LivCos

Code Path Doc

Table of Contents

1 Introduction

A code path documentation describes the path of one thread through the program code.

See also the corresponding brainstorming Web Map.

2 Example

--content--

3 Rules

To keep the documentation simple and easy to read, the author needs to follow some rules:

- Focus on one path only!

  While walking through your code, you will be offered many junctions, leading to alternative paths. You need to decide on one direction.

  Think of it as the travel diary from the last trip you have taken. There might have been some places you have missed, since you have taken only one concrete tour. Visit again next year and take another route to cover all your interest.

- Try to make a point!

  There can be thousands of possible paths through your code. Pick the ones, that are really meaningful. They could provide a better understanding of the system or clear a critical situation.

- Choose relevant state values!

  Only certain object states, parameter-, global configuration- or system-values have an influence on the route of the path. Some others can be descriptive, but all the rest is only confusing the reader.

  You can also choose value ranges, if, and only if, all these values lead the path in the chosen direction.

4 Features

Some benefits can ease the communication of your program code noticeably.

- Reduced to the max! Presents only the calls, relevant to your topic, in one view.
- Includes calls across different services, patterns and/or components.
- Includes value states from the local, object, global or system context.
- Includes external state-change event notifications.

5 Why another documentation?

Why do we need yet another documentation? All the information about the code is available in the source code and containing comments and JavaDoc.

5.1 Source Code

Don't I find this information in the source code directly (with containing comments and JavaDoc)?

Yes, most of the information about a code path is of course available in the source code. With a proper IDE to comfortably navigate the code (go to declaration,...) and extensive comments about pre- and post-conditions, cross-cutting aspects and concurrent data sharing, you'll probably find yourself all the information needed.

The information in the source code can become so extensive, it's hard to focus on a particular topic. When we want to explain a certain limit-behavior or to focus on a specific flow of events, maybe even concurrent ones, the relevant lines of code and comment can become hard to grasp.

5.2 Debugger

With a debugger I can walk the code path myself. No need for a documentation.

Well, most situations, explained with a code path documentation, require certain specific value states to start with. Some even external changes along the way. It could need some work to provide these values to the debugger context.

As with browsing the source code directly, it can become difficult to focus on the relevant calls only.

Not every reader has a debugger right at hand.

5.3 Code Path Doc

5.3.1 Unit Tests

5.3.2 Javadoc Tool

Javadoc is a tool for generating API documentation in HTML format from doc comments in source code.

5.3.3 diagrams

5.3.4 Use Cases