



# **Treating Use Case Abuse: An Experience-Based Guide**

**By Katherine Marshak**

**IconATG**  
**[www.iconatg.com](http://www.iconatg.com)**  
**866-785-4266**

## Abstract

*Use cases are a popular way to define functional requirements because well-written use cases are easy to read, useful to a wide array of team members, and explain the requirements from an actor's perspective, giving them context. However, many project teams improperly apply the use case approach. This white paper explains how you can diagnose and treat this use case abuse, helping your project to recover from or prevent these common problems. This white paper is written for people actively applying the use case approach. It does not explain fundamental use case concepts.*

*It is important to note that the IconProcess is use case driven, which makes preventing "use case abuse" an important area of concern for those applying the IconProcess. The IconProcess is IconATG's best practices for Web development. Information is available at [www.iconprocess.com](http://www.iconprocess.com).*

Abstract.....	2
What Is Use Case Abuse?.....	4
Abuse 1: User-Perspective-Phobia .....	4
Symptoms.....	4
Diagnosis.....	4
Treatment and Prevention .....	4
Abuse 2: Schizo-Workflow-Phrenia.....	5
Symptoms.....	5
Diagnosis.....	5
Treatment and Prevention .....	5
Abuse 3: Skopos Imbalance .....	6
Symptoms.....	6
Diagnosis.....	6
Treatment and Prevention .....	6
Abuse 4: Compulsive User Interface Disorder .....	7
Symptoms.....	7
Diagnosis.....	7
Treatment and Prevention .....	7
Abuse 5: Compulsive Architectural Disorder.....	8
Symptoms.....	8
Diagnosis.....	8
Treatment and Prevention .....	8
Conclusions.....	9
For More Information.....	9

## What Is Use Case Abuse?

Use cases are a popular way to define functional requirements because well-written use cases are easy to read, useful to a wide array of team members, and explain the observable requirements from an actor's perspective, providing context for the requirements. Use cases are part of the Unified Modeling Language (UML). Well-intentioned teams often adopt the use case approach but then unknowingly misuse their use case model and specifications. Each form of use case abuse – the accidental or purposeful misuse of use cases – causes harm to the project. IconATG's consultants help project teams apply use cases to hundreds of projects in many industries. This gives us a valuable perspective on common problems, as well as recommendations for correction and prevention. This treatment and prevention guide is based on that experience.

## Abuse 1: User-Perspective-Phobia

### Symptoms

- » You fear meeting with real users.
- » You fear demonstrating the application for real users.
- » You have never actually met a real user of the application you're building.
- » Users are not included when identifying or reviewing requirements.
- » Discussions about features and requirements often include phrases like these:
  - "I guess users would do that."
  - "The developers say this way is cooler, so that's what we'll do."
  - "We'll teach the users how to do that."
  - "We already know what the users want. We don't need to talk with them."
  - "Let's ask Carla. She had that job 5 years ago."
  - "Herb from marketing is assigned to represent the users."

### Diagnosis

You have User-Perspective-Phobia. You have not analyzed the target users sufficiently to understand them and to define and design an application to meet their needs.

The symptoms above indicate a lack of knowledge about the roles that users play while interacting with the application. Actors are roles that users, systems, and devices play while interacting with the application under development. Many people developing use case models spend little time analyzing actors because the actors are outside of the application boundary. However, the actors representing human users and the users themselves need and deserve more research and analysis because they are the primary reason for developing software.

### Treatment and Prevention

In addition to identifying roles people play in relation to the application being built, performing user research gives your team a much deeper understanding of user needs.

One<sup>1</sup> very effective technique for analyzing users is developing personas to supplement actor definitions. A persona<sup>2</sup> is a description of a fictional, archetypal user, which is based on research of real users. Research focuses on demographics (e.g., age, geography, education, etc.) and social history, culture, goals, and motivations. The user research becomes much more powerful when synthesized into a few key personas. Personas are used to guide and evaluate business strategy, software requirements, and user experience decisions. Develop personas early in the project (e.g., during Inception). Soon they become concrete, life-like characters your team uses as objective guides in project decisions. If you're not sure if feature A or B is really needed, just check with the personas. Personas are not wishy-washy in their needs. Personas represent the needs of real users in ways that even real users cannot. Personas do not bend to comply with what is easy for the development team to produce. They are not easily persuaded that their expectations are unreasonable. They help the team understand and strive to satisfy real users' needs.

