Autonomic Computing Software for Autonomous Space Vehicles

Carlos C. Insaurralde[1] and Emil Vassev[2(✉)]

[1] Institute of Sensors, Signals and Systems, Heriot-Watt University, Edinburgh EH14 4AS, UK
c.c.insaurralde@hw.ac.uk
[2] Lero–the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland
emil.vassev@lero.ie

Abstract. Current space missions increasingly demand more autonomy in control architectures for Unmanned Space Vehicles (USVs), so unmanned long-term missions can be afforded. Continuous assurance of effective adaptation to unpredictable internal and external changes, along with efficient management of resources is essential for such requirements. One of the attractive solutions is that inspired by the physiology of living systems, where self-regulation helps to achieve continuous adaptation to the environment by changing internal conditions. The physiological functions are performed by nervous system reflexes that are the foundations for self-regulatory mechanisms such as homeostasis. Building artificial self-regulation similar to biological ones into USVs makes them highly-viable and ultra-stable in order to support very long missions. This paper presents aspects of how to endow USVs with Artificial Nervous Reflexes (ANRs) by means of applying physiological principles of self-regulation to the USV's control architecture, so resilience and persistence can be supported. A case study of a composite orbiter is presented. The studied ANRs are needed to guarantee the self-regulation of response time (latency), operation temperature (thermoregulation), and power consumption (energy balance). Results from a cross-checked analysis of the above self-regulation mechanisms are also presented.

1 Introduction

The technological evolution of Unmanned Space Vehicles (USVs) is making them progressively more sophisticated by increasing the complexity of their structural control architecture (e.g., integration of multiple capabilities for robotic exploration of very large and hostile areas in planet surveys), and the degree of behavioral autonomy (e.g., non-stop operation supporting in-service adaptation to expected and unexpected situations). The main challenge in dealing with the above systems involves the continuous assurance of effective adaptation to unpredictable internal and external changes, and efficient management of resources. One of the attractive inspirations for tackling these issues is that provided by the physiology of living systems, in particular, auto-configuration, auto-reproduction, and auto-regulation abilities. These self-

P.C. Vinh et al. (Eds.): ICTCC 2014, LNICST 144, pp. 33–41, 2015.
DOI: 10.1007/978-3-319-15392-6_4

adapting capabilities endow organisms with resilience having the vital goal of surviving. The self-adaptation is inspired by the physiological functions performed by single/multi-operational combination of nervous system reflexes. It is able to support autonomic management and persistent sustainment (including self-maintenance and self-suitability) in order to make systems more viable and stable. Resilient operation is a qualitative aspect supported by highly-viable and ultra-stable control engineering systems. Applying this system quality to USVs means that they can know how to regulate themselves internally to cope with different external operational conditions. The implementation of this self-management in USVs is rather a very complex development task that requires concurrent control architecture.

The motivation of this research work is to propose a physiologically-inspired control approach for USVs by endowing them with well-defined self-regulatory capabilities to persist (even in adverse conditions), i.e., reflex-driven homeostasis properties as in living systems. By means of homeostasis, a system regulates its internal environment and tends to maintain a stable and constant condition regarding the external environment.

This paper presents aspects of how to endow USVs with Artificial Nervous Reflexes (ANRs) by means of applying physiological principles of self-regulation to the USV Control Architecture, so resilience and persistence can be supported. The architectural approach is realized on the basis that autonomy requirements for USVs are satisfied. A case study based on orbiters for the BepiColombo Mission to Mercury [1] is presented. The ANRs, studied in this paper for those orbiters, are needed to guarantee the self-regulation of response time (latency), operation temperature (thermoregulation), and power consumption (energy balance).

The rest of the paper is structured as follows. Section 2 presents a review of fundamental biology concepts and related work. Section 3 presents the Autonomic System Specification Language (ASSL) used in this project to specify the ANRs. Section 4 presents a case study based on the BepiColombo Mission where an algorithm of ANRs for the mission's orbiters is proposed. Section 5 presents our experiments and results. Finally, the last section presents concluding remarks and directions for future work.

