

reational
ollaborator
egnerate
ontroller

3

Method Information

- External used methods
- Internally used methods

otypes

Small
Controller
Pure Controller
Large
Lazy
Degenerate
Data

4

Class Information

- Class name
- Class stereotype description
- Classes used by the class
- Dependent classes
- Relevant keywords

– models
– width
– height
– extent
– parent

Internally:
– shape
– extent
– canvas
– model
– encompassingRectangle

My instance variables are:
– paint
– path
– border
– parent
– isFixed
– encompassingRectangle
– model

My defining keywords are:
border, is, shape, with, paint, color, parent, rectangle, encompassing, has

Evaluation

24 class comments (auto generated)

3 developers: each comment evaluated by three

42 evaluations: 7 developers completed it

3 metrics: Adequacy, Conciseness, Comprehensibility

Evaluation

Adequacy: 44% participants said the comment is missing some important information

Evaluation

Adequacy: 44% participants said the comment is missing some important information

Conciseness- 54% participants said the comment contains a lot of unnecessary information

Evaluation

Adequacy: 44% participants said the comment is missing some important information

Conciseness- 54% participants said the comment contains a lot of unnecessary information

Comprehensibility- 14% participants said the comment are hard to read and understand

Future work

Improve heuristics to generate other types of information

Building ML approach for automatically inferring role-stereotype

Develop tools to validate comment quality

Can We Automatically Generate Class Comments in Pharo

Tool

https://github.com/PR-research-data-tools/Smalltalk-class-comment-generator

Paper

https://scg.unibe.ch/archive/papers/Rani22b.pdf

Replication Package on Zenodo

https://doi.org/10.5281/zenodo.6622011

Contact us

https://twitter.com/poojaruhal

https://seg.inf.unibe.ch/team/