# Strategy

If you need to validate what the organization is there to do.

Vision:

## Problem statement

Use standard format of <this problem> affects <these people> with these <specific bad results>. Our solution will help by providing <general good results>.

The access to banking services is restricted to bank hours affecting the majority of our bank customers by having to wait to the next business day, adding more staff or hours, or finding another solution. Our solution will allow the bank to provide services at any hour and in locations where a bank branch is not located reducing overhead costs. Replacing teller and not having bank open with automation of transaction external to bank reduces overhead and improves customer experience.

### Related problems

During problem definition, other problems can easily come up and should be captured.

## Constraints

The rules that will constrict either the scope, the budget, or the timeframe of the project. Risk is a probability that these will be in force. Assumptions for design will be to take the risk or not and trace back to these.

Infrastructure, technology, laws, etc.

## Stakeholders

A list of the people and systems that have an impact on the project requirements. Actors can be extracted and grouped from this list.

### People

|  |  |  |
| --- | --- | --- |
| Name | Position / Dept | Notes |
| I. Hafmunee | Bank president |  |
| Gee Kee Tekhed | CIO of bank |  |

### Systems & data sources

# Analysis – system level

Post-elicitation stage done with all analysts after the initial interviews and document reviews. Do this first.

## Actors

First step in analysis is to brainstorm and validate these roles and systems.

* Customer
* System
* Maintenance

## Use case names

Second step in analysis is to brainstorm and validate scope at the goal level.

### System use cases

Only those use cases which start after the software is running and an actor directly interacts with the system.

Grouped use cases are either system or business use cases and can be groups of value/goal level use cases or groups of partial use cases like used in this example. Also common in this section are the Manage Entity type of use case which expands into the CRUD use cases. e.g. Manage Account = Create Account, Read Account (Display details of account), Update Account, Delete Account.

* Withdraw cash
* Deposit funds
* Transfer funds
* Request balance
* Display idle screen

### Business use cases

Any use case that does not completely exist as an interaction between software and the actor.

* Fill cash
* Reload receipt paper
* Remove deposits
* Stock envelopes
* Do system preventative maintenance
* Do ATM session (startup and shutdown)

### Use case diagram

Here would be a good place to put a Visio diagram(s) showing the relationships of all the use cases.

## Use case groups found

Use cases can be structured to roll up to a group name. Helpful to reduce complexity.

* Do transaction
  + Withdraw cash
  + Deposit funds
  + Transfer funds
  + Request balance
* Do maintenance
  + Fill cash
  + Reload receipt paper
  + Remove deposits
  + Stock envelopes
  + Do system preventative maintenance

## Partial goal names found

Use cases can be structured to drill down to a named sequence of tasks not ending in a goal or repeatable. Helpful to identify reusable parts.

* Retain card

# Project management

## Prioritization

Add other columns of ranking variables to mix in with weightings if useful. Keep the values simple.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Impact | \* Urgency | = Priority |  |
| Use case name | Business usage /  market potential 3=100-67%, 2= 66-34%, 1=0-33% | Business need / perceived value 3=exec/high,  2=mgmt/med  1=staff/low) | Multiplied  Result | In a group, ask for the top 3 and  count the  total. |
| Withdraw cash |  |  |  | 5 |
| Deposit funds |  |  |  | 4 |
| Transfer funds |  |  |  | 3 |
| Request balance |  |  |  | 3 |
| Display idle screen |  |  |  | 0 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Project Iterations

Move the use cases with priorities down to a schedule so that you can see what kind of sprints (Agile) you will have.

### set up

There needs to be a software stage to set up and get all the pieces initially running as a framework. It’s called stubbing out the system or mocking the system depending on what you need. It may be combined with the 1st iteration.

### 1st

The first iteration has to be the most complex use case or one that would likely fail. Stop the project early if this doesn’t work.

* Withdraw cash – complex

### 2nd

* Deposit funds - medium
* Request balance - easy

### 3rd

* Transfer funds - complex
* Display idle screen - easy

# Analysis - detail

Final versions of the requirements documents used for walk-throughs with the main stakeholders.

## Use Cases

### Summary of all use cases

A summary of the use cases in bullets or numbers serving like a table of contents. Summary focuses on inputs and outputs.

### ID – Use case name

Information about the use case, metadata

**Author** –

**Date written** –

**Actors** –

Does not include those participants involved but do not initiate/trigger this flow of events.

**Related systems** –

Sometimes called (supporting actors)

**System** –

Would not be used if this were a business use case

**Priority** –

Get this from the priority chart.

**Type** –

Options are: goal | partial goal | group of goals | group of partial goals

**Design constraints** –

Useful when you don’t use just business terms e.g. web site is required, SQL Server is required, location must be…

#### Pre-conditions

Rules for beginning this use case: state of system prevents usage, must be testable

#### Course of Events

The sequence of tasks in conversation format between actor and system. Start each number with the system except the trigger and combine actor responses to system events. Rules are placed under the task unless they can be reused and then they are referenced and placed in a separate file.

1. The use case starts when the actor …
2. The system responds by … The actor does something else.
3. The system validates something… The actor responds
   1. **RULE – Name:** The conditions that must be met for the validation.
   2. **RULE – Name:** The conditions that must be met for another validation.
4. The system responds by … The actor does something else.
5. The system does something…

#### Extension points – optional

Sequences that return control back to the course of events after finished.

* Sequence name (#) –

#### Extension points – errors, exceptions

Errors occur at any point where there is a validation of a rule.

Errors occur at communication to other systems.

* Sequence name (#) –

#### Post-conditions

Is it really important to review the necessary outcomes of this use case? If so, summarize here.

#### Notes/ Special Requirements

Any kind of quality, capacity, security, availability, disaster recovery information.