| To (department acronym name) | | For information (department acronym name) | |
| --- | --- | --- | --- |
|  |  |  |  |

SesammTool2 SAD

Contents

[1 Introduction 2](#_Toc27747459)

[2 Architecture 3](#_Toc27747460)

[2.1 Presentation Layer 3](#_Toc27747461)

[2.1.1 View 4](#_Toc27747462)

[2.1.2 ViewModel 4](#_Toc27747463)

[2.1.3 Model 4](#_Toc27747464)

[2.2 BusinessLayer 5](#_Toc27747465)

[2.2.1 ObjectModel 5](#_Toc27747466)

[2.3 Datalayer 8](#_Toc27747467)

[2.3.1 Services 8](#_Toc27747468)

[2.3.2 Db Access module 8](#_Toc27747469)

[2.3.3 OAS Access module 8](#_Toc27747470)

[2.3.4 File Access module 8](#_Toc27747471)

[3 Third party libraries 9](#_Toc27747472)

[3.1 DevExpress 9](#_Toc27747473)

[3.1.1 Design decision 9](#_Toc27747474)

[3.2 yFiles for WPF 9](#_Toc27747475)

[3.2.1 Design decision 9](#_Toc27747476)

[3.3 log4net 9](#_Toc27747477)

[3.3.1 Design decision 10](#_Toc27747478)

[3.4 Prism 10](#_Toc27747479)

[3.4.1 Design decision 10](#_Toc27747480)

[3.5 Ionic.Zip 11](#_Toc27747481)

[3.6 DotNetRdf 11](#_Toc27747482)

[3.6.1 Design decision 11](#_Toc27747483)

[3.6.2 Implementation 11](#_Toc27747484)

[3.6.3 Dependencies 11](#_Toc27747485)

[4 Integrations 11](#_Toc27747486)

[4.1 JIRA 11](#_Toc27747487)

[4.2 OAS 11](#_Toc27747488)

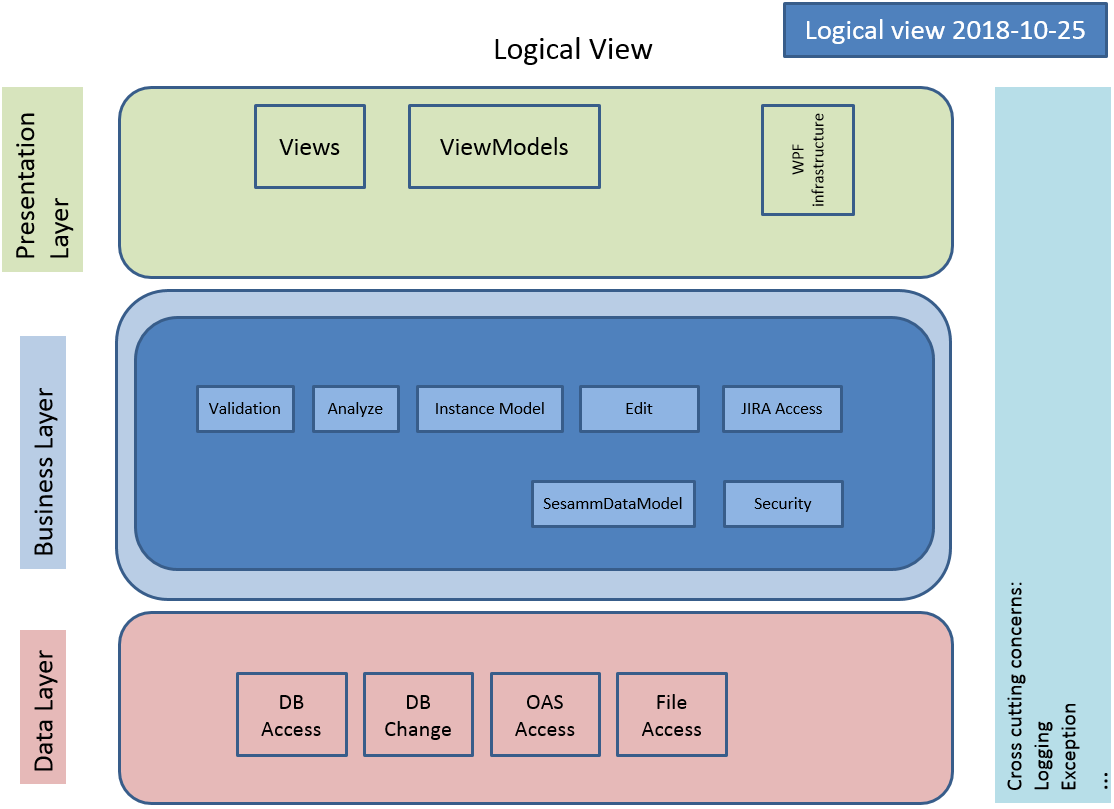
[5 References 11](#_Toc27747489)