2 Related Work

The nervous system has neural pathways named reflex arcs that control reflex actions in order to implement regulatory functions. Reflex arcs are divided into two types: somatic reflex arcs and autonomic reflex arcs. The former are reflexes from SNS classified as withdrawal, stretch, and extra-pyramidal reflexes. The latter are from the ANS classified as autonomic reflexes. Same examples of reflexes are [2]:

- Withdrawal Reflexes, e.g., pain impulses initiated by touching a very hot surface with the finger.
- Stretch Reflexes, e.g., the knee jerk; the sensory nerve endings in the tendon and the thigh muscles are stretched by tapping the tendon just below the knee when it is bent.

- Extra-pyramidal (Upper-Motor) Reflexes, e.g., maintenance of upright neck and head where many muscles are contracting in a coordinated manner.
- Autonomic Reflexes, e.g., the self-regulation of the cardiovascular functions such an increase of the heart rate to increasing blood pressure.

Homeostasis is the property of a system to maintain stable its condition regarding the external environment by regulating its internal environment.

Major pioneering research is focused on mobile robots as an excellent test-bed for research on Autonomic Computing (AC) [3] and Organic Computing (OC) [5]. It recognizes self-management power by exploring the use of AC techniques in the domain of ground-based mobile robots [4]. The main focus is on robustness and fault-tolerance. This research work only presents the ideas to apply AC to mobile robots but not any implementation.

Active adaptation of systems requires non-stop monitoring and control. Thus, the two main OC components are an observer and a controller dealing with the system under observation and regulation. Since its emergence, OC has brought the attention of researchers from different domains. Once the methodology has been proposed, the question is how to design and implement OC systems. Some approaches coin the combination of model-driven engineering with OC [6].

The viability provided by the Viable System Model (VSM) [7] is based on the ultra-stability concept. A system is said to be ultra-stable when it can survive arbitrary and un-forecast interference. This high stability is also applied to systems that are able to deal with various principles for states. If a system can cope with its environment by successfully absorbing the variety from it (attenuating the incoming variety, and amplifying its own variety when needed), it achieves an ultra-stabile state. If a system is capable of working in such a manner, then it can maintain homeostasis. This means it can maintain itself in a state of equilibrium. Maintaining a balance of variety is essential for self-organizing systems. An approach based on VSM principles was used to build resilience into enterprise systems [8]. It demonstrated how a combination of systems thinking and a physiology inspiration based on homeostatic mechanisms of the human body can provide a blueprint for resilience.

The idea of building a self-adaptable man-made system capable of taking into account environment changes was proposed by mid-20th century. The “homeostat”, as its inventor W. Ross Ashby called it, was developed to support habituation, reinforcement and learning through its ability to maintain homeostasis in a changing environment [9]. The homeostat caught the attention of the control community that saw it as an interesting implementation for adaptive control based on cybernetics and general systems theory [10].

3 Autonomic System Specification Language

The Autonomic System Specification Language (ASSL) [11, 12] is defined through formalization tiers. Over these tiers, ASSL provides a multi-tier specification model that is designed to be scalable and exposes a judicious selection and configuration of infrastructure elements and mechanisms needed by an AS. ASSL defines the latter

with interaction protocols and autonomic elements (AEs), where the ASSL tiers and their sub-tiers describe different aspects of the AS under consideration, such as policies, communication interfaces, execution semantics, actions, etc. There are three main tiers in the ASSL specification model:

- The AS Tier specifies an AS in terms of service level objectives (AS SLO), self-management policies, architecture topology, actions, events, and metrics. The AS SLO is a high-level form of behavioral specification that establishes system objectives such as performance. The self-management policies could be the four self-management policies (also called self- CHOP) of an AS: self-configuring, self-healing, self-optimizing, and self-protecting, or they could be others. The metrics constitute a set of parameters and observables controllable by the AEs.
- At the AS Interaction Protocol tier, the ASSL framework specifies an AS-level interaction protocol (ASIP). ASIP is a public communication interface, expressed with channels, communication functions and messages.
- At the AE Tier, the ASSL formal model considers AEs to be analogous to software agents able to manage their own behavior and their relationships with other AEs. In this tier, ASSL describes the individual AEs of the AS.

We used ASSL in this project to specify the orbiters' ANRs.

