What's a "Good" Use Case?

The following are criteria to keep in mind when developing and reviewing your use cases. We present these criteria not as ironclad rules, but as guidelines that have been useful to experienced use case developers in the past.

The use case developer is free to create use cases that do not satisfy some of these criteria; however, we strongly suggest that such use cases be challenged. If a "nonconforming" use case is left intact, a good reason for deviating from the criterion should be identified and documented.

If these principles are not followed, you run the risk that the use cases are too big, too small, confusing, at the wrong level of abstraction, and/or difficult to use for estimating implementation effort.

The examples below are drawn from a Hotel Reservation domain.

Criterion / Guideline	Motivation / Implication	Good Example(s)	Bad Example(s)
Likely require at least 3-5 distinct transactions to fully implement, but more important the use case must provide a meaningful result to the end-user.	Use cases that only deal with a single straightforward CRUD operation are likely to be too simple. Note that we are not suggesting that the database and transaction design be done from the use case description. However, most analysts should have a flavor for what a transaction (clarify transaction as a transaction meaningful to an end-user) involves, so this principle can be used to get a reasonable initial granularity for the use case. The word transaction may imply CRUD transactions. We want you to think business transactions or business interactions.	A use case "Create reservation" which includes identifying the customer (and creating the customer if necessary), entering reservation details, determining and calculating discounts, and storing reservation details. We would not include updates and cancellations in this unless you change the use case name to Manage Reservation.	A use case titled "Change Hotel Room" which includes accepting a reservation or confirmation number and only changing the hotel room.
Always consider variations of a single theme.	Use cases should incorporate variations of a business transaction. Common mistake it to breakup these variations into multiple use cases. If the variations are complex you may wish to leverage alternate paths / flow in your use cases.	A use case "Customer checks-in/checks out." Because of the size of these business transactions they were collapses into one use case.	A use case "Customer checks-in with a reservation," "Customer check-in without a reservation," "Customer checks out early," and "Customer checks out on time."

What’s a “Good” Use Case?

Criterion / Guideline	Motivation / Implication	Good Example(s)	Bad Example(s)
Describe interactions and mechanisms, not policies.	The text in a use case description is intended to show actor/system interaction, not all details of the policies and validations used to implement the interaction. We want you to think of the use case descriptions as a topical dialog between actor and system. **Note:** It still may be very useful to obtain some of this information from subject matter experts and link the use case description to other specifications of the policies, which could be decision tables, formal IF-THEN rules, algorithms, GUI fields and formats, etc. The point is that this information should not be expressed in the use case text when another means of expression is more appropriate (e.g., Supplementary Specification, Business Rules Document, Wireframes, etc.).	“The customer provides a method of payment (optionally list the current payment types) to hold a reservation”	“The system computes a checksum for the credit card according to the following formula…”
		“The system validates the reservation to ensure that all required information has provided, and then stores it.”	“The system checks the following referential integrity constraints …, ensures that the entered ZIP code has 5 digits, and saves the reservation to the database.”
		“The system calculates the net price for the room, including any discounts, and displays the total price.”	“The system first applies any percentage discounts, then any specific amount discounts, then any combination discounts. A maximum of three discounts of two different discount types can be applied. The net price for the room cannot be < 0.”

What's a "Good" Use Case?

Criterion / Guideline	Motivation / Implication	Good Example(s)	Bad Example(s)
Exclude user/system interface implementation choices.	For flexibility and reusability, the use case description should be independent of any particular technology. Sometimes sample GUI screens are prototyped or whiteboard mock-ups are created during use case modeling as a visual aid, but this is a separate activity.	"The customer provides or locates a reservation number"	"The customer selects a reservation number from a pull-down list…" or "The following wildcard characters may be used in the search…."
		"The customer provides identification information to obtain access to the guest room"	"The customer slides his room key card through the room card reader…"
		The system asks the credit bureau to verify the credit card number..."	"The system sends the credit card request using the TCP/IP protocol. The inquiry has the following 256-byte format…"
5-10 pages are typically adequate to describe a use case.	Page length is a rough indicator of the complexity of the functionality provided. (Such guidelines are notoriously subject to manipulation via font size, spacing choices, etc., but it is still a useful commonsense principle if not abused).	See www.williamnazzaro.com and than select Modeling to view good use case examples.	1-page or 50-page+ will draw our attention for a review of the use cases.

What's a "Good" Use Case?

Criterion / Guideline	Motivation / Implication	Good Example(s)	Bad Example(s)
One <initiator> actor.	If multiple <initiator> actors it may be too big (more like a complete workflow) or we feel you may not have identified the proper actor abstraction. Note: This is a guideline. We have seen situations where multiple <initiator> actors may make sense, but this is the exception rather than the norm. It also may be indicative of not leveraging generalization with actors.	Use case: Customer Checks In / Checks Out (Actor: Front Desk Clerk) Generalization Example: (Actor: Front Desk Clerk and Manager, where Manager inherits from Front Desk Clerk)	Use case: Customer Checks In / Checks Out (Actor: Associate Desk Clerk, Desk Clerk, Trainee)
Include major business exceptions and business exception handling focusing on resolving the business interaction.	This will be extremely helpful later during your iteration when use cases drive the creation of test cases. Should always state how the exception is resolved and if the use case continues or abandons its goal. Note: A large exception may yield an alternate path or flow.	Exception: Room Not Available: the customer is informed and must provide a different room type for the reservation to proceed.	Exception: Room Not Available (with no exception handling specification).