A generic architectural framework for the closed-loop control of a system

Gérard Verfaillie and Michel Lemaître
ONERA
2 av. Édouard Belin, BP 74025, F-31055 Toulouse Cédex 4
Gerard.Verfaillie@onera.fr, Michel.Lemaitre@onera.fr

Marie-Claire Charmeau
CNES
18 av. Édouard Belin, F-31401 Toulouse Cédex 9
Marie-Claire.Charmeau@cnes.fr

Abstract

In this article, we present a generic framework for the functional architecture of the closed-loop control of an engine or a system. Besides its genericity, its main features are (1) a decomposition of the system control into hierarchically organized modules, (2) an encapsulation of control and data inside each module, (3) standardized communications between modules via requests, reports about request execution, and information about the system state, (4) a standardized organization of each module around the following four components: tracking of the received requests, tracking of the emitted requests, tracking of the system state, and decision-making upon request emission, and (5) a common framework for the interaction between reactive and deliberative tasks inside the module components and especially inside the state tracking and decision-making ones.

We show how this framework can be applied to the control of an autonomous satellite dedicated to Earth watching and observation.

1 The AGATA project

The architectural framework presented in this paper is one of the first results of the AGATA project (Autonomy Generic Architecture: Tests and Applications, http://agata.cnes.fr). From mid-2004, this project brings engineers

and researchers from CNES[1], ONERA[2], and LAAS-CNRS[3] together around the global objective of increasing spacecraft autonomy [CB05].

Most of the satellites and space probes that are today in operation are permanently and tightly controlled by human operators in ground control centers. Except for specific tasks, such as thermal, energy, attitude, telemetry, or telecommand control, for which a reactive control loop is necessary onboard, they have no autonomous control capability. They cannot reconfigure themselves autonomously after a subsystem failure. They cannot decide and control autonomously orbital manoeuvres in case of a too large drift from their reference orbit. In case of satellites or probes dedicated to observation of Earth, of other planets, comets, or asteroids, or of the universe outside the solar system, they cannot decide autonomously on the observations they perform. In case of rovers at the surface of planets, they cannot decide on the areas they explore.

For all these tasks, they must wait for decisions made on the ground by human operators or at least under their supervision. The first difficulty is that communication may not be permanently possible between satellites and space probes, on the one hand, and their ground control centers, on the other hand. This is the case with Earth observation satellites for which visibility windows may be rare (about 10% of time) due to their low orbit altitude. This is also the case with planet exploration probes or rovers when they are hidden by the planet. The second difficulty is that communication and normal mission execution may be incompatible. This is the case with planet exploration probes for which communication with Earth, on the one hand, and planet observation, on the other hand, require incompatible spacecraft orientations. The third difficulty is that the communication time may be incompatible with the requirements in terms of onboard reactivity. This is the case with planet exploration probes or rovers. For example, communication takes some tens of minutes between Mars and Earth at the light speed.

The result is a loss in terms of reactivity. In case of subsystem failure, the whole system is unavailable until a communication be possible with the ground control center, human operators make decisions, and send them to the satellite or to the probe. In case of observation systems, observation

[1]CNES: Centre National d'Études Spatiales, French Space Agency, http://www.cnes.fr

[2]ONERA: Office National d'Études et de Recherches Aérospatiales, French Aerospace Lab, http://www.onera.fr

[3]LAAS-CNRS: Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique, Laboratory for Analysis and Architecture of Systems of the French Research Center, http://www.laas.fr

opportunities may be missed. This is the case with the satellites that are currently under consideration for the surveillance of phenomena at the Earth surface, such as forest fires, volcanic eruptions, pollutions, or floods. If they are not equipped with autonomous decision-making capabilities in terms of observation, they may miss important observation opportunities immediately after detection. This is also the case with planet exploration rovers. If they are not equipped with autonomous decision-making capabilities in terms of movement and observation, they may miss scientifically important observation opportunities. To take another example, most of the images taken today by optical Earth observation satellites (about 80%) are lost because of the presence of clouds. To put this right, cloud detection systems in front of these satellites are currently under consideration, but autonomous decision-making capabilities in terms of observation are necessary to take into account onboard the information provided by these systems.

In all cases, autonomy can improve system dependability and global mission return in terms of quantity, quality, and quick delivery of collected data.

