# Design Document – Motivation Assessment Component

*TUGraz – T2.3C – Apache 2.0 – Client side*

## About this component

This component will be implemented as a client-side component. It is on the one hand concerned with supplying a motivation model for a certain player. Therefore a local stored domain model is loaded. On the other hand, it updates the motivational state of a player based on motivation model and evidences gained from the game. The component architecture will prevent multiple component creation; only one component per game is needed.

## Component mechanics

This model includs three major motivation aspects – attention, satisfaction and confidence. We assumed the game to be split up in closed game situations, in which the player solve given tasks. Within this setting, it should be possible to

- Call for help,
- Try to solve the task without success,
- Try to solve the task with success,
- Reach a new level.

In each game situation, traces are sent to the component; it calculates the following motivation quantities:

- Number of help requests,
- Number of attempts to solve the task without success (called error guess),
- Reaction time (timespan between task start and first player reaction),
- Solving time (timespan between task start and successful attempt to solve the task).

The motivation aspect satisfaction updates based on the in-game achievements, i.e. it upgrades every time the player reaches a new level. Reverse, when such an achievement is not reached in a given time period the satisfaction downgrades.

The aspects attention and confidence are evaluated together. If the player answer too fast, i.e. the reaction time is shorter than the time to understand the given task, the attention downgrades. The player is most likely guessing; all other information about the motivation quantities is ignored. An appropriate reaction time lead to an evaluation adapting both motivation aspects – attention and satisfaction. First, the solving time is compared to a maximal solving time. If it is fine, the motivation aspect attention upgrades. A too long solving time leads to a higher attention value. Second, the help requests and error guesses are compared against their upper restraints. If one of them is too high, the motivation aspect confidence downgrades. If both values are fine the confidence upgrades.

## Component interfaces

- This component will be working with motivation indicators. These indicators will form motivation evidences when being grouped (grouping is done internally). A motivation evidence is used to update the motivational state of a player. By providing a motivation hint id as enum a motivation hint can be added to the player's history. A C# representation in the component interface could be:

  ```
  void addMotivationHint(MotivationHintEnum hint)
  ```

- It will be possible to request the motivation state of a player. In C# the motivation state will be a string-double dictionary/map containing the motivation aspects and the current value in the interval (0,1). A C# representation in the component interface could be:

  Dictionary<string,double> getMotivationState()

- The motivation model, a custom-type serializable data structure containing information regarding motivation aspects, interventions and instances can be. A C# representation in the component interface could be:

  MotivationModel getMotivationModel()

## Component dependencies/requirements

- It also depends on the client-side tracker for sending the current motivaiton state to the server and performing analysis.

## Milestones

Milestone 1

- 1.1: Creating the first version of the design document, defining the API and creating a dummy component with the API implemented

