Contents

[Design Summary 2](#_Toc467682606)

[Required Components 2](#_Toc467682607)

[Decisions 3](#_Toc467682608)

[Getting started 3](#_Toc467682609)

[Old Design 3](#_Toc467682610)

[New Design 4](#_Toc467682611)

[Journal of time 6](#_Toc467682612)

[Issues 8](#_Toc467682613)

[General 8](#_Toc467682614)

[Caliburn Micro 9](#_Toc467682615)

# Design Summary

## Required Components

* Straight away from the description the following was required:
  + A Farms Repository
  + A Millers Repository
  + A Paddocks Repository
  + A Prepolulate Repository Component
  + Farm entity
  + Miller entity
  + Paddocks entity
* After drafting the master/details view I discovered the following:  
  (*see image below ‘First Design’*)  
  + Views/Models  
    - Master/Details view   
      *(in first implementation used to learn Caliburn Micro)*  
        
      Refactoring lead to:
    - Shell  
      *(only containing ContentControls and all the required models)*
    - Farms  
      *(shows Farms as a list and allows to select one farm or add a new one)*
    - Farm  
      *(shows the details of a farm, allows editing and saving)*
    - Paddocks  
      *(shows paddocks of a farm including total area, allows to add, remove and edit)*
    - FilterFarmsByFarmName
    - FilterFarmsByMiller
  + Services  
    (after refactoring)  
    - ShellService  
      (to instantiate other singleton components)
    - FarmsService  
      *(get and save a farm)*
    - MillersService  
      *(request all millers)*
    - PaddocksService  
      (add and remove paddock, get all paddocks for farm)

