On-board autonomy of lander units for comet nucleus exploration

ATTILA BAKSA
KFKI RMKI
baksa@rmki.kfki.hu

Keywords: space research, spacecraft, lander, computer, software, autonomy, high reliability

Keeping lander units functional in the hostile, energy-lacking environment in the outskirts of our Solar system is a great challenge. An autonomous real-time control system of a lander is expected to response on board to both nominal and non-nominal events without any external intervention. Recent developments in microelectronics make it possible to use such space-qualified microprocessors that allow the development of highly autonomous on-board software systems. But the increased computing power itself is not all - equally advanced software methods are also needed to provide real autonomy. Considering the complexity of a number of mutually interacting tasks, it is necessary to model them by well-described abstract logical modules. Our focus was on managing the static and dynamic behaviour of the system separately and eventually we developed the Mission Sequencing Object Model Language for describing the long-term autonomous mission control mechanism. This model was implemented on the Philae Lander for the Rosetta mission of the European Space Agency, which was successfully launched on 2 March 2004.

Many space missions are nowadays under preparation, which use the most advanced space technology to explore the unknown depths of our Solar system. The most recently developed microelectronic devices, such as low power consumption high-speed microprocessors, FPGAs, high efficiency solar cell modules and high storage density batteries open the way to keep functional even in hostile, energy lacking environment in the outskirts of our Solar system.

Problems

Operating so far from the Earth poses not only the problem of the rocket engines capable to get there but also causes a long dead time in the remote controlled operations issued from the Earth. While in the distance of Mars the radio systems signal propagation delay is just 20 minutes then, for example in the distance of Jupiter the control loop delay can take several hours. In the cold of the outer solar system there comes an additional problem: it is necessary to heat all the electrical equipment to keep them functional, but we have very low energy budget. We can, however, use radioactive energy source, which is not recommended for ecological reasons.

Fig.1. Rosetta lander (Philae)

It can easily happen that we have to acquire solar energy for days but this energy will be enough just for a few ours of scientific activity. It is clear that a spacecraft far from the sun shouldn't waste its energy and time by waiting for control signals from the Mission Control Centre on the Earth. Direct control is potentially dangerous because the energy balance of the whole spacecraft may collapse in case of an unexpected problem, due to a time-consuming command exchange.

The solution

The only solution for these situations is to increase the rate of on-board autonomy. We have to rely on a built-in intelligent, adaptive control system, which provides the following functions:

- Managing the scientific operations continuously without any interactions with the Control Centre
- Adaptation to the non-predictable timing requirements during the scientific operations
- Giving real-time autonomous reaction to the nominal and non-nominal external events
- Handling emergency situations
- Taking care of the energy balance
- Capability to store the measured scientific data, even in case of energy loss

Having a control system without these capabilities can easily lead to a failed mission. The earlier surface modules had insufficient computing power for long term autonomous missions. In the past in most cases it was unfeasible to plan missions not requiring the Earth intervention for more than a few days. Recent developments in the field of microelectronics make it possible to use such space-qualified microprocessors, which allows creation of on-board software systems having high rate of autonomy. Although these processors still do not provide enough computing power to employ real Artificial Intelligence but a good design model makes it possible to create a real-time autonomous system which can handle all the tasks of a lander unit even in long-term missions.

Considerations

The extent of fault-tolerance of the software system is especially important when forming autonomous strategy of the on-board software. In the design of the software model we have to take the following policies:

- For the sake of the safe operation the measured environment values must be verified according to limits of values, trends, etc. It is equally important to verify the software variables before they are used as actuator signals.
- The internal control model has to be sensible also for the anomalies of its environment.
- There has to be state transitions for every events. This condition was hard to achieve In the case of high number of potential events or bad predictability of the complete event set in the traditional models. What is innovative about our conceptual model is that it provides solution to reconfigure the state transition definitions even during live operation.
- We have to apply timeouts in management of every states.
- In order to keep the reaction time low we have to minimise the execution time in critical or non-interruptible states.
- To minimise the danger of crash or malfunction of the system the received telecommands have to be fully decoded, checked and verified.
- The model can not have logical path causing system deadlock

