# Key rules of our architecture

## Growing your application.

We start from a green field. We start with a minimal harness to run our first test successfully. What I mean with that is that although our test needs to fail first we have to foresee a testing framework to run our test in. We need to be able to run our test and some other non-functional requirements to work fast and get the necessary feedback. Every time a new test is introduced and to make this test succeed we foresee that the absolute minimum is extra included.

The second requirement is that the implementation of each test

## Technology independence

Technologies, libraries and other functionalities serve the tests and not the other way around. When a test requires for example that data will be stored in a database we introduce a database connectivity library (in our case it will be Entity Framework). We follow the advice and introduction steps that has been given in this document to implement it and go as quickly as possible to make our test succeed. We keep the principles in mind and we develop against them.

## Tests first

Never introduce more code then is needed to either get a test to a green state. Do the introduction steps of a certain technology, or during refactoring to tighten already written code.

It is not easy to not introduce code because you know it will have to be implemented in the near future. Keep it in mind and introduce it when a test requires it.

# Analysis and Design

We use Sparx Enterprise Architect version 12 as our design tool.

## The design process

We have a particular design process that we strictly need to follow when we have a new requirement that needs to be implemented.

Requirements

With the project of EAS we always start from a requirement. If a requirement is too complex or too large to handle we need to split it up in sub-requirements. You have to be careful you do not already go too much into detail and define use cases instead of sub-requirements. There is no clear definition when it is a sub-requirement or a use case but when there is an external system (an actor) that could call the definition we know that it is probably a use case.

*“If it is not clear enough to comprehend try to split it up in edible pieces.”*

This is an advice that works everywhere especially when you are working with a complexity as we have here in the requirements.

In some cases the requirements need to be decomposed and being studied in more detail with all stakeholders. For that we have the Archimate Business Layer Diagram where you can design the business processes being used.

Requirements

Archimate

Business Layer

Take notice that the business layer diagram is a complementary diagram and its sole purpose is to clear out certain ignorance.

When you have finished with your requirements and your business layer diagram when needed it is time to create the features and your use cases.

Requirements

Features

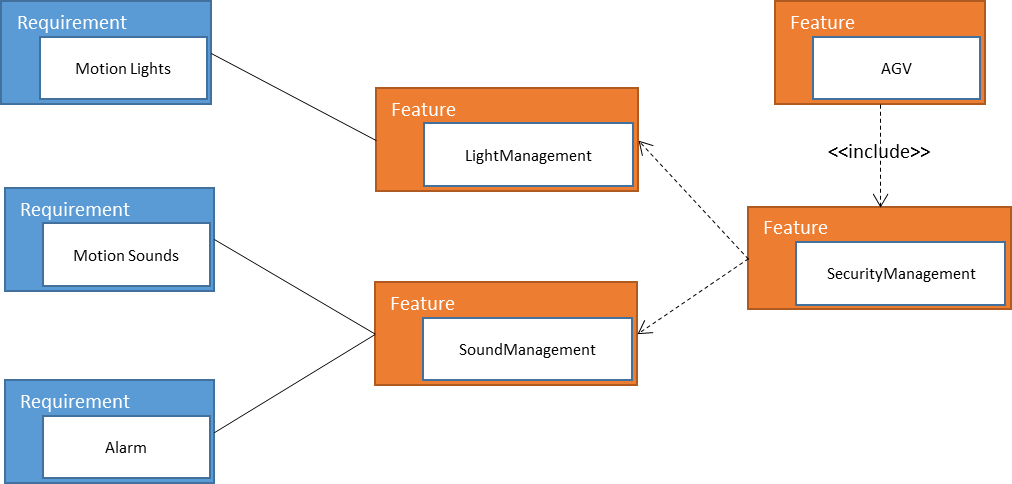
Use Cases

Steps in clear words

Requirements: What are the needs?  
Feature: Is an answer on the requirement. Composing the requirements in features.  
Use case: Happy flow to complete the requirement.

Several requirements can be composed together in features and can lead to effective groups. If we look at it from a technical view we can see that some requirements can lead to build a certain service, domain, screen or even higher to a separate application. It is possible that one requirement can be part of more than one composition or none.

An example says more than a theoretic description and so you get three requirements that we define the features for and relate them to already existing features.



We have three requirements (shortened in their description for demonstration purposes) motion lights, motion sounds and alarm. We draw a diagram to define the features like the example in the picture above (we keep using the requirements diagram for that [[1]](#footnote-1)). How they need to work and what are the consequences will be described in the use cases but here we are going

1. See the step-by-step guide for EA how exactly this example is implemented. [↑](#footnote-ref-1)