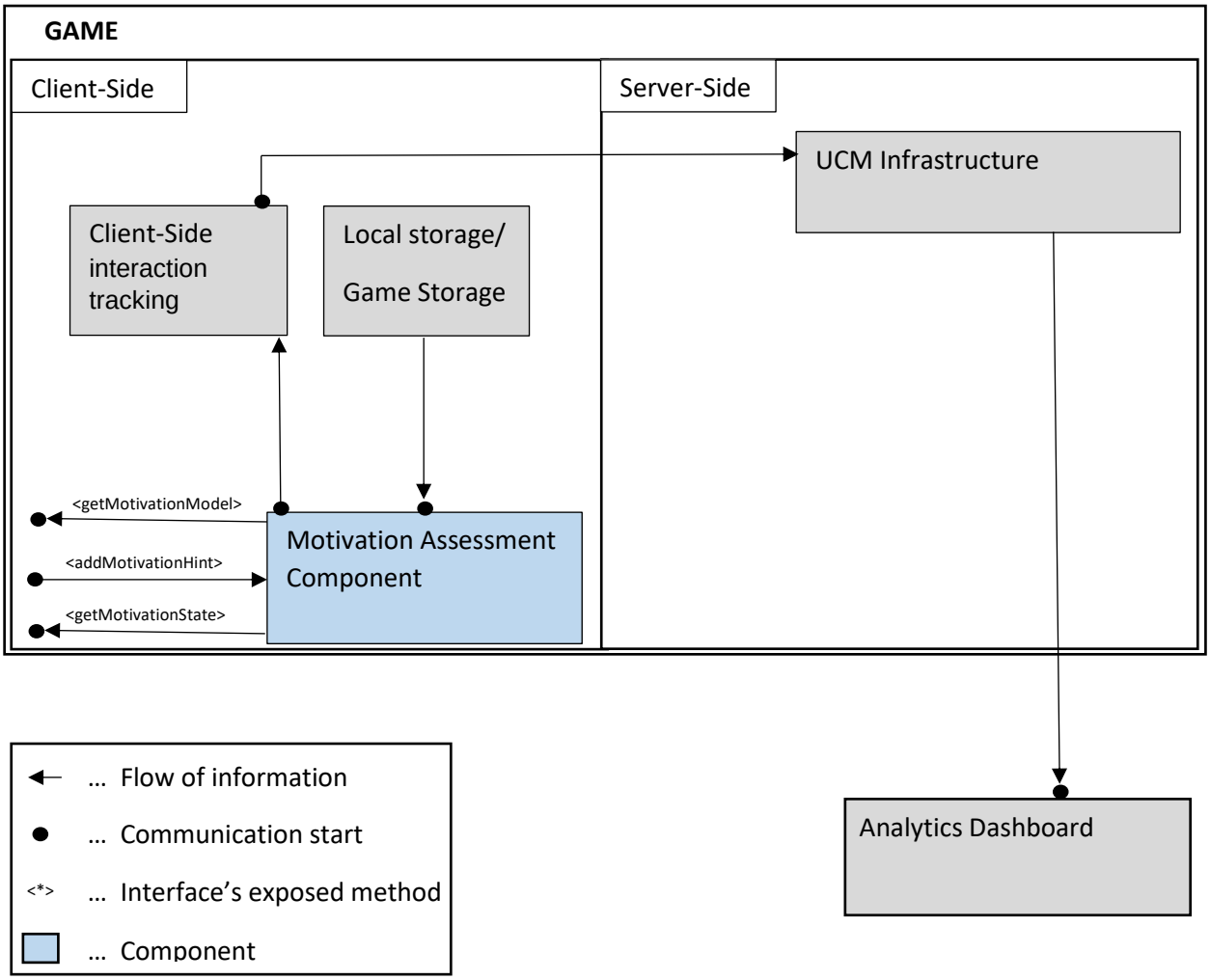
Milestone 2

- t2.1: Elaborate XML-Structure containing motivation related data.
- t2.2: Create Software component in line with Component-Manager infrastructure.
- t2.3: Elaborate Settings-structure within the Component-Manager infrastructure.
- t2.4: Integrate Tracker functionality.
- t2.5: Elaborate example data-sets (XML-structures).
- t2.6: Integrate Game Storage functionality into the component -functionality.

Milestone 3

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

## Graphical representation




## Set up the Component

The motivation assessment component is used to load the motivation model on the one hand and to update and access the current motivational state of the player. In the component settings the id of the locally stored motivation model (= data source for the motivation assessment acomponent sset) can be adopted. The C# code could look like the following, for adopting the id:

```
MotivationAssessmentAsset masa = MotivationAssessmentAsset.Instance;
MotivationAssessmentAssetSettings masas = new  MotivationAssessmentAssetSettings();
masas.XMLLoadingId = "id1";
masa.Settings = masas;
```

## Use the Component

- For loading the motivation model (which is not necessary for the regular use) the following code can be used:

```
MotivationAssessmentAsset masa = MotivationAssessmentAsset.Instance;
MotivationModel mm = masa.loadMotivationModel();
```

Note, that therefore the IDataStorage Bridge need to be 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).

- The motivational state of the player is updated by supplying hints to the component. Each hint can be one out of the following table:

| HINT - string | Meaning |
|---|---|
| success | A task was successfully. |
| new level | A new level was reached. |
| help | The help was called. |
| fail | An attempt to solve the task was unsuccessfully made. |
| new problem | A new problem was presented to the player. |

The following code snipped shows how to send a hint to the component:

```
MotivationAssessmentAsset masa = MotivationAssessmentAsset.Instance;
masa.addMotivationHint(MotivationHintEnum.Help);
```

- To access the current motivational state the following method can be used:

```
MotivationAssessmentAsset masa = MotivationAssessmentAsset.Instance;
Dictionary<string, double> motState = masa.getMotivationState();
```

The return value is interpreted as the pair of motivation aspect name and motivation aspect value (which ranges from zero to one).

## Deployment

For the source code the following GitHub-link can be used https://github.com/RAGE-TUGraz/MotivationBasedAssets - it contains the Visual Studio solution of the motivation based component. Furthermore, the broken links to external component 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.