Service Oriented Architecture (SOA) Demonstrated as a REST Web Services Game System - Term Project Proposal

Ron Harwood
CAS 703 – Winter 2015
harwood@mcmaster.ca

1. PROJECT PROPOSAL SUMMARY

The assigned architectural style, Service Oriented Architecture (SOA), will be demonstrated through a game playing system implemented as a web service infrastructure. The web services will use a REST (representational state transfer) [1] style of communication. This will allow for human readable Uniform Resource Locators (URLs) for the services and greatly simplify the documentation process. In turn, this should make the development of client software much easier.

2. SYSTEM COMPONENTS

The project will be developed as several distinct service components (see Figure 1).

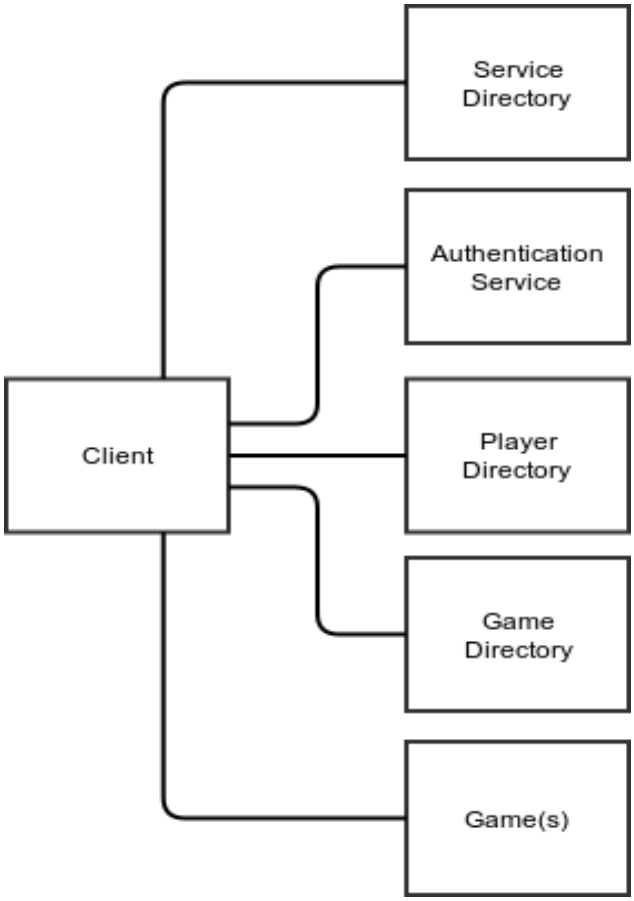

Figure 1

2.1 Top Level Service Directory

The service directory will be a central source and starting point for discovery of the other services available and how to access them.

2.2 Player Directory

The player directory will be an interface to the player database. Other services in the system would access the player directory for pertinent information - such as player name, contact information or to identify "friend" relationships between players.

2.3 Authentication Service

Separate from the player directory, the authentication service will allow the system to positively identify system users, respond to password reset requests or even authenticate against external services (Google, Facebook or Twitter).

2.4 Game Directory

The game directory will list the available categories of games in the system, their descriptions and allow players to start, join or view individual games in the system.

2.5 Individual Game Services

Each category of game will have its own service to allow system users to play the games. The game directory may also query a game service for the current status (open for new players, in progress or completed), lists of players for each game and create new instances of games as players request them.

Games for the project will be simple - for example tic tac toe or checkers - but in theory, any turn-based game (such as a board or card game) could be adapted to the system.

2.6 Client Software

The client could be built as a web, desktop or mobile application. The very loosely coupled nature of the services would allow even a simple command line client and one could even enter some commands via the URL bar of a web browser.

For purposes of the project, a simple web client will be developed.

3. PROJECT JUSTIFICATION

The proposed project will serve as a demonstration of SOA for the following reasons:

- Each service will be self-contained and limited to its area of responsibility. Versioning of the service's Application Programming Interface (API) will allow legacy applications to use the service while allowing new applications to implement improved features of the service - allowing for extensive re-use without breaking backwards compatibility.
- While other services may query or use the service, they will not be dependent on each other in order to operate.

A failed service will cause a querying service to report back and handle the failure appropriately.

- The service directory will serve as the central point of publishing of services for the system. This central point could then be, in turn, published to external directories of services.
- The use of a REST style web service protocol allows for a common and standardised interface between the system components and clients. Using Hypertext Transfer Protocol (HTTP) [2] request methods (GET, POST, etc.) allows the re-use of a single URL for a system resource for different requests and commands. Additional features of HTTP (eg. cookies, response codes, caching) allow further flexibility for the protocol while still maintaining standards.
- The individual game services and the game directory can each be seen as a level of aggregation of other services within the system - accessing player information and authentication services, for example.

4. TECHNOLOGY

The services will be developed using a Linux, Apache, MySQL and PHP (LAMP) software stack. Simple client software will be developed for common web browsers using HTML and possibly Javascript.

5. REFERENCES

[1] Fielding, R. T.; Taylor, R. N. (2000). "Principled design of the modern Web architecture".

https://www.ics.uci.edu/~fielding/pubs/webarch_icse2000.pdf

[2] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee (June 1999). Hypertext Transfer Protocol -- HTTP/1.1. IETF. RFC 2616.

https://tools.ietf.org/html/rfc2616