4 Autonomic ANRs for BepiColombo Mission

The BepiColombo Mission is to be performed by two orbiters: a Mercury Planetary Orbiter (MPO) and a Mercury Magnetospheric Orbiter (MMO) [1]. The physiologically-inspired adaptation for the orbiters is defined through three self-regulatory functions based on autonomic ANRs by parameterizing the autonomicity and quality attributes of BepiColombo [13].

There are three parameters in the studied USVs (MPO and MMO) that are under self-regulation: (1) the end-to-end latency regulation; (2) the system-context temperature regulation; and (3) the power consumption regulation. These self-regulated parameters have the following requirements as to operation ranges.

The states generated by the USV latency (L):

$$S_L(L) \equiv \{s(L)\} \quad \forall L \in 50\ \mu s < L < 200\ \mu s$$

The states generated by the USV temperature (T):

$$S_T(T) \equiv \{s(T)\} \quad \forall T \in -65\ ℃ < T < 175\ ℃$$

The states generated by the USV power (P):

$$S_P(P) \equiv \{s(P)\} \quad \forall P \in 10\ W < P < 30\ W$$

The states of homeostatic balance in the AES (S_{hb}) are derived as the intersection of the above states:

$$S_{hb} = S(L) \cap S(T) \cap S(P)$$

The states of homeostatic imbalance in the AES (SHI) are the exclusion of states given by:

$$S_{hi} = S(L) / S(T) / S(P)$$

The self-regulation of the above three operational parameters (L, T, and P) is working as follows:

End-to-end Latency Regulation (L). This requirement is basically to optimize by guaranteeing the USV performance in terms of time response, including processing and communication times. The USV under study is required to work within a certain end-to-end latency range (50 µs < L < 200 µs), no matter the processes and tasks it has to execute, and in order to get its right time response. Thus, any system state generated by L between 50 µs and 200 µs makes the USV to be in homeostatic end-to-end latency balance. Otherwise, the USV is in homeostatic end-to-end latency imbalance.

System-Context Temperature Regulation (T). This requirement is basically to optimize by guaranteeing the USV performance in terms of temperature of operation. The USV temperature can be affected by the heat generated by the electronic devices and other heat sources inside the system as well as outside it (environment). The USV under study is required to work within a certain temperature range (-65 °C < T < 175 °C), no matter the environmental temperature the USV has to deal with, and in order to maintain operational performance. Thus, any USV state generated by T between -65 °C and 175 °C makes the USV to be in homeostatic temperature balance. Otherwise, the system is in homeostatic temperature imbalance.

Power Consumption Regulation (P). This requirement is basically to optimize by guaranteeing the USV performance in terms of power consumption. The USV under study is required to work within a certain power range (10 W < P < 30 W), no matter the USV operation, and in order to make a good use of the energy. Thus, any USV state generated by p between 10 W and 30 W makes the USV to be in homeostatic power consumption balance. Otherwise, the USV is in homeostatic power consumption imbalance.

The artificial homeostatic balance state (collective; three-parameters) can be formally specified as follows.

$$S_{hb} = S_L \cap S_T \cap S_P, \forall 50\ \mu s \leq L \leq 200\ \mu s \wedge \forall \text{-}65\ °C \leq T \leq 175\ °C \wedge \forall 10\ W \leq P \leq 30\ W$$

Any other state outside S_{hb} makes the system to be in homeostatic imbalance. The homeostatic balance states can be formally defined as follows:

$$S_{hi} = S_L / S_T / S_P\ \forall 50\ \mu s \leq L \leq 200\ \mu s \wedge \forall \text{-}65\ °C \leq T \leq 175\ °C \wedge \forall 10\ W \leq P \leq 30\ W$$

5 Experiments

The self-regulation behavior based on the three operational parameters (**L**, **T**, and **P**) was specified with the ASSL framework and consecutively, Java code was generated. For more information on the self-regulation specification model, please, refer to [13]. Note that all Java applications generated with ASSL can generate run-time log records that show important state-transition operations ongoing in the system at runtime and the behavior of the generated system can be easily followed by the generated log records. Hence, the log records produced by the generated Java application for the ASSL self-regulation specification model for USVs allowed us to trace the simulated USV behavior and so, to perform a variety of experiments outlined in this section.