## Abuse 2: Schizo-Workflow-Phrenia

### Symptoms

- » Your use case model has a complex set of relationships between use cases.
- » The use case relationships need to be updated when the user interface navigation changes.
- » The use case model does not become stable as the project progresses.

### Diagnosis

You have Schizo-Workflow-Phrenia. You are trying to make your use case model fulfill multiple purposes, specifically to document business process or workflow decisions and user interface flow, as well as to define the application's scope.

Most project teams do not develop business process models but later find they need to explain the preferred way to concurrently perform a business process and use a new application. These teams often try to force the use case model to satisfy this need. They create poorly scoped use cases to represent individual screens or web pages and then they misuse the UML «include» and «extend» relationships to visually model user interface flow. A variation on this is to apply the use case notation to document the functional decomposition of a business process instead of properly defining use cases. In both of these situations, you are misusing the use case model.

### Treatment and Prevention

1. Determine which purposes you are trying to make your use case model fulfill.
2. Remember that the use case model's primary purpose is to communicate the scope and intended usage of the application's functional requirements.

<sup>1</sup> For information on other techniques such as industry analysis, contextual inquiry, and usability benchmarks, refer to the IconProcess at [www.iconprocess.com](http://www.iconprocess.com).

<sup>2</sup> Alan Cooper (Cooper Interaction Design) developed the use of personas. Cooper's book *The Inmates are Running the Asylum* is an excellent resource.

3. Select an appropriate approach to fulfill the extraneous purposes you were forcing the use case model to fulfill. In both cases noted above, documenting user interface and business process flow, UML activity diagrams are better suited to the purpose. Non-UML flow charting models could also be used if your project is not using UML or if the business team or others are already using another flow charting notation. Developing a business use case model may also be useful.
4. Remove the extraneous relationships from the model. If your project has this illness, you probably also suffer from Skopos Imbalance.

## Abuse 3: Skopos<sup>3</sup> Imbalance

### Symptoms

- » You have hundreds of very small use cases.
- » Use cases represent system functions.
- » You have a few very large use cases.
- » Subject matter experts, stakeholders, and others cannot understand the use case model because it is so complex or so abstract.
- » Your team would rather use a functional decomposition approach.
- » Your team does not realize that there **is** a difference between use cases and functional decomposition.

### Diagnosis

You have Skopos Imbalance. Skopos (meaning “scope”) Imbalance occurs when the purposes of the use cases are imbalanced or improperly defined, resulting in some use cases that are very short or very long.

### Treatment and Prevention

The first step in treating this disease is to re-evaluate each use case by asking whether or not the use case provides an observable result of business value to the actors associated with the use case. The emphasis is on two points: “observable” and “business value.” The second step is to refactor the use case model and the associated use case specifications to reflect the improved organization of the use case model.

Each actor is interacting with the application for some reason; the actor is trying to accomplish a goal. Those goals are as diverse as finding a specific piece of information, controlling a diagnostic medical device, or buying a book. The goals should guide the formation of the use cases. Use cases focus on **an actor’s usage** of the system. Use cases like “Save Data” or “Validate PIN” are too small because they do not allow an actor to achieve an observable result of **business value**. When you believe a use case may be too small to yield observable business value, ask a series of “why” questions to uncover the underlying need. For example:

Q: “This use case is named Save Work. Why would someone want to save their work? What are they really trying to do when they want to save their work?”

---

<sup>3</sup> Greek for “Scope”

A: “The customer service reps save their work when they are handling a new complaint. They like to save their work often because the current system is unreliable and they don’t want to lose their work. They also update other complaints with the resolution or if the same customer calls back with more information about the initial complaint.”

Q: “Oh, so users are really trying to maintain complaints. We should replace these five use case with a single use case called Maintain Complaint...”

This technique can also be used to prevent poorly scoped and imbalanced use cases.

## Abuse 4: Compulsive User Interface Disorder

### Symptoms

- » Your use cases are never finished.
- » Your user interface designers want sign-off authority on the use cases.
- » You update use case specifications when the user interface design changes.
- » Your use case specifications reference user interface design decisions such as:
  - page or screen names,
  - pick lists or other widgets,
  - buttons, or
  - the order of specific fields.

