**Feature Slicing**

Few of the approaches for slicing a Feature are:

* Implement Partial Acceptance Criteria
* Partial Business Value (use Lean Start up)
* Defer Optional Behavior
* Separate Different Channels
* Address Different User Groups Individually
* Incremental Data Sourcing
* Isolate Special Variations
* Breaking out into Functional and Enablers

**Implement Partial Acceptance Criteria** approach encourages you to slice a Feature based on partial achievement of Acceptance Criteria in a series of smaller features.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Benefit hypothesis** | **Acceptance Criteria** |
| Accepting online payment at our e-commerce website | People can purchase goods at our website. | 1. People should be able to use Visa, Master Card, Discover, and American Express 2. People should be able to use ACH check (cheque) |
| Accepting online payment at our e-commerce website via check (cheque) | People without plastic can purchase goods at our website. | 1. People should be able to use ACH check (cheque) |
| Accepting online payment at our e-commerce website via check (cheque) | People with plastic can purchase goods at our website | 1. People should be able to use Visa, Master Card, Discover, and American Express |

**Partial Business Value** approach based on Lean Startup (<https://en.wikipedia.org/wiki/Lean_startup>). A Feature is sliced into smaller ones. Each resulting Feature delivers part of initial Business Value. This approach enables organization to test waters.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Benefit hypothesis** | **Acceptance Criteria** |
|  |  |  |
|  |  |  |

**Defer Optional Behavior** approach encourages to split a big feature into smaller ones based on deferring the not so urgent functionality.

Does the feature include lots of optional behaviors? Consider making the optional behaviors separate features to be implemented once the core functionality is delivered and more learning has occurred. You may find that these optional behaviors are not needed.

**Separate Different Channels**: Could the feature be released incrementally to support different channels, routes to market, or technologies? For example, different operating systems, or the different channels for retail banking, web, mobile app, and in-branch banking. It is logically the same feature, however, the implementation for each channel could be sliced into a different feature (you may discover latter that you do not need all).

**Address Different User Groups Individually**: Do different groups of users want different types of functionality? For example, the feature may appeal demographics, however, each group may want to use the feature in a different way. In this case, slicing the feature on just a subset of functionality for each demographic may appeal to the interest of the more important groups earlier.

**Incremental Data Sourcing**: Is all the data needed before any benefit can be provided? Perhaps the data can be consumed incrementally or sourced from existing secondary sources?

**Isolate Special Variations**: First focus on the popular / high volume cases then add the more specialized corner cases as additional features. You may discover that the value of the additional corner cases is low and should not be implemented.

**Breaking out into Functional and Enablers**: Business features often rely on the same underlying system behaviors, making the implementation of the first set of features appear large and complex. Slicing them into business features and enablers can reduce the risk and size of the features.