# Introduction

This is the Software Architecture Document for SesammTool2. The purpose is to show the architecture, principles used and design decisions.

# Architecture

Sesammtool2 is based on a three layered architecture with well-defined interfaces between layers.

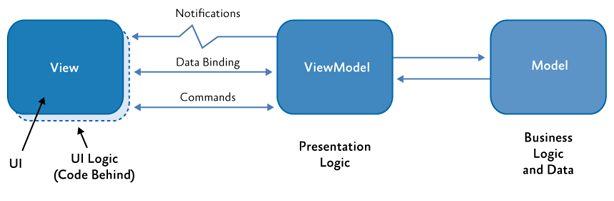


## Presentation Layer

The presentation layer uses the MVVM architectural pattern [MVVM] to separate the GUI and the presentation logic from the business logic. MVVM is Microsoft’s recommended architectural pattern when developing WPF-applications.

We use the Prism framework that supports building loosely coupled, maintainable, and testable XAML applications in WPF. More info about Prism can be found in the section “Third party libraries”.

The following illustration shows the three MVVM parts and their interaction [PRISM].



### View

The View should only contain UI-elements and UI-logic (code behind). The View communicates with the ViewModel by sending and receiving commands. Both one way and two ways data binding is possible between the View and the ViewModel. Notifications are sent from the ViewModel to the View.

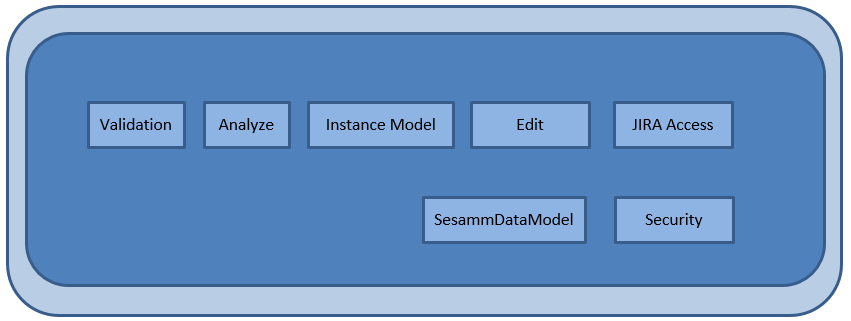
### ViewModel

The ViewModel requests data from the Model or stores data to the Model (that includes Data). It receives data from the Model upon a request from the ViewModel. The ViewModel communicates with the View by sending and receiving commands. Both one way and two ways data binding is possible between the ViewModel and the View. Notifications are sent from the ViewModel to the View.

### Model

The Model in MVVM corresponds to the Business Layer. The Model sends data to the ViewModel upon request from the ViewModel. It also stores data sent from the ViewModel – either only to its internal representation of the data when persistence is not needed, or to the Data layer when persistence is needed.

## BusinessLayer



### ObjectModel

The different entity states when creating/editing/etc… objects are shown in the figure below. The following is a description of some of the properties that are not self-explanatory.

**Entity state Created**

All properties are changed, because a new object is saved in the database.

