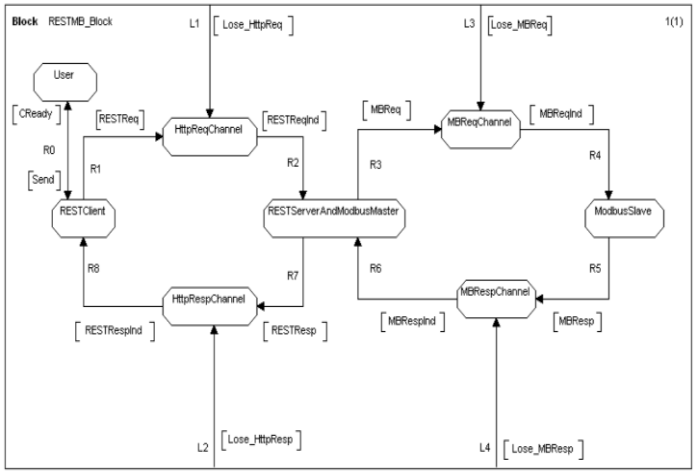

Figure 4: RESTMB block specification

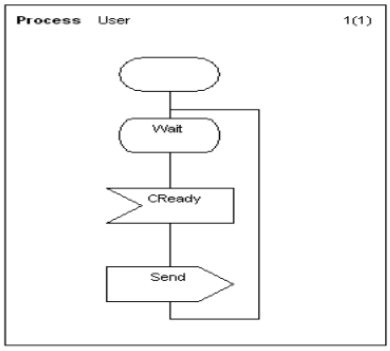

Figure 5: User Process specification

Figure 4 and 5 shows the User process and RestClient process respectively.

1.User initiates a rest request through the RestClient once the RestClient is ready.

2. The RestClient after sending the request starts a timer and waits for a specified time for the response.

3. if response is not received with in specified time, timeout happens and an error message is as indicated in the process diagram.

