International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

Int. J. Sci. Res. Arch.
IJSRA
International Journal of Science and Research Archive
Research Journal Archive, INDIA

(Review Article)

Check for updates

Writing effective use cases for Application Programming Interfaces (APIs)

Nataliia Marko *

Department of Business Analysis, International Institute of Business Analysis.

International Journal of Science and Research Archive, 2025, 15(02), 1712–1714

Publication history: Received on 19 April 2025; revised on 27 May 2025; accepted on 30 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1635

Abstract

The article presents the review of use cases as one of the effective means of requirements documentation for technology project, in particularly for those that involves data exchange via Application Programming Interfaces (APIs). It includes review of certain pitfalls that may occur and steps how to avoid them. The article also offers several best practices and a template, based on the real-life implementation of change requests.

Keywords: Use Case; Implementation; Stakeholders; Complicated Solution; Product Ownership; Business Analysis; Application Programming Interface (APIs).

1. Introduction

Implementation of the API is not an easy process, as it requires good understanding of the user journey and thus a data flow that usually are documented in various forms of requirements. Writing use cases is one way of documenting requirements. They serve a good guidance for implementation of functionalities, can be a used in discussion, clarification of the flow with requestors, or clarification of system behavior.

2. Why should we incorporate use case?

Drafting use cases for a technology project is crucial not only for documenting specification of change request implementation, but also for code validation. ‘There are still no trusted guides about how to write (or review) use cases, even though it is now nearly a decade since use cases have become the “norm” for writing functional requirements for object-oriented software systems, and are gaining acceptance for embedded software and business process reengineering work [1, 2].’ This highlights that use cases are used to present what functionality technical team should implement, required features and functions of the system.

When examining functionalities of Application Programming Interfaces (APIs), it’s essential to consider that they act as connectors for interaction between applications, defining how services communicate within an ecosystem. The core concept of an API’s flow is to have an application that triggers the API, which should be capable of receiving the call and responding with the relevant information. It is important to know that APIs ‘specify accepted requests and their formats, enabling seamless data and functionality exchange between software entities [3].’ With this basic understanding of API flow, we can begin drafting a use case, although it’s insufficient on its own.

There are numerous helpful guides available on how to start writing use cases. ‘Work top-down: Scope, Actors, Goals, Main story, Alternative conditions, Alternative paths. Work middle-out: Start at user goals, go up to strategic level, then down to subfunctions. Get clear about the scope of the system you are describing [1, 12].’ Taking into consideration highlighted break down, it is important to understand the scope of the requested change in order to focus on solution.

* Corresponding author: Nataliia Marko

That will help in ‘description of the possible sequences of interactions between the system under discussion and its external actors, related to a particular goal [1, 15].’ This part will help to highlight who the actor or the user is, how this actor is going to interact with the system and what the system should respond with. These steps help to determine main skeleton of the use case. Such use case should be filled in with much more detailed descriptions based on clarifications with stakeholders.

3. Example of Use Case for Validation of API functionality

It is vital to understand from the very beginning of use cases documentation that ‘the use case is associated with the goal of one particular actor, who is called primary actor for that use case [1, 16].’ Based on the actor or a user business analyst or product analyst should determine all the ways how this actor can use particular system to get specific goal and document that. When business analyst deals with description of the use cases for API’s functionality, use cases present the value user achieves while dealing with the particular system. ‘A use case describes the possible outcomes of the attempt to accomplish a particular goal that the solution will support. It details different paths that can be followed by defining primary and alternative flows. The primary or basic flow represents the most direct way to accomplish the goal of the use case. Special circumstances and exceptions that result in a failure to complete the goal of the use case are documented in alternative or exception flows. Use cases are written from the point of view of an actor and avoid describing the internal workings of the solution [2, 356].’ This statement underlines that use case should be mainly focused on the requirement when there is interaction between the actor and the system. However, when project deals with APIs, it is required to mention that API is responsible for data transfer between the systems, which are triggered.