### Diagnosis

You have Compulsive User Interface Disorder. You include user interface information in your use cases.

This form of use case abuse leads to use cases that are never finished or signed-off. User interface and user experience<sup>4</sup> decisions change frequently during a project. User interface design is a solution to the needs defined by the requirements. By tightly coupling the requirements to the user interface design, you either delay completing the requirements until the solution is complete, which causes confusion for much of the development and test teams, or you unnecessarily limit your ability to improve the user interface design.

### Treatment and Prevention

Having ever changing or non-signed-off use cases can be prevented by keeping the user interface information out of the use cases. Use artifacts such as site maps, wireframes, page designs, screen flows, or prototypes to capture user interface design decisions. Separating requirements and user interface decisions increases your ability to clearly define requirements **and** retain the flexibility you need to incorporate changes based on feedback from project stakeholders and usability testing.

<sup>4</sup> For an explanation of the term User Experience and how it fits within the lifecycle for Web development projects, visit [www.iconprocess.com](http://www.iconprocess.com) or call 1-866-STL-ICON and ask for the User Experience white paper.

If your use cases are already infected with user interface information, you should carefully review each use case to identify and then remove the inappropriate information. While doing this you may also realize that your user interface assumptions or decisions changed your requirements from what the users need to what is easiest to develop. Revise the use cases to reflect the true requirements.

## Abuse 5: Compulsive Architectural Disorder

This illness is similar to Compulsive User Interface Disorder. It is also caused by the desire to put useful but inappropriate information into the use case specifications.

### Symptoms

- » Your use cases are never finished.
- » Your use case specifications reference specific components, APIs, RPCs, stored procedures, or other aspects of the software design and implementation.
- » Software designers complain that the use cases do not reflect the latest component or class names.
- » Subject matter experts, users, or business team members cannot understand the use case specifications.

### Diagnosis

You have Compulsive Architectural Disorder. You include architectural or software design information in your use cases.

Left untreated, this form of use case abuse has two major side effects. First, it prevents you from having stable use cases until the software architecture is stable. It then forces you to update the use cases whenever the software architecture is significantly refactored. Second, it makes it difficult for team members with a business background to understand the use cases. This increases the project's risk of developing an application that does not meet users' needs. If your key subject matter experts or other stakeholders do not understand the primary requirements source, then they cannot detect or correct any errors.

### Treatment and Prevention

If your use cases already contain architectural design information, you should revise them by removing the solution-oriented information. When you are repairing the holes left by the removal of the architectural information, you may uncover unanswered requirements questions. The answers to those questions may cause you to rethink some of your assumptions. You may also need to redefine your actors. This produces shorter and more straightforward use case specifications.

To prevent this situation from recurring, remember that actors represent roles played by people, systems, and devices that interact directly with the application under development. If an actor representing something internal to the application emerges, you should question its appropriateness and not add it to the use case model. Further, you

should encourage the architect or software designers to use UML component, class, collaboration, or sequence diagrams to document their software design decisions instead of forcing that information into the use case specifications.

## Conclusions

There is no need for use cases to make your team sick or your project's progress unhealthy. The symptoms, diagnoses, treatments, and preventative actions described in this paper can immediately improve your project's health while keeping the benefits of a use case driven approach.

## For More Information

IconATG has highly experienced use case mentors and analysts available to provide short- or long-term support for your project.

### Training

>> [training.iconatg.com](http://training.iconatg.com)

Courses include:

- Defining & Managing Requirements with Use Cases
- Advanced Use Case Lab
- Requirements Elicitation & Facilitation
- Facilitated Use Case Workshop
- Testing Use Case Driven Projects
- Blended Agile Methodology (Use Cases and Agile PM)

### IconATG

>> [www.iconatg.com](http://www.iconatg.com)

### Consulting & Mentoring Services

>> [www.iconatg.com](http://www.iconatg.com)

### IconProcess, a Web development process

>> [www.iconprocess.com](http://www.iconprocess.com)

Please send feedback to [info@iconatg.com](mailto:info@iconatg.com)