# Comparison to other platforms

## IAGO

Java negotiation engine with web interface. Communication to engine through sockets with exchange of offers, information and emotions: SEND\_EXPRESSION, SEND\_MESSAGE(human readable text), SEND\_OFFER, and TIMING\_PROMPT. Partial offers supported. Multiparty nego is not supported. Relative importance of issues (>) and ranges of interest can be set. Seems also possible to inquiry others about relative importances. Agents are programmed in Java. They not running on their platform but somewhere else, they propose Amazon MTurk or Qualtrics. Logs apparently are sent to an email account.

References

<http://people.ict.usc.edu/~mell/publications/iago.pdf>

<https://www.youtube.com/watch?v=WFs_4YdK5UM&feature=youtu.be>

<http://people.ict.usc.edu/~mell/IAGO/tutorial.html>

<http://people.ict.usc.edu/~mell/publications/iago_4agent.pdf>

<http://people.ict.usc.edu/~mell/IAGO/installation.html>

## Diplomacy

Diplomacy is 7 player board strategic war game from 1954. Each player controls an army (or multiple armies). The goal is to conquer parts of Europe. Conquered area helps building more army. Players can form alliances. All players write down their moves, after which they are all executed simultaneously. All rules are deterministic.

The server is the Parlance game server. Parlance is programmed in python. The game server command set (daide\_syntax) is extensive and includes predicate style statements for queries (insistences, suggestions, accusations, etc) with logical operators. Atoms (‘tokens’) for all kind of objects are available (albania, ankara, black sea, ..) but also more mental objects (Insist, I don’t know, order, accept ..). Server can also send messages to client. Actions can have deadlines.

The clients are using Bandana, a java framework.

References

[http://web.tuat.ac.jp/~katfuji/ANAC2017/#diplomacy](http://web.tuat.ac.jp/%7Ekatfuji/ANAC2017/#diplomacy)

<https://www.wizards.com/avalonhill/rules/diplomacy.pdf>

<https://sourceforge.net/projects/parlance/>

<http://www.iiia.csic.es/~davedejonge/bandana/>

<https://pypi.python.org/pypi/Parlance/1.4.1>

[www.iiia.csic.es/~davedejonge/bandana/files/Bandana%20Framework%201.3.1.zip](http://www.iiia.csic.es/~davedejonge/bandana/files/Bandana%20Framework%201.3.1.zip)

<https://sites.google.com/site/diplomacyai/home/daide>

<http://www.diplomacy-archive.com/diplomacy_rules.htm>

<http://www.ellought.demon.co.uk/dipai/>

<http://www.ellought.demon.co.uk/dipai/dpp_syntax.rtf> (dead link to message syntax…)

<http://www.ellought.demon.co.uk/dipai/daide_syntax.pdf>

## PowerTac

Agents are retail brokers in a local power distribution region. They purchase power from a wholesale market as well as from local sources (homes and businesses with solar panels), and sell power to local customers and into the wholesale market. The winner is the broker with the highest bank balance at the end of a simulation run.

The agents are written in java, all message based (ApacheMQ).

The server code seems available from github. I understand the server simulates the “clients” that subscribe, negotiate a tariff, and take or deliver power.

## References

<http://www.powertac.org/>

<http://www.powertac.org/get-involved>

<https://github.com/powertac/sample-broker/archive/v1.0.0.zip>

<https://poseidon01.ssrn.com/delivery.php?ID=641117074119116066083028091085085024016073027027075062101005095022079120102112125011096100063045053098009088086109105027109018059021009023036091092100026085094079035040080105064117002102030001118014082026127071088100066089121031075110097126064071031&EXT=pdf>

<https://github.com/powertac/powertac-server/wiki>