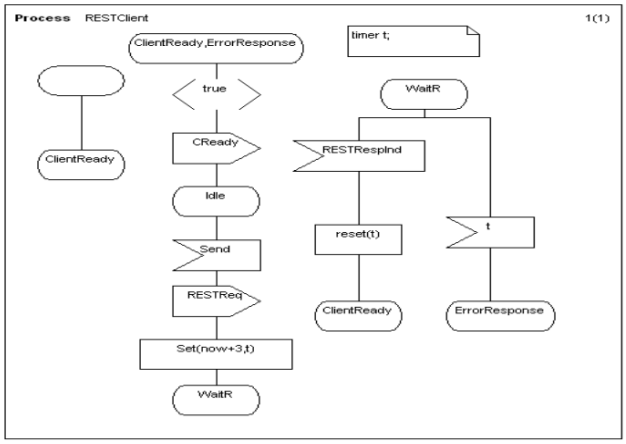

Figure 6 : RestClient process specification

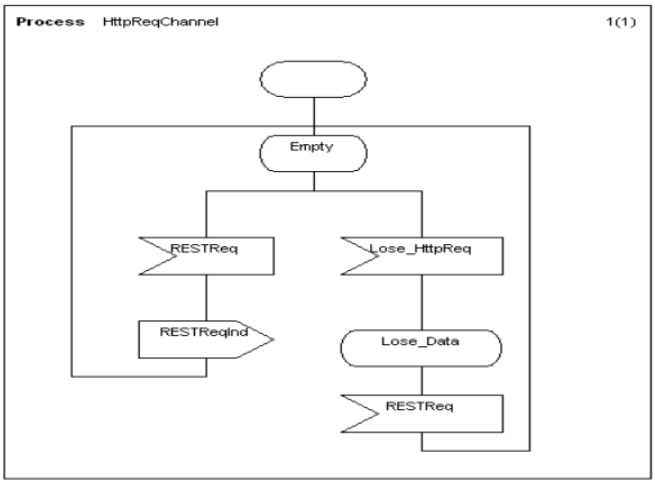

Figure 7: RestClient Process

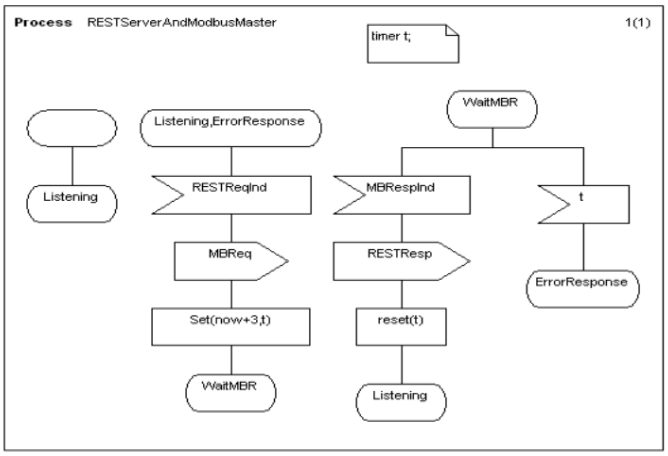

Figure 8: RESTServerAndModbusMaster Process

Figure 7 and 12 shows http request and response channels respectively which acts as channel for sending and receiving REST request and response.

Figure 8 shows the RESTServerAndModbusMaster. It processes and converts http (ReST) request from RestClient to modbus request and modbus response from ModbusSlave to http(ReST) response.

Figure 9 and 11 shows modbus request and response channels respectively which acts as channel for sending and receiving modbus request and response.

Figure 10 shows Modbus Slave process which waits and listens for modbus request from the Modbus master (RESTServerAndModbusMaster). On receiving a request, it sends back a modbus response.

