Computer Languages, Systems & Structures 46 (2016) 44–65

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Recording concerns in source code using annotations

Matúš Sulír*, Milan Nosáľ, Jaroslav Porubän

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia

ARTICLE INFO

Article history:
Received 25 January 2016
Received in revised form
12 July 2016
Accepted 12 July 2016
Available online 18 July 2016

Keywords:
Program comprehension
Concerns
Source code annotations
Empirical studies

ABSTRACT

A concern can be characterized as a developer's intent behind a piece of code, often not explicitly captured in it. We discuss a technique of recording concerns using source code annotations (concern annotations). Using two studies and two controlled experiments, we seek to answer the following 3 research questions: (1) Do programmers' mental models overlap? (2) How do developers use shared concern annotations when they are available? (3) Does using annotations created by others improve program comprehension and maintenance correctness, time and confidence? The first study shows that developers' mental models, recorded using concern annotations, overlap and thus can be shared. The second study shows that shared concern annotations can be used during program comprehension for the following purposes: hypotheses confirmation, feature location, obtaining new knowledge, finding relationships and maintenance notes. The first controlled experiment with students showed that the presence of annotations significantly reduced program comprehension and maintenance time by 34%. The second controlled experiment was a differentiated replication of the first one, focused on industrial developers. It showed a 33% significant improvement in correctness. We conclude that concern annotations are a viable way to share developers' thoughts.

1. Introduction

Programmers developing a program continuously create a mental model, which is a representation of the program in their mind. They try to express the mental model in the programming language constructions. However, some parts of the mental model are not explicitly expressed in the source code. They are either implicitly indicated in complicated implementation details or lost.

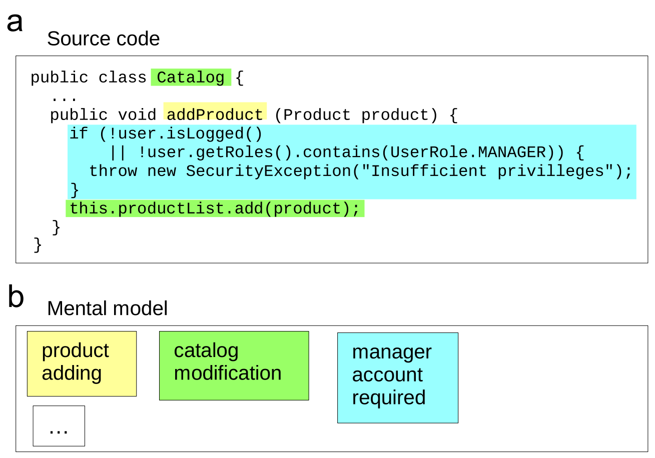
1.1. Motivation

Suppose a developer encounters the Java code excerpt shown in Fig. 1(a). While reading it, the following ideas may gradually appear in his or her mind:

- The method `addProduct` adds some product somewhere.
- As it is in the `Catalog` class, it adds the product to the catalog.

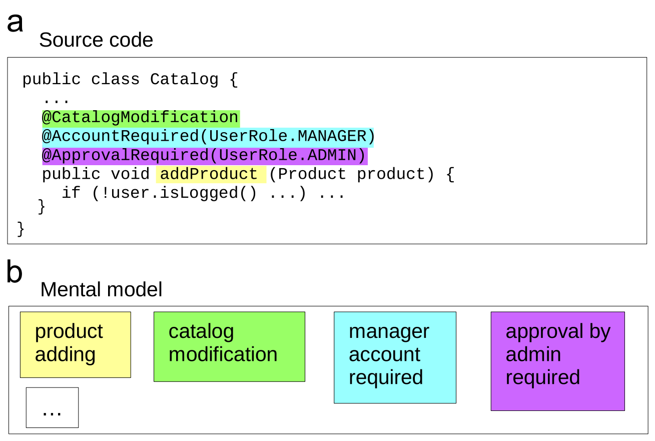
* Corresponding author.
E-mail addresses: matus.sulir@tuke.sk (M. Sulír), milan.nosal@gmail.com (M. Nosáľ), jaroslav.poruban@tuke.sk (J. Porubän).

http://dx.doi.org/10.1016/j.cl.2016.07.003
1477-8424/© 2016 Elsevier Ltd. All rights reserved.



```
public class Catalog {
  ...
  public void addProduct (Product product) {
    if (!user.isLogged()
        || !user.getRoles().contains(UserRole.MANAGER)) {
      throw new SecurityException("Insufficient privilleges");
    }
    this.productList.add(product);
  }
}
```

Fig. 1. An example of unannotated source code and the corresponding mental model.



```
public class Catalog {
  ...
  @CatalogModification
  @AccountRequired(UserRole.MANAGER)
  @ApprovalRequired(UserRole.ADMIN)
  public void addProduct (Product product) {
    if (!user.isLogged() ...) ...
  }
}
```

Fig. 2. The annotated source code and the corresponding mental model.

- An addition to a catalog changes its state – the catalog is modified.
- There is a condition checking whether a user is logged.
- To successfully perform this operation, a logged user must be a manager; otherwise an exception is thrown.

While this process may seem natural, it has the following shortcomings:

- The fact that the code modifies a catalog is formed only after reading two parts of the code far apart.
- The information that a "manager account" is necessary is formed after reading a rather lengthy piece of code.
- There is one important piece of information not identified at all: the added product actually appears in the catalog only after the administrator approves it. This information is not apparent from this excerpt, and may be the result of cooperation of many remotely related classes.

Now suppose the developer originally writing the code, or any other programmer maintaining it, annotated the source code with concern annotations (source code annotations are declarative marks used to decorate source code with metadata [1]), as displayed in Fig. 2(a). The developer later reading it would rapidly obtain a mental model containing useful information about the code, without even looking at the method implementation, scrolling through the code to stitch the pieces of information, or even debug it to explain unexpected behavior. Furthermore, it would be possible to find all methods requiring approval in case of refactoring or business policy changes. As annotations are a part of the Java language, no additional tools are required: standard IDE (integrated development environment) features like Find Usages could be used.