Based on practice, documentation for functionality of the APIs should contain description of persona, condition, expected output, logic or high-level logic, required header request parameters, optional request parameters, required response parameters, optional response parameters, response code, and message. Specification of this parameters may be helpful for understanding what exactly is required to trigger the system, or API. Such description should focus on user journey, persona, and data flow.

Table 1 Example of Use Case for Validation of API functionality

Who?	**What?**
Persona	Specifies who the actor interacting with the system is
Condition	Specifies special condition that should be satisfied
Expected output	Specifies what the expected outcome from interacting with the system should be
High-level logic	Specifies if any specific logic should be taken into account when the actor is interacting with the system
Required header request parameters	Specifies required parameters to invoke API
Optional response parameters	Specifies optional parameters with which API may be invoked
Response code	Specifies API responds with error or with response parameters
Message text	Specifies possible message text related to alert that may occur during interaction with the system

This Table “Example of Use Case for Validation of API functionality” may serve as a general template for use case, though it should be expanded when applying for the real-life technology project. It describes persona or actor that will interact with the system, what actions are required to trigger the system, as without required parameters the system will is not supposed to be triggered. It also specifies the expected result that the actor may expect to get back from the system. Specification of use case in the form of grid is easy to explain, clear, and easy to update if required. Thus, documentation of use cases in this form is preferred by tech and business analysts.

While writing use cases it is recommended to avoid the following pitfalls that occur quite often and may lead business or product analyst not into the right direction. ‘The system boundary is undefined or inconsistent [4,13].’ That is why it is mandatory to define system boundary before writing the use case. ‘There are too many use cases’ or ‘Use Case specification is too long’ or ‘Use Case is never finished [4,13].’ That is why it is better to focus on one piece of it. In case huge functionality has to be described, then it is better to break it into smaller parts. ‘Use cases have been successfully used to document the behavior of software systems, often just before a new round of maintenance work is done [1, 16].' Therefore, use cases help clearly specify functionality within the defined boundaries of scope, describe design, testing outcome, maintenance, present requirements.

The use case outlines the potential results of trying to achieve a specific objective that the solution will facilitate. Special situations, exceptions, encoded logic should be taken into account. Use cases are crafted from the actor's perspective and refrain from detailing the solution's internal mechanisms.

4. Benefits of use cases in testing

Applying use cases for verification of the implemented functionalities is one the most beneficial reason why business analyst or product analysts should have them. Use cases help to highlight what exactly is to being tested, what the expected results should be for happy path and for unhappy path. While writing use cases, it is necessary to understand that documentation should describe the behavior of the system, or what the system does, in case of functional requirement, and with non-functional requirements it should describe how the system does that. 'What all these uses of the use case idea have in common is that they show how the system responds to the outside world, the responsibilities and behavior of the system, without revealing how the internal parts are constructed [1, 16].' Therefore, use cases help in identifying whether the system works expected to specific scenario.

Use cases detail the interactions among the primary actor, the solution, and any secondary actors necessary to achieve the primary actor's goal. Generally, use cases are started by the primary actor, but in some methods, they may also be initiated by another system, an external event, or a timer.

5. Conclusion

Documentation in the form of use case is beneficial for successful implementation and validation of the coded functionality. It allows to present complex solution to tech team, describe system functions, and how the actor interacts with this system. Use cases help to test happy and unhappy paths scenarios, validate if the expected results are received as an outcome. Use cases serve as important way of maintaining documentation for change requests and clarification with stakeholders.

References

[1] Alistair Cockburn. Writing Effective Use Cases. Humans and Technology.in preparation for Addison-Wesley Longman. 2000.

[2] BABOK. A Guide to The Business Analysis Body of Knowledge. IIBA. International Institute of Business Analysis, Toronto, Ontario, Canada. Version 3; 2015.

[3] Eric Pulsifer, what is an API (Application Programming Interface)? Available from: https://konghq.com/blog/learning-center/what-is-api

[4] S.Lilly. Use Case Pitfalls: Top 10 problems from Real Project Using Use Cases. Procs of Tools USA'99, IEEE. 1999 (p13)