The USV deals with the following two drivers that impact on the self-regulatory functions (applied to the parameters under auto-regulation, i.e. end-to-end latency, system-context temperature, and power consumption) based on autonomic ANRs:

- **Data Processing Rate (DPR).** The USV under study is able to process data and dispatch tasks with a sampling time, an execution time, a deadline, a delivery time, and within a range defined by them from 825 Mips to 3300 Mips. The USV data processing is defined by delays generated by software/hardware controllers. A typical application case is that a USV is a real-time system. Thus, data processing time constraints must be guaranteed all the time.
- **Data Transfer Rate (DTR).** The USV under study is able to transfer data up to 2000 Mbits/sec. However, it is required to work at least at 500 Mbits/sec in order to keep a desired operational performance. The USV optimizes its performance by increasing the transfer rate when the data volume is bigger, and decrease it when the data volume is smaller. USV DTR can also be changed when a priority list for messages is applied; transmission rules for different communication channels are set or network availability policies are required.

Increments in the DPR increase the temperature and power consumption of the USV but decrease its end-to-end latency. On the contrary, decrements in the DPR decrease the temperature and power consumption of the USV but increase its end-to-end latency. It is the same situation for the DTR. In addition, a higher USV clock speed means an increase of the DPR and DTR, and a lower one means a decrease of them. There are other drivers that influence on the USV parameters under regulation such as the environment temperature, and cooling mechanics of the USV. A higher environment temperature increases the USV temperature, and lower one decreases the USV temperature. An activated cooling mechanism lowers the USV temperature but increases the power consumption and the environment temperature.

The cross-checked tests of the self-regulatory functions and their drivers is carried out by running a software application which code was automatically generated by the ASSL framework.

Different tests were carried out. They go from reduced-load system operations with slight load changes up to full-load system operations with strong load changes. Reduction in loads entails the USV in a state defined as follows: power-up and communication and computation load (randomly variable but up to 20 % of the maximum).

Full loads entail the USV in an operational state where transferring and processing loads were simulated in order to evaluate the USV self-regulatory capabilities under load (randomly variable, and up to 100 % of the maximum).

The following figures show results from the tests of the self-regulatory mechanisms studied on a USV. They show the cross-regulation impact during 140 minutes, and with a randomly-variable load on the data processing and transferring rates. Figure 1 shows the evolution of the self-regulation for the system-context temperature (one of the three parameters under self-regulation) when the loads can vary from 0% to 100 % of the maximum value (with slight load changes from 0 min to ~65 min, and strong ones from ~65 min to 140 min).

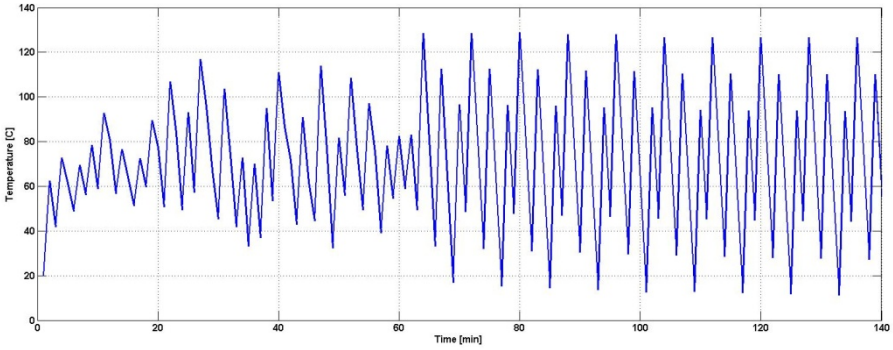

Fig. 1. Thermoregulation performance (variable load) for a USV

Just after the USV is started up, its temperature quickly reaches 60 ºC, and from there it starts increasing and decreasing based on the system demands. The power consumption evolution is according to the system process, and somehow it is proportional to the variation of the temperature which makes sense. The end-to-end latency got an increase at the beginning (no transferring or processing demands).