**Entity state Commited**

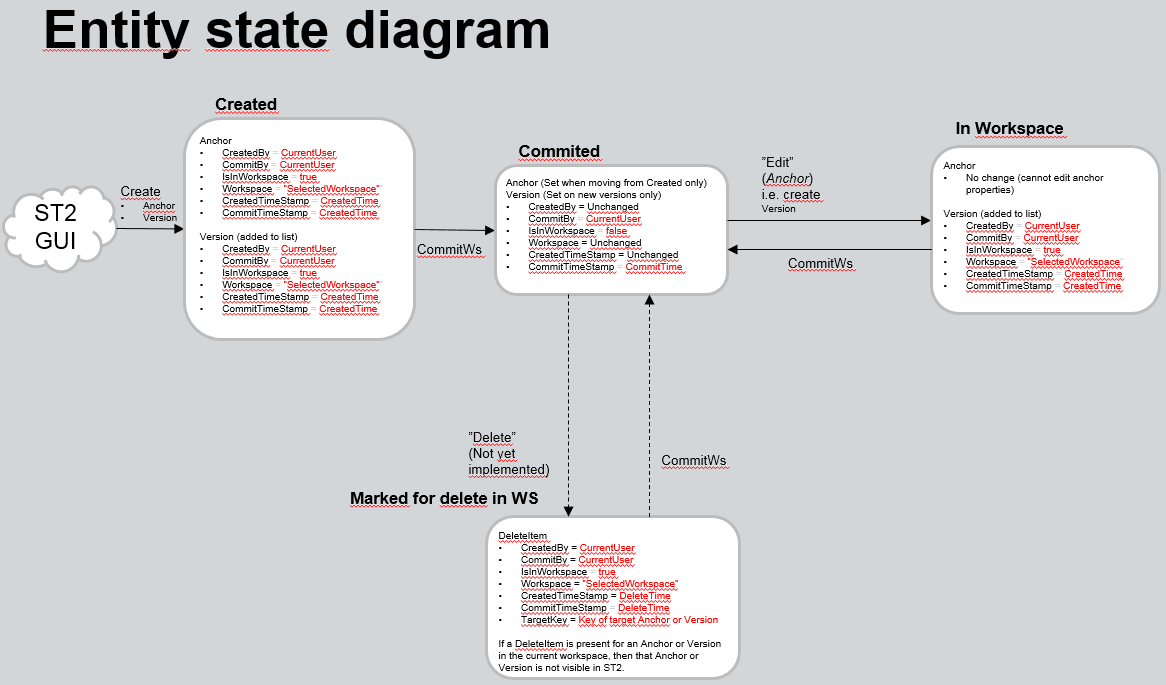
The CommitBy property is changed, because another user has the right to Commit. When using partial commit, it is not allowed to check in Version entities if Anchor is still in workspace.

**Entity state In workspace**

When editing, all properties are changed on the Version object.

**Entity state Marked for Delete in WS**

The property Workspace is used to track who has deleted the object.



## Datalayer

The data layer is divided into a number of modules that handles access to a specific data source. On top of the modules there is a Services interface layer that exposes services to upper layer. A single service might use data from multiple modules/data sources if that is needed.

### Services

The service interface supposed to be used by external users of the DataLayer. Exposes limited interfaces of data layer internal modules. One service interface might depend on functionality from one or more internal modules. The service interface is supposed to expose internal interfaces in a consistent way.

### Db Access module

This module is responsible for handling CRUD access to the Sql-server database. Microsoft Entity Framework 6 is used for ORM-mapping. We switched from “model first” to code first due to problems with the design tool that crashed. Another reason for the switch was the upcoming EF 7.0 that only supported “code first”. The data model is based on a concept called “Anchor modelling”.

### OAS Access module

**Note!** This module is only approved for limited use. Check before building functionality depending on this. A copy of this code is used by administrators of SesammTool1 to copy data read from OAS and persist in SesammTool database.

Module responsible for accessing OAS using REST-interface. Currently only supports reading Functional structure.

### File Access module

Groups various data sources/sinks that are file-based, both read and write access are included.

#### Can

Supports read and write of CAN-data in json-format. Supports write of CAN-data in word-format.

#### Dbc

Supports read and write of CAN-data for single can-segment (vector proprietary “dbc”-format) in dbc and json-format.