Studies about satellite and space probe autonomy have been active since the nineties. One can cite the generic seminal work performed at NASA Ames and JPL in the context of the DS-1 technological space probe [MNPW98] and works performed at JPL in the context of the technological Earth surveillance and observation EO-1 satellite [CST+05, CCD+05], which demonstrated operationally the feasibility of onboard ground phenomena detection and autonomous observation planning and replanning. One can also cite works performed at MIT in the domain of autonomous failure diagnosis and reconfiguration [WICE03]. In Europe, one can cite works in the domain of autonomous Earth surveillance and observation [DVC05] and in the domain of autonomous orbital manoeuvres [LCL+04] whose feasibility has been operationally demonstrated in the context of the Demeter satellite.

On this basis, the goal of the AGATA project is to check off, to understand, to adapt, to develop, and to combine all the technological pieces that are necessary to the development of spacecraft autonomy. To progress in this direction, its short-term objective is to develop a ground simulator of an autonomous spacecraft which will demonstrate, at least in the context of some specific missions, that the current technology allows a spacecraft to be autonomously correctly controlled.

2 A generic architectural framework

One of the first works in the AGATA project was to define what should be the architecture of the software responsible for the control of an autonomous satellite. For that, classical architectures used in robotics were considered. Among them, one can cite the classical three-level planning-centered architectures independently developed at NASA-Ames [MNPW98] and at LAAS-CNRS [ACF+98] and the execution-centered architecture developed at ONERA [BL99]. But, we were especially interested in modular architectures such as the GENOM architecture developed at LAAS-CNRS [FHC94] for the hardware-software interface and the IDEA architecture developed at NASA-Ames [MDF+02]. For the AGATA project, the result has been the definition of a generic architectural framework whose main features are:

1. a decomposition of the system control into hierarchically organized modules;
2. an encapsulation of control and data inside each module;
3. standardized communications between modules via requests, reports about request execution, and information about the system state;
4. a standardized organization of each module around the following four components: tracking of the received requests, tracking of the emitted requests, tracking of the system state, and decision-making upon request emission;
5. a common framework for the interaction between reactive and deliberative tasks inside the module components and especially inside the state tracking and decision-making ones.

One must stress that, although this architectural framework has been developed in the context of the specific AGATA project, its principles are applicable far beyond the space domain, in fact for the architectural design of any autonomous system.

It may be also important to stress that what is discussed in this paper is a functional control architecture and not a software architecture. There are certainly many ways of implementing such an architectural framework.

3 Control decomposition into hierarchically organized modules

The first very simple idea is that, due to the increasing complexity of satellites, it is not reasonable to try and build a unique software module responsible for the control of the whole satellite. As far as possible, independencies must be exploited. This led us to the decomposition of the satellite control into a hierarchy of control modules, each one being responsible for the control of a subsystem.

One can see in Figure 1 a possible decomposition of the control of a satellite dedicated to Earth surveillance and observation. This satellite is equipped with a permanently active detection instrument of wide swath, able to detect phenomena at the Earth surface in front of the satellite, such as forest fires, volcanic eruptions, ... In case of detection, it is able to send an alarm to the ground using the relay of geostationary satellites. Moreover, the satellite is equipped with an observation instrument of narrow swath, active on request and able to take images of the areas where phenomena have been detected. Data produced by this instrument can be downloaded to users on the ground when the satellite is within the visibility of a ground station.

Starting from the top, one can see that the satellite control is, as usually in the space domain, decomposed into a platform control module and a payload control module. Both modules are then decomposed into lower level control modules, each one being responsible for the management of one of the main functionalities in the satellite: orbit, attitude, energy, thermal, telecommand, and telemetry control for the platform, and detection, observation, and data downloading control for the payload. Going deeper, each of these modules is itself decomposed into lower level control modules, called monitors, each monitor being in charge of handling a set of hardware equipments. For example, the GPS monitor is in charge of handling the two GPS receivers present onboard and the thruster monitor in charge of controlling the pool of thrusters that can be activated when one wants to correct the satellite orbital trajectory. Following the ideas of the GENOM architecture [FHC94], the monitors allow the control software to access the hardware via a software interface which is independent from the precise hardware configuration, for example independent from the number of redundant equipments and from the one that is currently used.