# Decisions

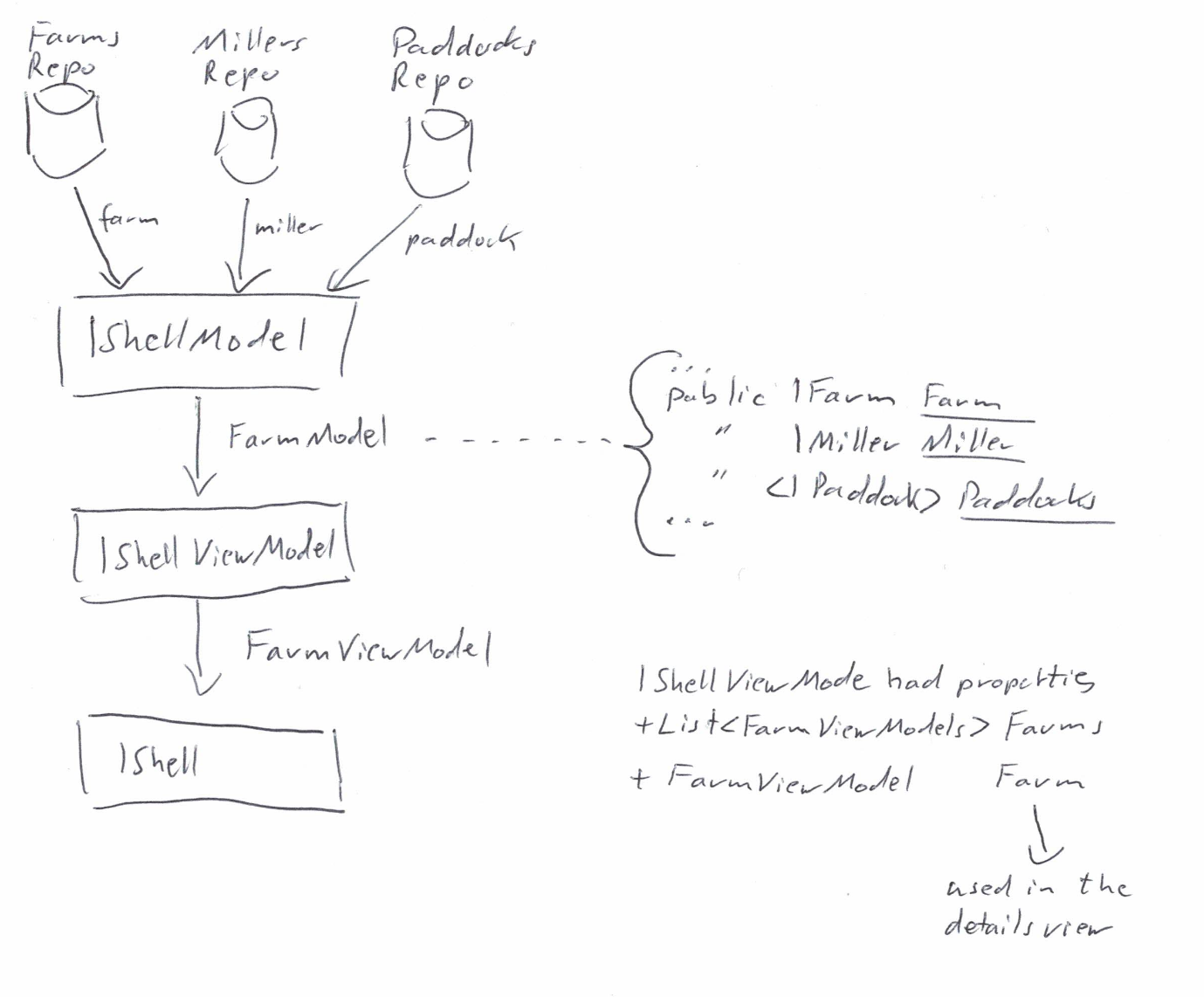
## Getting started

* Draft a design for the UI
* Split the UI into different components
  + First implementation used master/details views only
  + Refactoring added new views/models
* Read documentation of Caliburn Micro and did examples
* Searched and found example for master/details view
* Used example to implement first version of application

**Attention**: I already had an event/message base solution in my head but trying to do to many things at the same time will not work. - Here learning about Caliburn Micro and creating the perfect solution in one go!

