Part I

Conclusion on handin 3

1 How our program works

Our handin is structured in two seperate projects (developed in eclipse) TaskManagerServices and TaskManagerClient

Our Services project is built in eclipse as a dynamic web project. This is done to easily get the soap functionality and the servlet interface. It can communicate with the task manager services through a SOAP and a REST client.

The REST client is located in dk.itu.smds.e2012.webservices.basic.client.TrustcareRestClient. This client uses the generic REST client located in the common package and simply performs http calls against the target server. In some cases it (de)serializes response and request as appropriate. This is done through classes found in the xml package.

Our SOAP client for the trustcare service is automatically generated from a wsdl file from the server. It's implemneted as a proxy, with its interfaces located in the $dk.itu.smds_e2012.lab.week_04$ package The service can then be used as a ITaskManagerService object which really is a ITaskManagerServiceProxy.

Our services are exposed both through REST and SOAP. The REST interface is done through the jerset library, with the ressources located in the resources package. We've made two endpoints to meet the requirements of the assignment. For instance our http client library (also jersey) didn't allow for sending entities in delete request. So we've made a resource endpoint which takes a url parameter.

Our SOAP service is exposed as defined in the class basic.services.SoapService. This class simply defines WebMethods which can be exposed as a soap interface. It then allows a client to be generated by an IDE.

2 How to test our web service

We've included the TaskManagerClient project which calls both our REST and SOAP interfaces. The project consists of some dependencies (which are simply duplicated packages, this should be made into Jars or perhaps released more gracefully as an ANT project or similar) and some pretty self explanetory test classes. We've run all the classes on the glassfish application server, but any server should do

3 Notable differences between REST and SOAP

There's clear technical differences between REST and SOAP. Our experience working with it mostly lives up to our initial thoughts about the two technologies. SOAP is made for machines, and as such the machine should spare us from a lot of work. The downside is among others the added size of data back and forth. An important thing to note about SOAP is that it's strenghts are also it weaknesses (at least in our experience) We were hugely dependent on the tools of our IDE and this takes away some flexibility in the team (everybody has to run eclipse) Another thing is that the tools doesn't always work as advertised (we spent literally hours trying to make eclipse do what we wanted) The community is also a big factor. It seemed that the defacto standard for services is the apache axis framework which hasn't received an update since 2006

Rest on the other hand should require more manual work, but is generally a more "lean" and independent technology. Our experiences seems to confirm this. More manual work is done, but we have much more control of what is actually going on. And the tools play a smaller role (and there are (increasingly) more to choose from) The work is mostly done for us with just the small rest client we were giving in this project.

All in all REST is both the most pleasent and modern technology to work with