Active Documentation: Helping Developers Follow Design Decisions

Sahar Mehrpour  George Mason University
Thomas D. LaToza  George Mason University
Rahul K. Kindi  Cornell University

Developer Experience Design Laboratory
Scenario: Using Documentation Today

- Alice is a developer in a Company.
- She is working to implement a small feature (a new Artifact B) in the codebase.
- Alice starts reading the documentation … But the documentation is too long.
- Alice reads one of the Design Decisions describing what alternative was chosen and why.
- Looking at the description of the design decision, she reads one of the Design Rules describing how to implement the design decision.
- She tries to connect the design rule to the code … But the documentation and the source code are large and hard to connect.
Scenario: Using Documentation Today

- After some time, Alice finds that she believes to be an **Example** illustrating how to implement an Artifact. Following this example, She tries to re-implement her new class.

- She writes some code and wants to know if it follows the design rules. But she is not sure that she is following the examples correctly, and that there aren’t other rules she missed.

- She looks at the **rule checkers** the company is using, but they only report defects about her use of Java and do not help with understanding these design decisions.

- Frustrated, she commits her code and waits for **code reviews** from other developers.
Active Documentation

Our solution: *active* documentation

- Design rules are translated into constraints and *actively checked* against code.

- Wherever a design rule applies to code, an *active link* between the documentation and code is generated.

- Developers can *actively update* the documentation.
Active Documentation

IntelliJ IDE plugin
Active Documentation System Architecture

- Independent from IDEs
- Two main components: IDE Connector and Main
- IDE Connector transfers data to/from the IDE
Active Documentation System Architecture

- IDE is responsible for reporting code change, active file in the editor, and updating the caret position

- Stored design rules (stored as .json) are accessible in the IDE
Active Documentation System Architecture

- IDE connector creates the AST of the source code.
- XML representation of the ASTs are easier to work with.
- We used srcML to create the AST. [Maletic et al. 2002]

IDE

Source code

Design Rules

Monitor code change

Monitor active file

Update caret position

IDE Connector

Run srcML

AST data

IDE Plugin

Main
The rule checker uses the design rules and the AST of code to extract snippets from code.

ACTIVEDOCUMENTATION is agnostic to the underlying rule checker.
Existing rule checkers *only* find violations of rules.

Developers need to search code to know *how* a rule is followed.

**ACTIVE DOCUMENTATION** shows snippets from code that **satisfy or violate** the rule.

<table>
<thead>
<tr>
<th>WHEN and HOW the rule should apply…</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quantifier</strong></td>
</tr>
<tr>
<td><strong>Constraint</strong></td>
</tr>
</tbody>
</table>

- Each Artifact must have a Command class.

**IF** a class is an artifact ___________________________ **Quantifier**
**THEN** it should have a Command class ___________________________ **Constraint**
In an IF/THEN structure of a rule:

IF part → Quantifier Query
THEN part → Constraint Query

Compare the results of queries → Satisfied and Violated Snippets
After generating code snippets, they are visualized in the user interface through different pages.

The User Interface sends and receives tasks to and from the IDE.
Rule Organization
Using Example Code Snippets
Instant Feedback

All Microtask commands must be handled by Command subclasses

If a method is a static method on Command THEN it should implement its behavior by constructing a new Command subclass instance. The Command class contains a number of static methods. Each method creates a specific type of Command by invoking the constructor of the corresponding subclass.

Examples: 5 out of 10; Violated: 1 out of 1

Commands must implement execute

IF a class is a subclass of Command THEN it must implement execute. Commands represent an action that will be taken on an Artifact. In order for this action to be invoked, each subclass of Command must implement an execute method. This method should not be directly invoked by clients, but should be used by the Command execution engine.

Examples: 5 out of 10; Violated: 1 out of 1

Artifacts should be marked as a data region with an @Entity annotation

IF an object is an artifact subclass THEN it needs to be an entity. To signal that instances of a class constitute a separate data region, the class should have the @Entity annotation. All Artifact subclasses should be marked as a data region.

Examples: 5 out of 10; Violated: 1 out of 1

Microtasks must have a reference to the Artifact that it belongs to

IF a class is a subclass of Microtask THEN it needs a field representing the reference to the associated entity. Each Microtask represents work to be done on an Artifact. As such, it needs to be connected back to its owning artifact through a reference to the Artifact. Without the reference, they need to have an ID of the artifact and for submitting they need to load the data beforehand.
Research Question

‣ Compared to traditional documentation, are developers able to use ACTIVEDOCUMENTATION to write code following design rules more quickly and successfully?

‣ In what ways does ACTIVEDOCUMENTATION support developers in writing code in an unfamiliar codebase?
Evaluation

21 Graduate students
Survey

Prior experience in Java
Programming experience 2-20 years (median 5)
Professional experience 0-15 years (median 2)

11 Control Participants
Warm-Up Task
Get Familiar with the IDE

10 Experimental Participants
Warm-Up Task
Get Familiar with the IDE and ACTIVE DOCUMENTATION

Task: Add a small feature to an existing code

- Existing code: web-based IDE, 9K LOC, 107 Java classes, abstraction based on artifacts (persisted in a persistence framework)
- Requested code: add a new artifact, add 20 lines of code, edit 2 lines of code
Result - Quantitative