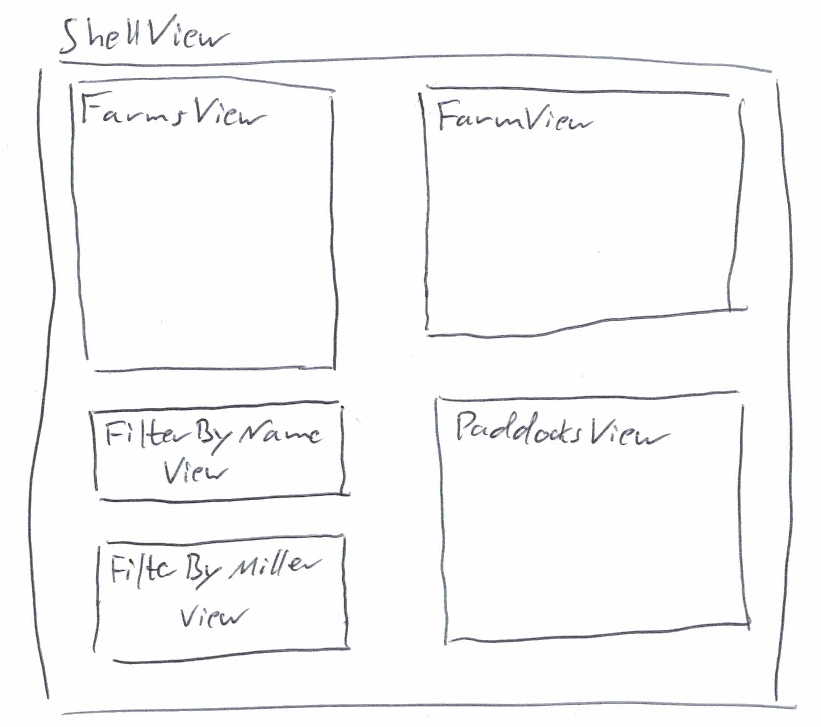
## Old Design

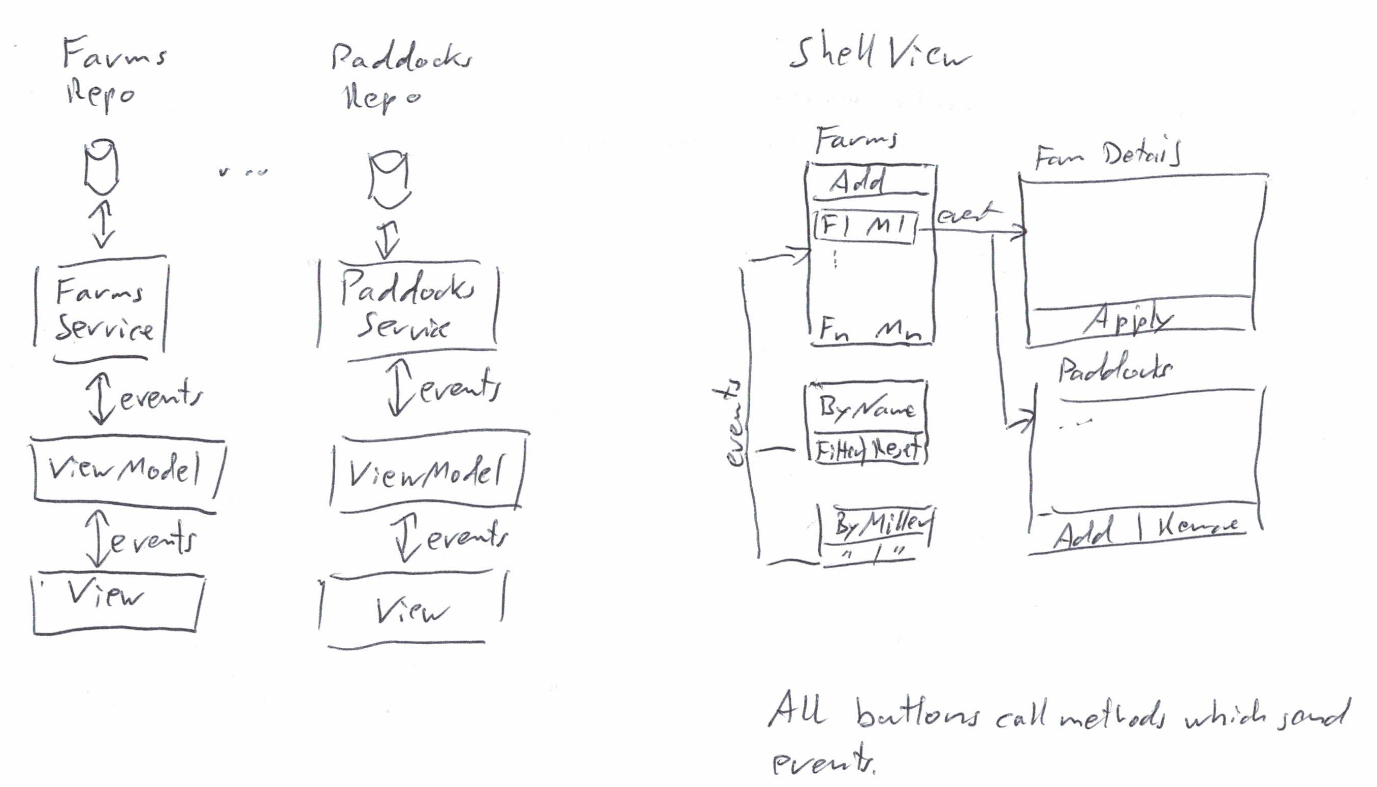
* Load everything into memory
* Create something working first, refactor/improve later