In Figure 1, the arcs between modules represent possible requests emitted from one module to another one. For example, the observation module

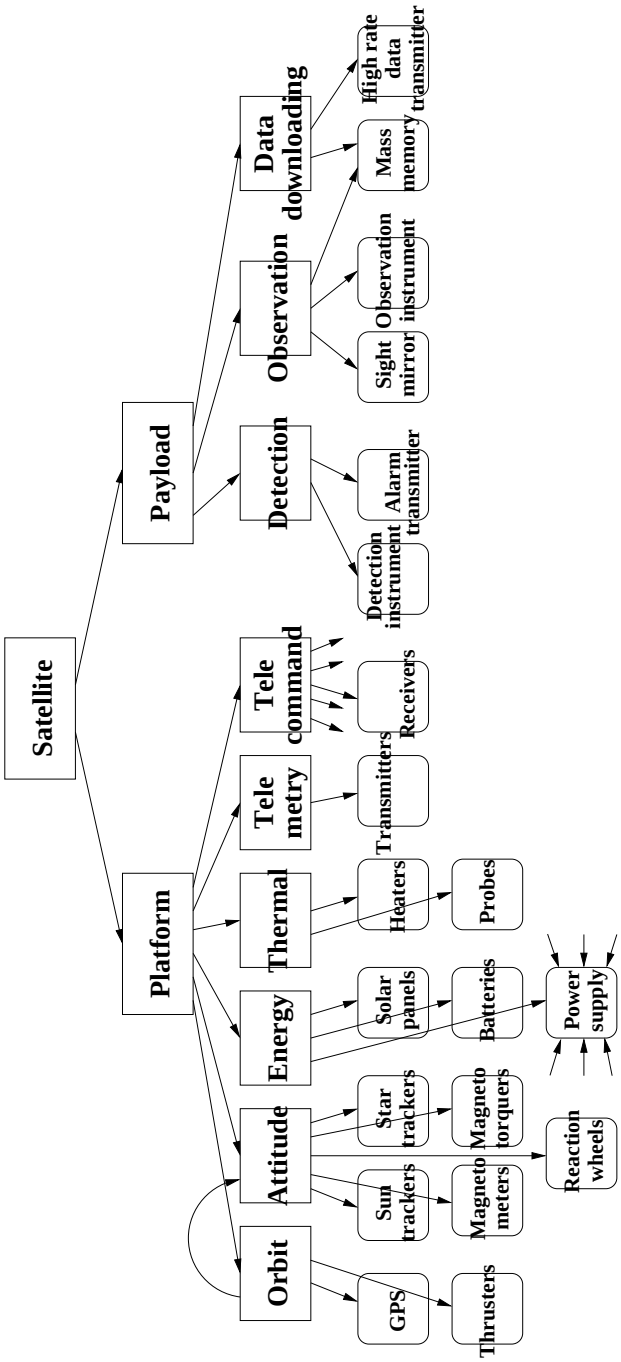

Figure 1: Possible architecture of the control of an Earth surveillance and observation satellite.

can send requests to the sight mirror monitor responsible for directing observation to the right area on the ground, to the observation instrument monitor responsible for triggering observation, and to the mass memory monitor responsible for recording observation data. These arcs result in a directed acyclic graph: no request loop.

But possible dependencies cannot be forgotten. One can see for example that the observation and data downloading modules can both emit requests to the mass memory monitor. These requests must be coordinated by the higher level payload module in order to guarantee that onboard memory be never overflown.

In Figure 1, we only represent the top-down flow of requests from high to low-level modules and do not represent the opposite bottom-up flow of information (reports about request execution and information about the system state) from low to high-level modules. For example, the attitude module gathers information coming from the sun tracker, star tracker, and magnetometer monitors

As dependencies between emitted requests cannot be ignored, dependencies between received information cannot be ignored too. For example, conflicts between information coming from the sun tracker, star tracker, and magnetometer monitors are managed by the higher level attitude module in order to build an estimate of the satellite attitude.

4 Encapsulation of control and data in each module

The second very simple idea is to reuse the principles of encapsulation that are at the basis of *object programming*. In terms of control, that means that, if a module M is in charge of the control of a satellite subsystem S, S cannot be controlled from any other control module M' without a request to M. Moreover, information about the state of S cannot be obtained without an access to the data that are maintained by S.

For example, any request for a change in the satellite attitude must be sent to the attitude module and to no other module, and any information about the satellite attitude must be obtained from it and from no other module.

We think that these principles, although they do not remove all the possible conflicts in terms of requests or information, can greatly help to limit and to manage them.

5 Standardized communications between modules

On this basis, we think that it is possible to standardize the communications between modules, taking into account the main three kinds of exchange that are necessary between them:

1. *control requests* emitted from a module to a lower level one;
2. *request reports* emitted in the opposite direction from a module to a higher level one;
3. *information* about the *system state* from a module to a higher level one.

About requests, one must stress that they are not limited to basic commands immediately and compulsorily executed. Some requests may be complex, such as the regular observation of a ground area. Some are not immediate, such as the observation of a ground area when the satellite will be within its visibility. Some are not mandatory and must executed if possible, such as observations which may conflict with each other. In this case, it may be useful to associate with each request a priority degree which will guide decision-making towards good choices.

