Architecture and class diagrams

# Design by Wouter first week of July 2018

See file architecture\_v01.pdf in the same folder that this file is in.

<https://drive.google.com/open?id=1n2fdeDRjs8GjWL2ADienZ-tk2EzV4UC8>

GUI design sketches

Top GUI



Start session

Note: for start to work all involved party factory servers must have available slots to run the required agents. If you give an address of an already running agent, it is assumed to be already running.

session progress

Note: if the session crashes during run, you loose the session information and will have to start from scratch.



Tournament setup

Note: you can set nr of sessions run in parallel. All servers must have this amount of free slots, otherwise you can not start the tournament.

If server crashes during the run, the tournament re-runs the crashed session and then proceeds running the remaining sessions.

“Load config” and “save config” let the user save and re-load his settings to a local (client machine) file.



Tournament progress



# Brainstorm results spring 2018





We need separate agent and profile factories, such that for example a tournament manager can store and maintain the profiles while the agents are created by others on other computers

We thought it would be good to use JSON for client-server communication. But for passing through domains, profiles etc it seems better to use XML.

For the session settings we need something like this

Agents:

“Party1” “factory1/nastyAgent” “pfactory/partydomain/party1”

“Party2 “factory1/niceAgent” “pfactory/partydomain/party2”

“Party3” “factory3/faSolLa” “ . /party3”

Protocol: “...SAOP”

Deadline: “10 seconds”

For the tournaments we need something like we have in Genius, allowing users to generate all the session settings.

The SessionServer also will have to be listenable by web applications and thus needs to have a WebSocket. Internally it will also need to have a fast event handler for logging purposes etc.