## New Design

* Load on data on-demand
* Use different user controls (views/models)
* Use events to communicate
* Loosely coupled components
* Remove ‘messy’ code from first implementation – Go for SOLID, LEAN, DRY





# Journal of time

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Time in hours** |
| First design | Read coding test description  Think about possible implantations  Draft view  Split-up into sub-views  Events/Message idea | **2** |
| Learn | Read Caliburn Micro documentation (basic) | 2 |
| Learn | Google for examples of Caliburn Micro | 2 |
| Learn | Google for master/detail example for Caliburn Micro | 2 |
| Data Access | Implemented required repositories and unit tests | **2** |
| Implementation  *First Version* |  |  |
|  | Master view/model | 4 |
|  | Client view/model | 4 |
|  |  |  |
| **Implementation *Final Version*** |  |  |
|  | Farms view/model | **1** |
|  | Farms service | **1** |
|  | ‘Filter By Farm Name’ view/model | **1** |
|  | ‘Filter By Miller’ view/model | **1** |
|  | Farm view/model | **1** |
|  | Farm service | **1** |
|  | Paddocks view/model | **1** |
|  | Paddocks service | **1** |
|  | *Total Time spent to get to an implementation I liked* | *26* |
|  | **Total time spent for ‘Final Version’** | **2+4+8** |

Available Farms (Millers)

Add New Farm

Farm Name 1 (Miller 1)

Farm Name 2 (Miller 2)

Farm Name n (Miller n)

…

Filter by Farm name

Farm name

Filter

Reset

Filter by Miller name

Filter

Reset

Miller 1

\/

Farm Details

Field Code

Field Code 1

Name

Farm Type

Name 1

Cane

\/

Harvested

Date & Time

Pick

Miller 1

\/

Miller

Farm’s Paddocks

Field Code 1 (1 hectares)

Field Code 2 (2 hectares)

…

Field Code n (n hectares)

Add

Remove

First Design

## Issues

### General

* **Issue**: WPF application is not covered by unit nor integration tests.  
  **Fix**: More time, done tests for data access.
* **Issue**: First implantation was loading all data into master view model. – Not a good idea, but working.  
  **Fix**: IEventAggreator to send events to select/load farm and paddocks on demand.
* **Issue**: Had to merge data from Farm and Miller so that I can show and filter farms by name and miller id.  
  **Fix**: Added a new FarmMillerDto which a service populates and sends to the view.
* **Issue**: How do I tell the other views that the selected farm has changed?  
  **Fix**: Send an event using IEventAggreator.
* **Issue**: Add new farm should be its own view/model.  
  **Fix**: Create new classes, but ran out of time.
* **Issue**: ShellService has to depend on repositories and other service to make sure the components are created in the container.  
  **Fix**: Use Lifestyle.Startable for these components.
* **Issue**: Where to calculate total of are in hectares. – First it was in the ‘Farm Details’ view.  
  **Fix**: Moved into ‘Paddocks’ view, were it belongs
* **Issue**: Resize application does not properly resize UI controls.  
  **ToDo**: Change control/containers. – No time left to do it!
* …

### Caliburn Micro

* **Issue**: Never used it before so I had to learn it before I could start coding. Documentation covers simple examples but not really what I needed at the beginning. By the description of the coding test I needed an application split-up into master/details.   
  **Fix**: Googled and ended up with a first version using two models and views. Code was messy!
* **Issue**: MEF didn’t work as expected, failed silently. Nor sure if I was doing something wrong or the configuration.  
  **Fix**: Switched to SimpleContainer, which is a bit better. But still want auto registration of components like MEF.
* **Issue**: ContentControl and views didn’t work with MEF. I needed that to split up the complex master/client view into smaller pieces.  
  **Fix**: Switched to SimpleContainer and made sure to depend on view models in ShellView.
* **Issue**: Split-up master/client view but how do the views communicate?  
  **Fix**: Learn how to use IEventAggreator
* …