About information, one must stress that actual communication mechanisms (systematic information, information on request, ...) and means (message passing, shared memory, ...) depend on implementation choices.

6 Generic organization of each module

Beyond the communications between modules, we think that it is also possible to standardize the organization of each module and to propose a generic organization built around four main components:

1. a *received request tracking* component responsible for receiving requests from higher level modules and for tracking and reporting their execution;
2. an *emitted request tracking* component responsible, in the opposite direction, for emitting requests to lower level modules and for tracking and reporting their execution;
3. a *system state tracking* component responsible for the tracking of the state of the subsystem the module is responsible for;

4. a *decision-making* component in charge of deciding upon the emission of requests to lower level modules in order to answer requests received from higher level modules.

To these main four components, it may be useful to add:

1. a *supervision* component in charge of initializing the module and of managing its possibly different control modes;

2. a *model* component in charge of managing the data that represents the model of the subsystem the module is responsible for; differently from the system state which evolves over time, this model is assumed not to change or to change at a far lower rate;

3. an *information processing service* component which gathers all the data processing services naturally associated with the module, for example the software responsible for orbit, eclipse, and visibility prediction inside the orbit module.

Figure 2 shows the generic scheme of a control module at any level in the module hierarchy.

7 Generic scheme of interaction between reactive and deliberative tasks

An autonomous system must be always correctly controlled in a dynamic environment, with possible changes in the system itself due for example to subsystem failures or in its environment due for example to new observation conditions or new observations to perform. As a consequence, its control must be globally reactive: the control system must be able to react immediately to any event.

Some of the tasks we identified in each control module can be considered as *reactive*, such as received request tracking, emitted request tracking and, in some cases, system state tracking and decision-making. By knowing the maximum event rhythm or by imposing event buffering, we can guarantee that each set of instantaneous events be managed before the following one.

However, some of these tasks cannot be considered as reactive, such as, in some cases, system state tracking and decision-making, or any other complex data processing task. For example, building a failure diagnosis, a predictive resource profile, or an activity plan over a given temporal horizon

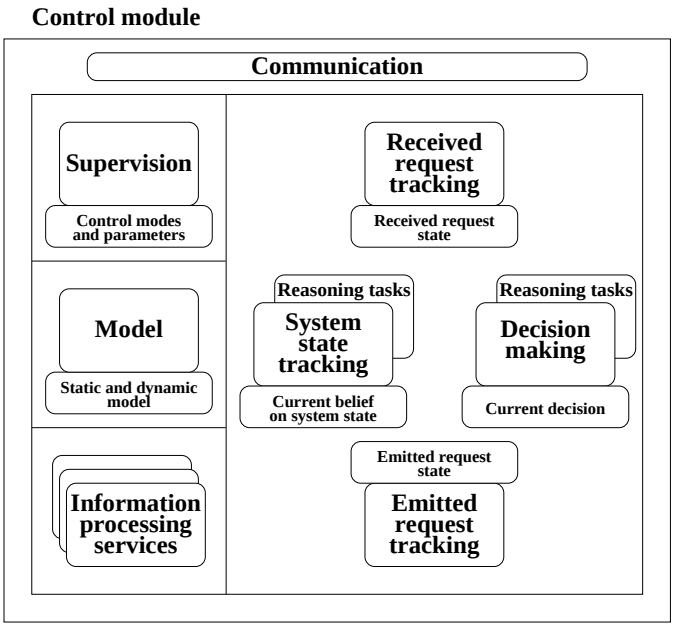

Figure 2: Generic scheme of a control module.

may take a time much greater than the maximum event rhythm or than the buffering rhythm. We refer to these tasks as *deliberative*.

The problem is to define what must be the temporal behavior of the deliberative tasks and what must be their interaction with the reactive ones, if we want them to be useful to the reactive control.

The generic scheme of interaction between reactive and deliberative tasks we propose is summarized in Figure 3. See [LV07] for more details.

According to this scheme, reactive control tasks are in charge of the interaction between the environment, on the one hand, and deliberative reasoning tasks, on the other hand. The latter are never in direct interaction with the environment. Reactive control tasks receive changes from the environment. They may react to them by immediately committing to actions. Note that waiting may be a candidate action. Concurrently, they may compute a deadline for deciding latter on the next action to perform. Then, they may run deliberative reasoning tasks, by providing them with relevant information about changes. On their side, deliberative tasks use this information to produce what we call deliberations, which can be state estimates, failure diagnoses, action proposals, or any other result useful for

decision-making. We assume that deliberative tasks are designed to have an anytime behavior, that is the ability to produce quickly a first result and to improve on it as long as time is available for reasoning. When the deadline occurs, reactive control tasks use the successive deliberations they received from deliberative tasks to make the right decision. If they received no deliberation, they make a reactive default decision.

