BUSINESS ANALYSIS PRINCIPLES:

GET REAL USING EXAMPLES

Contributed by Mihaela Popescu, CBAP

INSIGHT

IIBA® International Institute of Business Analysis™

Whether you're working within a traditional waterfall initiative, an agile initiative, a hybrid approach, or another framework, adopting and applying business analysis principles will help you succeed as a business analysis professional.

The *Business Analysis Standard* outlines seven principles that guide an effective approach to excellence in business analysis:

- **See the whole**
- **Think as a customer**
- **Analyze to determine what is valuable**
- **Get real using examples**
- **Understand what is doable**
- **Stimulate collaboration and continuous improvement**
- **Avoid waste**

In separate articles, I explored the first three principles: *See the whole*, *Think as a customer*, and *Analyze to determine what is valuable*. In this article, I will explore the fourth: *Get real using examples*.

Examples clarify complex concepts and provide evidence to support claims. Business analysis professionals frequently use this principle to clarify requirements and create a shared understanding when engaging with stakeholders.

Tools and Techniques

What techniques can help us get real using examples? Here are some key techniques and approaches to consider:

- Use cases and scenarios
- Behaviour-driven development (BDD)
- Minimum viable product (MVP)/prototyping

I will now explore each of these in more detail.

Use Case and Scenarios

As business analysis professionals, we always strive to bring clarity to discussions with stakeholders—whether we're eliciting requirements, uncovering processes, or mapping workflows. One effective way to achieve this is by documenting use cases and scenarios. These describe the interactions between users (actors) and the product or solution, helping to identify the triggers for certain actions and the outcomes pursued.

Use cases not only clarify the goals behind the actions but also reveal the different paths an actor might take to reach those outcomes, through both straightforward and alternate routes. This ensures there are no ambiguities in how a system, solution, or product is used. Use cases are valuable for testing processes and validating requirements. They can also help filter "needs" from "wants," ensuring only true requirements make it into the implementation backlog.

Steps to Analyze and Document a Use Case

Identify and describe the product, system, or process being built or enhanced

Identify and describe:

Actors: Customers, users, stakeholders, or external forces interacting with the system, product, or process

Usage: Why and how these actors engage with it

Define the actors' goals, focusing on their "wants" rather than the capabilities of the product, system, or process

Create scenarios: Document the sequence of actor actions and the resulting effects, including the "happy path" (i.e. the successful scenario that accomplishes the goal)

Consider alternate flows: Identify other possible paths, including misuse or exceptions, to highlight areas requiring exception handling

Repeating these steps will help you identify all the use cases applicable to the system, product, or process.

Tip: Remember that there are two different use case types: *business* use cases (interactions between an actor and a process or business function) and *system* use cases (interactions between an actor and a system or application).

Behavioral Driven Development (BDD)

Behaviour-driven development (BDD) is another effective tool for applying this principle. It defines feature behaviour through plain-text examples, which serve as both acceptance criteria and test scenarios. These examples are created before development begins and, when automated, ensure the system continues to meet specified behaviours as it evolves.

BDD focuses on real-life scenarios provided by business users, customers, or stakeholders, describing how they use a system, product, or process to meet specific requirements.

BDD scenarios follow this format:

- **Given**: A context
- **When**: An event
- **Then**: An outcome

This structure ensures requirements are verifiable, testable, and clarified early through concrete examples, thereby reducing ambiguity.

Sample Template

- **GIVEN** (context),
- **AND** (further context) — *optional*
- **WHEN** (action/event)
- **AND** (further action/event) — *optional*
- **THEN** (outcome)
- **AND** (further outcome) — *optional*
- **AND** (further outcome) — *optional*

Here's an example:

Feature: ATM money withdrawal

- **Scenario**: Account has insufficient funds
  - **Given**: I'm overdrawn
  - **And**: The ATM has sufficient cash available
  - **When**: I request $20
  - **Then**: I receive no money
  - **And**: My card is returned

This is just one of many possible scenarios for this feature. Writing additional scenarios using BDD helps clearly define its business rules and requirements.

A picture is worth a thousand words. The more examples you include when eliciting requirements, the clearer and more aligned everyone's understanding will be—from the business rules to the final product.

Minimal Viable Product

First introduced by Eric Ries in Lean Startup, the minimum viable product (MVP) is another powerful tool. According to Ries, an MPV is *"the version of a new product which allows... to collect the maximum amount of validated learning about customers with the least effort."*

Like prototypes in traditional waterfall methods, an MVP helps teams explore whether a product meets customer needs. However, while prototypes often showcase a semi-functional version, an MVP is built incrementally to test small ideas and validate whether they deliver real value. Essentially, each MVP acts as a concrete example of a feature or functionality, allowing it to be tested and validated before investing significant time or resources in full-scale development.

For example, imagine a customer wants a way to get from point A to point B. You might start by building a simple scooter. Based on continuous feedback, you might then evolve it into a bicycle, then a motorcycle, and eventually a convertible car, with each step validating what the customer truly needs along the way.

The power of MVP is that it relies on elicitation, feedback, and iteration to arrive at clear examples of the requirements. With this constant feedback, teams can prioritize what truly matters, compromise on non-essential elements, and clarify which features or aspects are unnecessary for success—hence the term "minimum."

Summary

The *Get real using examples* principle is about clarifying why a requirement exists and illustrating it with real examples of needs, business rules, or desired outcomes. It also emphasizes validating the requirement, ensuring all use cases and scenarios are covered. While many techniques can achieve this, the three approaches I've explored are among the most common for business analysis professionals to ensure both their own success and the success of the initiatives they support.