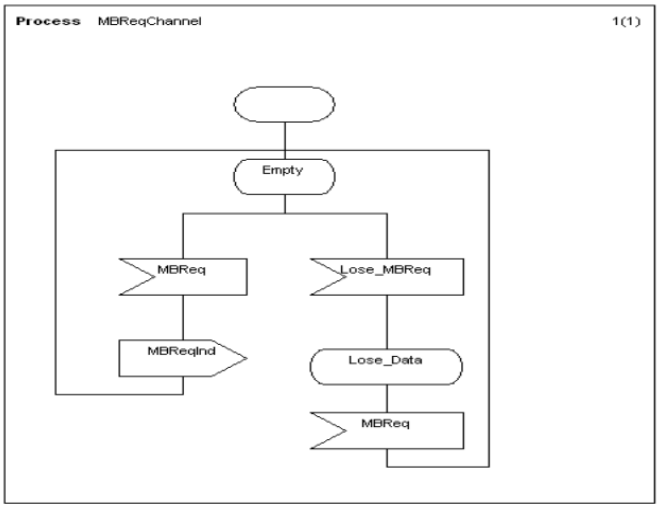

Figure 9:MBReqChannel Process

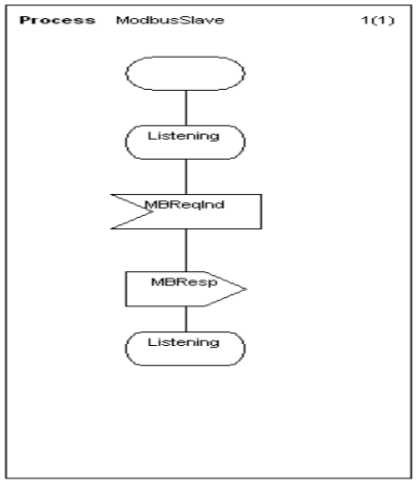

Figure 10:ModbusSlave Process

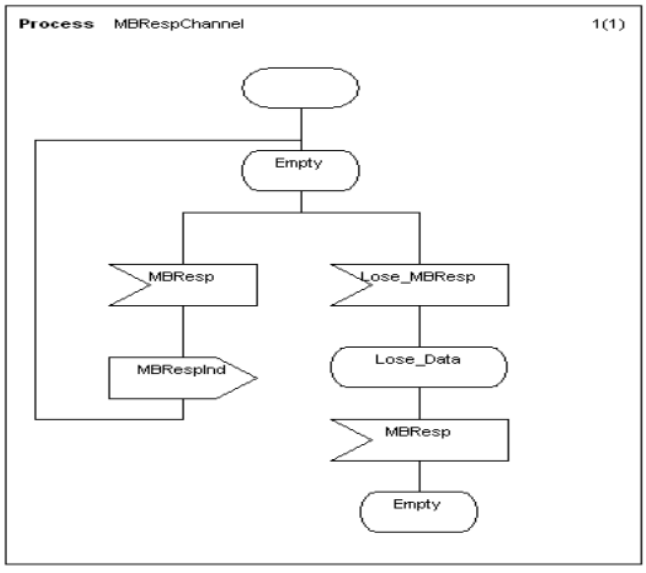

Figure 11:MBrespChannel Process

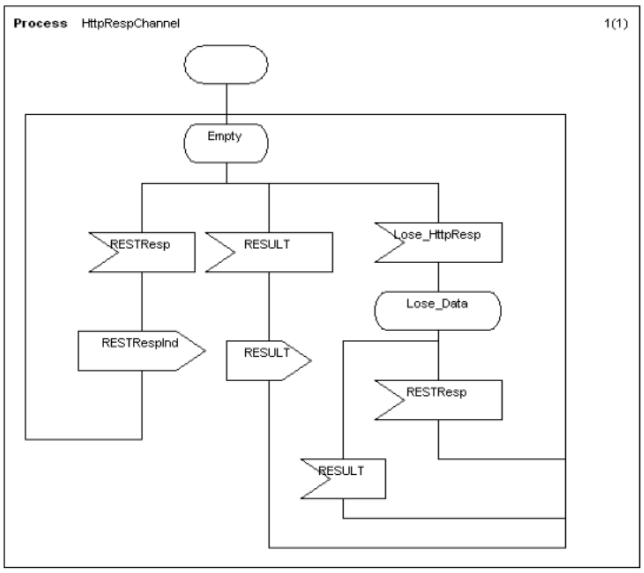

Figure 12:httpRespChannel Process

5.Verification and Validation

The design of the framework must ensure its ability to operate under increasing load, increasing complexity of requests and increasing size of resulting composite services. Hence verification and validation is done to check for correctness of protocol specification and check the protocol for liveness property and safety properties. Protocol validation can be used to increase the quality of protocol design by verifying the system against the requirement.

Validation ensures that the protocol specifications will not get into protocol design errors. (Deadlock, unspecified reception etc).
The message sequence chart in figure 13, shows interaction between entities and complete service discovery process.

Safety Property:
1. Ensures non-violation of assertions.
2. System ensures that proper data is send from "RestClient" and "RESTServerAndModbusMaster", although it may get lost in the channel

Liveness Property:
1. Ensures proper termination of protocol.
2. Even if the data send from RestClient" / "RESTServerAndModbusMaster" gets lost in the channel,it is ensured that the protocol terminates correctly.

Validation
1. From the message chart diagram, it is seen that the proper response is obtained for valid requests.

2. It is also ensured that incase of lossy channel/data loss,the system does not go to a deadlock.
3. The timers in the “RestClient” and “RESTServerAndModbusMaster” ensures that if the response is not obtained within a timelimit,an error response is generated and the system again is ready to accept new requests.

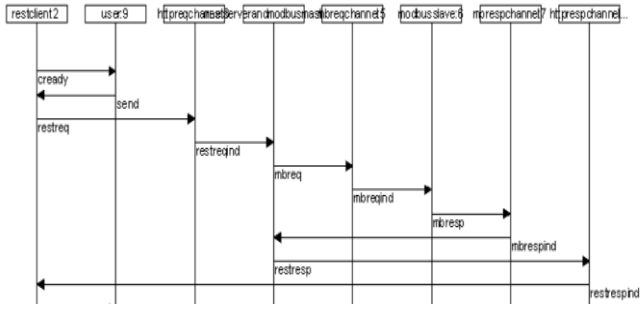

Figure 13 : MSC chart generated using SDL

The Message Sequence chart generated for the system for normal flow is shown in figure 13.

1. The RestClient sends a http request “restreq” through http request channel once the user initiates it.
2. The http request is converted to modbus request “mbreq” by the server and send to the modbus slave through modbus channel.
3. On receiving a modbus request, the modbus slave sends back the modbus response “mbresp” to the server.
4. The server processes the modbus response and generates the http response “restresp” which is send back to the rest client through http channel

5.Implementation

Following tools were used for implementation of the system

5.1 Slave Simulator

A slave simulator is used for simulating a device which has modbus capability.Basically it listens for modbus requests and responds based on the modbus command received. “Diagslave”, a freely available slave simulator for modbus was used for the project, is shown in figure 14.