The task

We've studied in details the requirements for a probe or rover, which should operate on the surface of a planet, asteroid or a comet. We divided all the requirements into the following topics:

- Controlling the Lander unit during approach, descent, landing and surface operations
- Keeping the Power and Thermal balance of the lander unit
- Management and execution of the scientific program
- Collecting and storing the experimental data acquired by the payload and service subsystems
- Management of telecommunication functions (Radio link management, telecommand reception and telemetry transmission)
- Providing fault tolerance by handling the built-in hardware redundancy

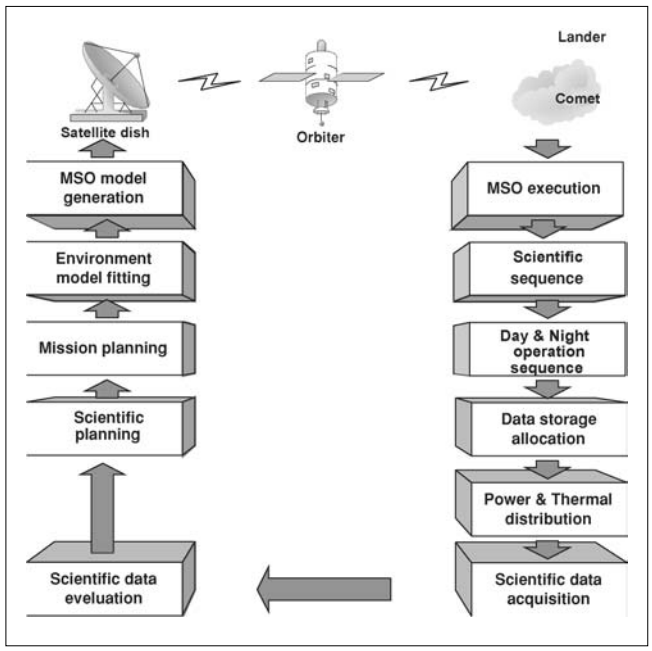
Examining each aforementioned items we came to the conclusion that all of them interact to the others. So an appropriate central logic should provide interaction between them. In implementation, however, it can lead to a very complex control algorithm that is very hard to implement software. To fulfil all the mentioned requirements in a manageable way it is necessary to construct an abstract architectural model which is rather flexible, but as simple as possible.

The model in general

Our central idea is to separate the static and the dynamic behaviour of the system. This modelling approach has many advantages opposite to the concept of the software design of earlier surface modules. This method minimises the required data traffic between the lander and the Mission Control Centre because the various combination of the static and dynamic algorithm reduces the required telecommands for the control. It is an essential point because the upload speed of the telecommands through the communication link is 10-200 bits/second at most from the Earth and the communication session length is usually limited to 10-20 minutes because of the high signal-to-noise ratio. We continuously kept in mind to provide the possibility to easily reconfigure the whole central logic during any mission phase in the lifetime of a lander. So we broke down both our models into a set of individual basic parts. A single part is called Mission Sequencing Object (MSO). The size of an MSO is varying to fit into the telecommand packet. The link between the MSO items make it possible to design them independently which gives us an understandable, easy-to-use man-machine interface for the Mission Control Team. Using this modelling language makes it possible to translate the Mission Control Information to a data format, which is uplinkable to the on-board Central Control Computer via Telecommands.

It is also possible to attach MSOs to the Mission Control Information Database of the Knowledge Management System. Additionally we designed an advanced storage and retrieval algorithm for the on-board control software for storing MSOs to and retrieving them from the on-board memory. This algorithm has dependable but space saving data storage capability and a very short seek-time for accessing further MSO items.

