Open Space Box Model: Service Oriented Architecture Framework for Small Spacecraft Collaboration and Control

Atif F Mohammad
Jeremy Straub

Available at: https://works.bepress.com/jeremy_straub/66/
Open Space Box Model: Service Oriented Architecture Framework for Small Spacecraft

Collaboration and Control
Atif Farid Mohammad, Jeremy Straub
Department of Computer Science, University of North Dakota

Abstract

A Cubesat is a small satellite with very less competence to compute, it requires software engineering techniques, which can enhance the computational power for this small box. A model-driven approach of software engineering, which is called OSBM or Open Space Box Modeling technique, is an excellent solution to this resource maximization challenge. OSBM facilitates apparition of the key solution processes computation and satellite related data elements using Service Oriented Architecture 3.0 (SOA 3.0) as base to work on to design services. The key challenges that can be handled by utilizing OSBM include concurrent operation and tasking of few as five computer-on-module (COM) units and a flight computer and the associated data marshaling between local and general storage installed within the satellite.

One of the most important challenges with this highly-distributed architecture is to make sure that tasks are convincingly consistently distributed between different elements of the computing units, such that a single unit is not acting as a bottleneck for other onboard computers’ tasks. OSBM is a work distribution model and was created to characterize the system needs and facilitate high-level task-type assignment among the computing systems. A communications model was also created to characterize the level of data throughput required and based on several communications approaches to the ground station software. The proportional benefits and drawbacks of the OSBM approach are discussed. The design process utilized is also presented.

Introduction

OPEN-SBM introduced in this paper is a new paradigm for the small and nano-satellites software systems, which uses SRD or System Requirements Design methodology, which was designed at MIT [1]. An introduction to SRD is given in Table 1.

Challenges

One of the principal challenges with this highly-distributed architecture is to ensure that tasks are reasonably evenly distributed between different elements of the computing units. This is to ensure that a single unit is not acting as a bottleneck for the other onboard computers’ tasks. A work distribution model has been shown earlier in the form of a service block, and is created to characterize the system needs, as well as facilitate high-level task-type assignments between the computing systems.

This model is designed using OPEN-SBM techniques using SOA 3.2. A communications model is also created to characterize the level of data throughput required (based on user needs) and possible (based on several communications approaches) to the ground. This is to allow the selection of the most optimal communications approach between satellite sub-systems as well as between the satellite and ground station, thus, allowing maximization of each communications procedure. This paper has presented the model-based software engineering approach utilized for the OpenOrbiter mission. The model has been compared with a Finite State Machine approach with a functional example.

Validity

The details discussed in this section provide a set of finite states machine with a set of states using a deterministic automation to show the validity of OPEN-SBM. A finite state machine contains a set of states, with a starting state. Each state has a set of transitions to other states, based on the next input symbol, in case the sub-systems as mentioned earlier for the satellite have a sequence of requests to process some functionalities. This can be considered in the shape of a symbol of a finite state machine.

A transition between states has a condition and an action, and this action can usually be considered as requests from one sub-system to another. If the condition is met, the FSM performs the action and then enters the new state in the satellite sub-system to another sub-system of the satellite. This cycle repeats itself until reaching the end state, or until it produces the requested result from ground station.

An FSM functions is stated as follows: It always begins operation in the starting state (q0). When provided a string of symbols over the input alphabet (Σ), it processes the string one symbol at a time, or in the satellite sub-system case, one request at a time, from left to right. As the input string is processed, the FSM changes states based on the current state and the current input symbol/request being processed (δ).

For instance, if the FSM or sub-systems within the satellite is in state q and the symbol being processed is a, then the next state of the FSM is determined by the value of the transition function at q and a: δ(q, a) = r. After processing the entire string, the FSM is found in some state s. If s is a member of the accepting states (F), then the input string w is said to be accepted by the FSM, otherwise the string is considered rejected.

A functional Finite State Machine example is give below, where 0 and 1 are some transaction requests among 5 states to show the validity of OPEN-SBM.

The use of basic Finite State Machine is given to prove that OPEN-SBM is based on computational theory, which is the founding stone of computer science and hence OPEN-SBM can be utilized to design and develop models for nano-satellite as an open source way.

Conclusion

This model is designed using OPEN-SBM techniques using SOA 3.0 and a communications model is also created to characterize the level of data throughput required (based on user needs) and possible (based on several communications approaches) to the ground. This is to allow the selection of the most optimal communications approaches between satellite sub-systems as well as satellite to ground station, thus, allowed maximization of each communications opportunity. This paper has presented the model-based software engineering approach utilized for the OpenOrbiter mission. The model has been compared with a Finite State Machine this approach with a functional example.

References


Acknowledgements

This paper presents work that will be presented in greater detail in A. Mohammad, J. Straub, C. Korvald. Model-Based Software Engineering for an Imaging CubeSat and its Extrapolation to Other Missions. IEEE Aerospace Conference. 2013.