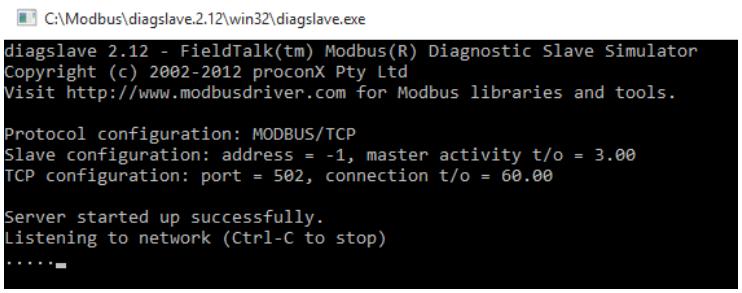

Figure14: Diagslave server

5.2 RestServer

This is required for performing REST operations, and to obtain the appropriate “REST” response when a query was given by user.
“EVE”, a freely available python library was used to perform necessary REST related operations and to generate appropriate REST response.Figure15 shows documentation of Eve framework.

Figure 15 – Eve python ReST API framework

5.3 Rest Client

A Rest Client is required for the user to send “REST” query to rest server. “POSTMAN” extension of chrome was used in the project to act as RestClient.Figure16 shows UI interface of POSTMAN tool.

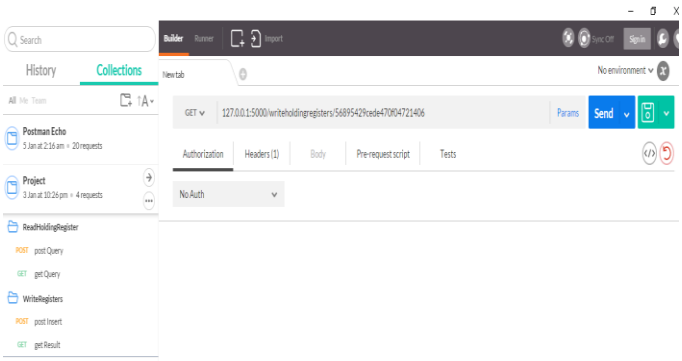

Figure 16: Postman extension for chrome

5.4 Modbus Master

A modbus master is required to send modbus command to the slave simulator and obtain the modbus response. “Pymodbus” a freely available python library for modbus was used in the project.Figure 17 shows Pymodbus documentation.

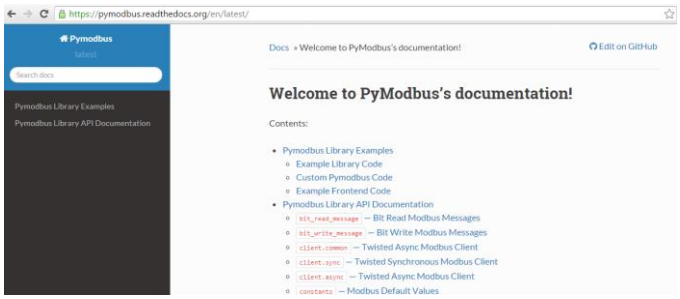

Figure 17:Pymodbus module

1.Using “Eve” Rest framework,the rest services for the Modbus function codes “Read holding registers” and “Write holding registers” were implemented.This can be extended to other function codes as well.

2.User can send modbus requests(function codes) by embedding the required data in the request made to the server.On receiving the request,the eve server processes the request and using the “pymodbus” makes a modbus request to the slave simulator and receives back the modbus response.

3.The Modbus response thus obtained can be send back to the user by embedding it in the response data as a response to a rest request.

6.Conclusion

Formal SDL (Specification and Description Language) to specify framework and study their performance evaluation was used. Any web/mobile application which requires interaction with devices can directly call the REST API for modbus and communicate with device.

References

1. http://python-eve.org
2. https://pymodbus.readthedocs.io/en/latest/
3. http://www.modbusdriver.com/diagslave.html
4. https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop
5. https://en.wikipedia.org/wiki/Representational_state_transfer
6. http://www.ibm.com/developerworks/library/ws-restful/
7. https://en.wikipedia.org/wiki/Modbus