Fig.2. Mission sequencing objects generation and usage on Rosetta lander (Philae)



The static model

The main task of the static model is to generate the actual operational state of the system. The basic MSO of the static model is the SMSO (Static Mission Sequencing Object). An SMSO is responsible for the following system attributes:

- Parameters of the current operation mode
  - Operation speed
  - Rate of failure tolerance
  - Rate of energy saving
- Current set-up of the scientific payload instruments
- Parameters of the data collection for scientific instruments
- Data transfer quotas for the optimised distribution of data storage capability
- Protection for the critical operation phases
- Set up priority order for the currently operating experiments from the following point of views:
  - Energy distribution
  - Data collection and storage
  - Service speed

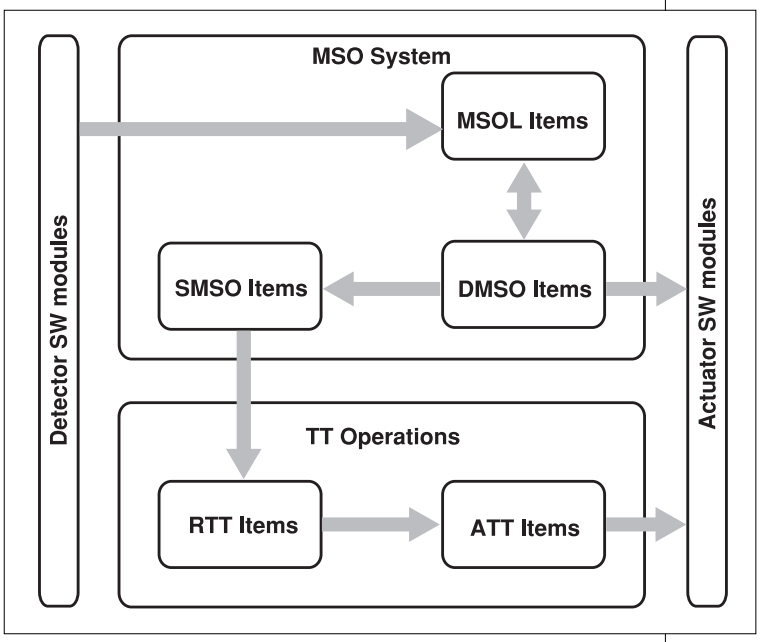


Fig. 3. Structure of the Mission Sequencing Object Model

The dynamic model

The dynamic model describes the required reactions to nominal and non-nominal events and defines the state changes in the static model. The basic MSO of the dynamic model is the DMSO (Dynamic Mission Sequencing Object). A DMSO is responsible for the following system attributes:

- Reference to the current SMSO item
- Nominal and non-nominal events definition
- Reactions to nominal and non-nominal events, which are categorised as follows
  - control-, failure prevention-, failure handler-, recovery- and safe mode-algorithm
- Time tagging and time-out mechanisms
- Link and connection definitions to other DMSO items. Possible connection types are the following:
  - chain like, call like and jump like

The implementation

This model is implemented for the ESA (European Space Agency) cometary mission called Rosetta. The on-board central computer (Command and Data Management Subsystem) of the Philae Lander in the Rosetta mission is equipped with this technique.

The scientific mission of the Lander has not defined yet because there are a lot of uncertainties concerning the attributes of the target object. Using this controlling technique made that possible to finish the on-board software development without detailed information about the scientific program of the mission. The final scientific program will be translated into MSO items and will be uplinked to the Lander via telecommands, just before starting the descent to the comet surface in 2014. Rosetta was successfully launched on 2 March 2004 and its is now on its decade long way to comet 67P/Churyumov-Gerasimenko.

We hope our model will help Philae to accomplish its landing and operating on an ice mountain bouncing around its three axis. In case of success this event may open a new chapter in the history of the Solar system exploration.

The software environment

The on-board software of the central computer of the Philae Lander consists of a real-time operating system and 8 application tasks. All these software modules are specially developed by our team for the Harris RTX2010RH microprocessor. The co-ordination of the scientific program and the overall control of the algorithms used by the application tasks are done by the MSO modelling language.

References

[1] Ron S. Kenett, Emanuel R. Baker:
Software Process Quality ,
1999 New-York

[2] Savio Chau, Abhijit Sengupta,
Tuan Tran, Ali Backhshi:
Ultra Long-life Spacecraft for
Long Duration Space Exploration Missions
Space Technology Vol. 23, 2003

[3] David P. Youll:
Making Software Development Visible,
1990, Chichester