Design Document – Domain Model Asset

TUGraz – T2.2B – Apache 2.0 – Client side

About this asset

This asset will be implemented as a client-side component. It is concerned with supplying a domain model for a certain game. Therefore a local stored domain model is loaded. The C# asset architecture will prevent multiple asset creation; only one client-side component of this asset per game is needed.

Domain Model

A Domain Model is a structure, which consist of competences that should be learned with the respective game and dependencies between these competences. Furthermore, a domain model includes information about actions that can be performed during the game and how the competences are linked to them (i.e. which competences are covered/conveyed by performing an action). See http://css-kmi.tugraz.at:8080/compod/rest/getdomainmodel?id=isr2013 as an example for the XML representation of a Domain Model – the structure will be adopted during the development.

Asset mechanics

This asset uses XML de-serialization.

Asset interfaces

- The client side asset component will return a serializable custom-type domain model. The information of which Domain model is delivered will be stored in the asset settings. A C# representation in the asset interface could be:

  DomainModel getDomainModel()

Asset dependencies/requirements

- This Asset does not depend on any other assets.

Milestones

Milestone 1

- t1.1: Creating the first version of the design document, defining the API and creating a dummy asset with the API implemented

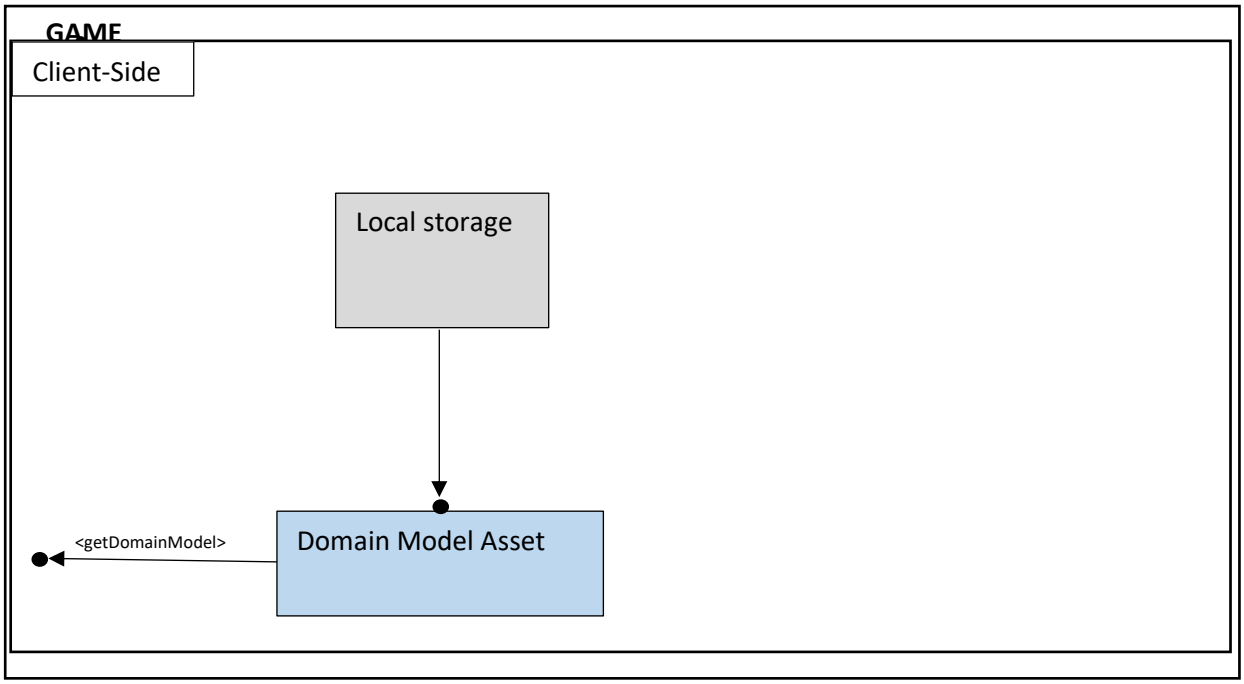
Milestone 2

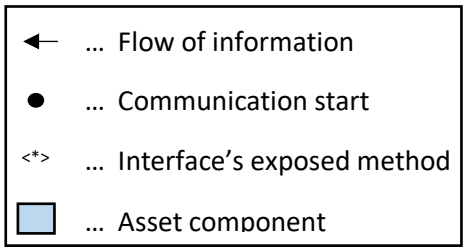
- t2.1: Create Software Asset in line with Asset-Manager infrastructure.
- t2.2: Implement Serializer/ Deserializer for the XML-structure.
- t2.3: Integrate the Asset settings.

Milestone 3

- t3.1: testing the asset with a game
- t3.2: instructions and scripts for building and deploying

- Graphical representation





Set up the Asset

The domain model Asset is used to load a domain model either from a local source or from a web server. The data source needs to be stored in the domain model asset settings.

- The settings need to be overwritten to change the data source. The C# code could look like the following, when specifying a local source with identifier 'domainmodel.xml':

```
DomainModelAssetSettings dmas = DomainModelAssetSettings.Instance;
dmas.LocalSource = true;
dmas.Source = "domainmodel.xml";
dma.Settings = dmas;
```

Use the Asset

To load the domain model a method is implemented. The data need to be supplied in a XML-file, created via a JS-HTML page (more information where to find and how to use this tool will be supplied at a later stage). To load this File, also an IDataStorage-Bridge or IWebServiceRequest needs to be implemented, dependent on the location of the data source. The following line of code can be used to load the domain model:

```
DomainModel dm = dma.getDomainModel();
```

Deployment

For the source code the following GitHub-link can be used https://github.com/RAGE-TUGraz/CompetenceBasedAssets - it contains the Visual Studio solution of the competence based asset. Furthermore, the broken links to external asset DLLs need to be fixed for each project and the Bridge code need to be adopted to the new environment, e.g. changing the IDataStorage path.

For integration into Unity, the resulting DLLs need to put into a folder in the Unity working-directory.

Unit test

For executing unit tests, the source code need to be open in visual studio and all links need to be fixed. In the test-explorer all tests can be executed.