Task 4: Group Reflections

Assignment 10 & 11, Group 3

December 16, 2022

1 Hard-Coded Information

Task: For each of the tasks, what is the information you had to hard-code into your executors?

The library WoT-TD-Java offers several ways to retrieve an affordance from the W3C WoT Task Description (TD). Here, we can retrieve an action affordance based on semantic type, action by name, and retrieve a list of all available actions.

At first we have implemented the retrieval based on name. After second thought, as names can easily change we have later decided to opt for retrieval based on the semantic type. For instance the URL to register operator was defined as follows:

- "https://interactions.ics.unisg.ch/cherrybot#RegisterOperator";

Here we are not dependant on a specific name, and thus further decouple the retrieval of the TD. However, we still had to hard code the base URI in this example. Similarly to lookup affordances for our executorRobotSearchEngineService we had to hard code it as well

- "https://interactions.ics.unisg.ch/cherrybot#Cherrybot¿";

One problem exists that if the name of the robot changes we cannot use our service. Semantically we would achieve more decoupling if we could specify the type of the robot (e.g. arm robot) At last, unsurprisingly we have hard coded the SPARQL query for our search engine service and the positions related to the movements of the robot.

2 Hypermedia APIs

Task: What are the advantages and disadvantages you see for using Hypermedia APIs?

2.1 Advantages

One big advantage of Hypermedia APIs is that the API consumer does not need to be aware of all possible Endpoints, wihch allows for changes at run time when URIs are resolved. This enables navigation of the API over hyperlinks provided (e.g. using HATEOAS constraint) Subsequently, having the freedom to follow hyperlinks consequently makes our REST API explorable by the API consumer (machine or human).

2.2 Disadvantages

Even though the previously discussed decoupling comes with many advantages, we also need to be aware of the shortcomings of hypermedia APIs. First, it must be stated, that if the API consumer does not take advantage of semantic descriptors and hard codes the URIs all advantages from decoupling are lost. Altough this hardly be prevented, one could opt for shallow ontologies instead of deep ontologise to make it easier (broad but less expressive). Similarly, enriching the semantics and relations is beneficial, it requires to API consumer to comprehend the semantics. This further makes building both server and client harder.

2.3 Final Remarks

In conclusion, even though hypermedia create more effort to both consume and build, there are many advantages creating several opportunities for applications and (web) architectures. Additionally, one could argue that this burden might be offset in certain applications, as it might be easier to maintain.

3 Relationship IDLs and Hypermedia APIs

Task: How do Interface Definition Languages (IDLs) and Hypermedia APIs relate to one another? You should justify your answer using concrete examples of IDLs and Hypermedia APIs.

3.1 Relationship

IDLs and Hypermedia APIs are related in that they both describe the interface of a software system. With traditional APIs, the interface is fixed and cannot be changed without breaking existing clients. However, with Hypermedia APIs, the interface can be more flexible and adaptable, as new actions and resources can be added simply by adding new links to the responses. This can allow the API to evolve and grow over time without breaking existing clients. By providing a clear and well-defined interface, combining IDLs and Hypermedia APIs enable different components of a system to communicate and interact with each other in a flexible manner (i.e. decoupling of semantics and enabling adaptability at run-time).

3.2 Concrete Examples

A concrete example of an IDL that is driven by hypermedia is the Web Application Description Language (WADL), which is used to describe RESTful web services. WADL specifies the methods and data types that are used to interact with the web service, as well as the links that the client can follow to access the various resources and actions provided by the service.
CORBA is another examploratory IDL discussed in the lecture, that is used to define the interface of a distributed system. One possible use case for driving CORBA with hypermedia is to enable the client to discover and navigate the available actions and resources within.

Altogether, by leveraging hypermedia, IDLs can provide a more powerful and flexible way to describe the interface of a software component or system.