<table>
<thead>
<tr>
<th>Control Group</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted Lines of Code</td>
<td>45.00</td>
<td>3.56</td>
<td>20.33</td>
<td>68.33</td>
<td>7.44</td>
<td>1.78</td>
<td>20.67</td>
<td>29.67</td>
<td>4.44</td>
<td>6.33</td>
<td>48.89</td>
<td>1.89</td>
</tr>
<tr>
<td>Added Lines of Code</td>
<td>36.00</td>
<td>3.00</td>
<td>12.00</td>
<td>70.00</td>
<td>1.00</td>
<td>1.00</td>
<td>8.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Removed Lines of Code</td>
<td>39.64</td>
<td>3.40</td>
<td>19.82</td>
<td>3.39</td>
<td>8.37</td>
<td>2.54</td>
<td>27.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Duration (Minutes)</td>
<td>20.67</td>
<td>8.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p value</td>
<td>0.142</td>
<td>0.015</td>
<td>0.038</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Irrlrvnt</td>
<td>0.056</td>
<td>0.043</td>
<td>0.082</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experimental Group</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted Lines of Code</td>
<td>29.00</td>
<td>3.00</td>
<td>6.00</td>
<td>47.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added Lines of Code</td>
<td>29.00</td>
<td>3.00</td>
<td>3.71</td>
<td>17.44</td>
<td>5.30</td>
<td>0.33</td>
<td>9.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Removed Lines of Code</td>
<td>6.36</td>
<td>5.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Duration (Minutes)</td>
<td>6.33</td>
<td>4.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p value</td>
<td>0.142</td>
<td>0.015</td>
<td>0.038</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Irrlrvnt</td>
<td>0.056</td>
<td>0.043</td>
<td>0.082</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All Participants</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted Lines of Code</td>
<td>37.33</td>
<td>4.00</td>
<td>13.33</td>
<td>58.61</td>
<td>4.67</td>
<td>0.94</td>
</tr>
<tr>
<td>Added Lines of Code</td>
<td>29.00</td>
<td>3.00</td>
<td>8.50</td>
<td>70.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Removed Lines of Code</td>
<td>28.65</td>
<td>4.46</td>
<td>15.59</td>
<td>15.77</td>
<td>7.37</td>
<td>1.95</td>
</tr>
<tr>
<td>Task Duration (Minutes)</td>
<td>13.33</td>
<td>15.77</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p value</td>
<td>0.142</td>
<td>0.015</td>
<td>0.038</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Irrlrvnt</td>
<td>0.056</td>
<td>0.043</td>
<td>0.082</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Experimental participants were **3 times** faster in starting editing the code and **28%** faster in finishing the task.
- Experimental participants added few lines of code and removed more lines of code.
- Experimental participants submitted **98%** fewer incorrect LOC.
# Result - Qualitative

<table>
<thead>
<tr>
<th>Control Group</th>
<th>Experimental Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>× Challenges in finding relevant design decisions within the design documentation</td>
<td>✓ Used Violated Rules page to find relevant design decisions.</td>
</tr>
<tr>
<td>× Challenges in connecting code with design decisions</td>
<td>✓ Used the violated snippets to identify relevant places to make changes.</td>
</tr>
<tr>
<td>× Challenges in finding relevant pieces of code, scattered in different classes</td>
<td>✓ Used example snippets listed to compare examples of the rule and the faulty lines of code.</td>
</tr>
<tr>
<td></td>
<td>✓ Used real-time feedback to detect errors and violations early, immediately after changing the code without running the application.</td>
</tr>
</tbody>
</table>
Active Documentation: Helping Developers Follow Design Decisions
Sahar Mehrpour, Thomas D. LaToza, Rahul K. Kindi

Scenario

- After some time, Alice finds a class assuming being an Artifact and considers it as an Example. She tries to re-implement them in the new class.
- She writes some code and wants to know if it conforms with the design rules … But she is not able to verify it herself.
- She looks at the rule checkers the company is using, but they are only reporting universal defects and not helpful.
- Frustrated, she commits her code and waits for code reviews from other developers.

Active Documentation

Our solution: active documentation

- Design rules are translated into constraints and actively checked against code.
- Wherever a design rule applies to code, an active link between the documentation and code is generated.
- Developers can actively update the documentation.

Instant Feedback

- Experimental participants were 3 times faster in starting editing the code and 28% faster in finishing the task.
- Experimental participants added few lines of code and removed more lines of code.
- Experimental participants submitted 98% fewer incorrect LOC.

Result - Quantitative

<table>
<thead>
<tr>
<th>Control Group</th>
<th>Experimental Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>45.00</td>
</tr>
<tr>
<td>Median</td>
<td>36.00</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>39.64</td>
</tr>
<tr>
<td>Diff Time (Minutes)</td>
<td>3.56</td>
</tr>
<tr>
<td>Submitted Lines of Code</td>
<td>20.33</td>
</tr>
<tr>
<td>Added</td>
<td>68.33</td>
</tr>
<tr>
<td>Removed</td>
<td>7.44</td>
</tr>
<tr>
<td>First Edit</td>
<td>1.78</td>
</tr>
<tr>
<td>Task Duration</td>
<td>20.67</td>
</tr>
<tr>
<td>Missing</td>
<td>1.78</td>
</tr>
<tr>
<td>Incorrect</td>
<td>20.67</td>
</tr>
<tr>
<td>Task Irrelevant</td>
<td>8.00</td>
</tr>
</tbody>
</table>

Result - Qualitative

- Experimental participants used violated rules page to find relevant design decisions.
- Experimental participants used example snippets listed to compare examples of the rule and the faulty lines of code.
- Experimental participants used real-time feedback to detect errors and violations early, immediately after changing the code without running the application.

Thank You!