This scheme requires only that reactive control tasks be able to compute a deadline, to check deliberations before making decisions, to make decisions even when no deliberation has been received, and to perform all of this reactively.

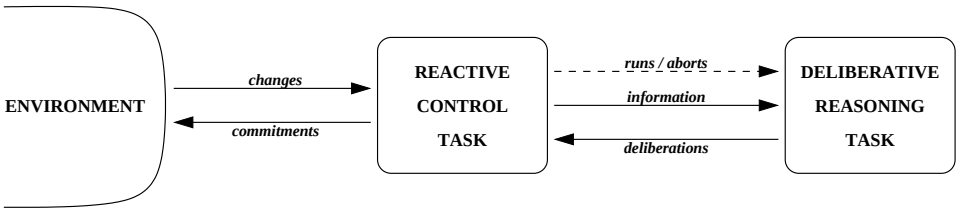

Figure 3: Generic scheme of interaction between reactive and deliberative tasks.

8 Conclusion

We are currently working on applying all these architectural principles to the design and the implementation of a control architecture for an autonomous satellite dedicated to Earth surveillance and observation, such as the one that has been roughly described in Section 3.

The following scenario we will consider is an autonomous agile satellite dedicated to Earth observation, equipped with an optical observation instrument and a cloud detection instrument in front of the satellite, able to provide the module in charge of deciding upon observations with information about the actual cloud cover, in order to avoid imaging clouds, as it is too often the case with currently operational satellites.

Beyond these experiments, we hope that the architectural principles we presented in this paper be applicable for the closed-loop control of many other autonomous systems.

9 Acknowledgements

We would like to thank the people who took part in most of the AGATA brainstorming meetings about architecture, especially Solange Lemai-Chenevier from CNES and Félix Ingrand from LAAS-CNRS.

References

[ACF+98] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An Architecture for Autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

[BL99] C. Barrouil and J. Lemaire. Advanced Real-time Mission Management for an AUV. In *Proc. of the SCI NATO Symposium on Advanced Mission Management and System Integration Technologies for Improved Tactical Operations*, Florence, Italy, 1999.

[CB05] M.-C. Charmeau and E. Bensana. AGATA: A Lab Bench Project for Spacecraft Autonomy. In *Proc. of the 8th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-05)*, Munich, Germany, 2005.

[CCD+05] S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, D. Mandl, S. Frye, S. Shulman, J. Jones, and S. Grosvenor. An Autonomous Earth-Observing Sensorweb. *IEEE Intelligent Systems*, 2005.

[CST+05] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication*, 2005.

[DVC05] S. Damiani, G. Verfaillie, and M.-C. Charmeau. Cooperating On-board and On the ground Decision Modules for the Management of an Earth Watching Constellation. In *Proc. of the 8th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-05)*, Munich, Germany, 2005.

[FHC94] S. Fleury, M. Herbb, and R. Chatila. Design of a modular architecture for autonomous robot. In *Proc. of the IEEE Inter-*

national Conference on Robotics and Automation (ICRA-94), San Diego, CA, USA, 1994.

[LCL+04] A. Lamy, M.-C. Charmeau, D. Laurichesse, M. Grondin, and R. Bertrand. Experiment of Autonomous Orbit Control on the DEMETER Satellite. In *Proc. of the 18th International Symposium on Space Flight Dynamics (ISSFD-04)*, Münich, Germany, 2004.

[LV07] M. Lemaître and G. Verfaillie. Interaction entre tâches réactives et délibératives pour la décision en ligne. In *Actes des Journées Françaises sur la Planification, la Décision et l'Apprentissage pour la Conduite de Systèmes (JFPDA-07)*, Grenoble, France, 2007.

[MDF+02] N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt. IDEA: Planning at the Core of Autonomous Reactive Agents. In *Proc. of the 3rd NASA International Workshop on Planning and Scheduling for Space*, Houston, TX, USA, 2002.

[MNPW98] N. Muscettola, P. Nayak, B. Pell, and B. Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(1-2):5–48, 1998.

[WICE03] B. Williams, M. Ingham, S. Chung, and P. Elliott. Model-Based Programming of Intelligent Embedded Systems and Robotic Space Explorers. *Proc. of the IEEE*, 91(1):212–237, 2003.