#### FunctionalStructure

The xml-format has now been fully agreed with YS department developing SDP3.

Supports read and write of functional in xml-format as specified by SDP3-tool.

#### Msc

Supports read and write of msc-data in xml, json and mpr-format.

#### Sops

Supports reading FPC-block from sops-file.

# Third party libraries

## DevExpress

WPF user controls are used to speed up the development.

### Design decision

We chose DevExpress after having evaluated other WPF User Control libraries and chose DevExpress because it covered our needs and seems to have great customer support.

## yFiles for WPF

Diagramming library used for AED (static diagrams) and dynamic diagrams.

### Design decision

We chose yFiles because it has support for orthogonal layout and can handle large graphs with good performance and good looking automatic layout (including routing) of objects. During evaluation we only found one competitive company (Tom Sawyer Software). We tried a layout of 100 nodes and 200 edges and both yFiles and Tom Sawyer Software performed almost equal in regards of performance and layout quality. We chose yFiles because Scania had previously used yFiles before in the “Architecture Browser” prototype with good results. yFiles support was also very good.

## log4net

The purpose of logging is to help developers find out what went wrong and to provide logging of metrics (performance, volume, etc…).

A logged object consists of:

* Timestamp that the logging framework is responsible for.
* Log messages should be written in English. These messages should NOT be localized.
* Stack trace (only valid for exceptions) to be able to pinpoint the code line that caused the exception.

The log should be written to disk. The maximum size is 9 MB to allow the customer to mail the log file if necessary.

### Design decision

We chose log4net since it seems to be the de facto open source standard. PSM also uses log4net. log4net also supports logging of objects. A comparison of different logging frameworks is described in [LOGG]. Enterprise Library Semantic Logging library does not support logging of objects and seems to be overkill at the moment. Prism has built-in logging support but is not an alternative, since we do not want references to Prism in the Data Layer.

## Prism

A Microsoft WPF framework to support common functionality in WPF applications. It is now open source [PRISM]. We are using IEventAggregator to handle the events sent in SesammTool and UnityBootstrapper to support dependency injection. IRegionManager is used to attach objects to regions. IModuleCatalog, BindableBase, ObservableObject and DelegateCommands are also used.

### Design decision

We chose Microsoft Prism because it seemed to be the most complete framework to help developing WPF applications. It comes with good documentation and examples. Microsoft was also very active developing it in contrast to some of the other evaluated frameworks.

## Ionic.Zip

## DotNetRdf

The purpose of rdfDotNet is to enable possibility to work with rdf data in SesammTool. Currently use is limited to create an in-memory rdf-graph representing the communication data model. The graph can be exported to a turtle-file (.ttl). The library also has support for SPARQL-queries towards the in-memory graph. That functionality might be used in the future.

The library is also used for creating an in-memory AE-graph representing the filtered result from Diagram Window. The graph is then serialized to turtle and sent to visualizer-component to be visualized.

### Design decision

The library was chosen because it seemed to support the current needs for SesammTool and at the same time support possible future needs. The library is also used by SesammSpecifier (now a deprecated tool).

### Implementation

The functionality is injected using Unity IOC-mechanisms so that we avoid hard dependencies towards the library.

### Dependencies

The library is installed using NuGet and has dependencies on the following libraries:

* VDS.Common
* Newtonsoft.Json
* HtmlAgilityPack

# Integrations

## JIRA

## OAS

# References

|  |  |
| --- | --- |
| [LOGG] | <http://www.dotnetlogging.com/comparison/>  Note that it is written by a company that sells a commercial logging framework. |
| [MVVM] | https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel |
| [PRISM] | <https://github.com/PrismLibrary/Prism> |
| [SEIF] | Scania Engineering Information Framework (SEIF)  http://tilprod.scania.com/w/scs\_0001905\_99 |
| [J1939] | J1939  \\global\DFS04\05\_General\S\SystemsArchitecture\10\_Architecture\NetworkCommunication\Standards\SAEJ1939 |
|  |  |
|  |  |
|  |  |