Figure 2 shows a snapshot of the trace when the application is executed. When the system temperature (ST) reaches 60 ºC, the cooling mechanism is activated. This makes ST to drop for a while and then ST varies according to the workload of the USV (transfer and processing loads) as shown in Figure 1. Figures 1 and 2 show results of the performance of the self-regulation mechanism implemented in the USV when transferring and processing loads are applied by means of slight and strong load changes. The USV starts in idle state, and then some variable loads are applied at ~4 min and ends at ~65 min. Then, some stronger variable loads are applied again at ~65 min and end at ~135 min. Then, an abruptly-decreased load is applied. In any case, the USV jumps accordingly from one state to the other, and comes back to the previous state adequately.

There is a slight increment in the power consumption when the cooling mechanism is activated but no changes are seen on the end-to-end latency due to this increment since the power consumption remains below the limit configured (30 W). This energy rise does not make any change in the system clock that can indirectly modify the end-to-end latency (through data processing and transfer rates). The cooling mechanism prevents the system temperature going beyond 140 ºC (upper temperature

threshold for performance optimization), and helps dissipate more temperature in the system even though the USV could support temperatures up to 175 ºC. The above figures do not show the evolution of the drivers, i.e., environment temperature, DPR, and DTR.

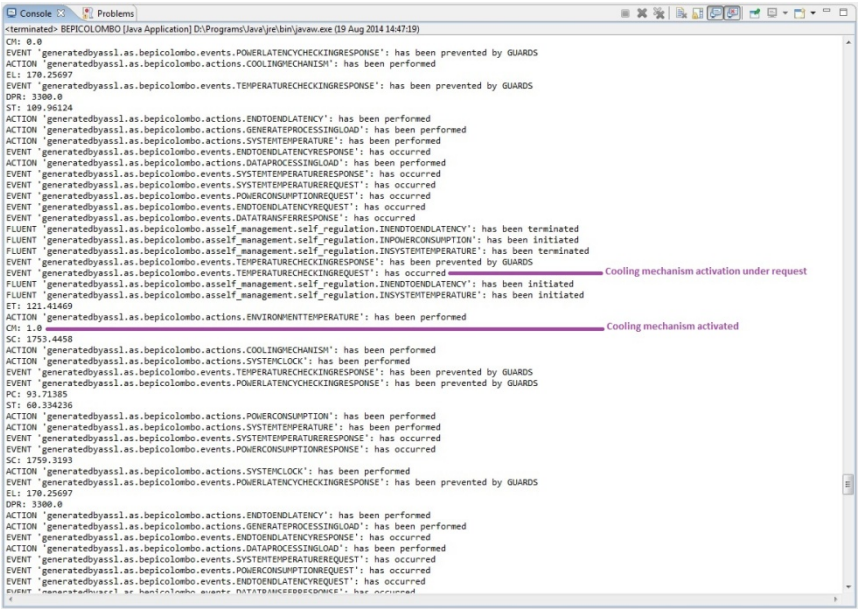

Fig. 2. Generated Java Packages for USVs

All three parameters are simultaneously and successfully regulated along the USV operation with and without workload. No one of them goes beyond the boundaries set by the artificial homeostasis principles to optimize the USV performance.

6 Conclusions

An approach to implementing operational resilience and persistence based on ANRs for USVs has been presented in this paper. Three reference technologies inspired by human physiology as well as biological foundations have been reviewed. A case study on orbiters (USVs) for the BepiColombo Mission to Mercury and outcomes of experimental tests have been presented. Initial results show the feasibility of the approach proposed. Three self-regulatory functions based on autonomic ANRs for Mercury orbiters (USVs) have been identified to show how physiological principles of self-regulation can be applied to the USV control architecture. This approach is able to comply with the USV autonomy requirements and extends the USV autonomicity through other self-managing capabilities that are suitable for either manned or unmanned spacecraft.

Future work will be mainly concerned with further development of the approach presented in this paper, including adding more self-regulated parameters and an improvement of the current code generation. It will also integrate KnowLang [14] – a formal framework that can be particularly used for formal specification of ANRs.

Acknowledgments. This work was supported by ESTEC ESA (contract No. 4000106016), by the European Union FP7 Integrated Project Autonomic Service-Component Ensembles (ASCENS), and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero–the Irish Software Engineering Research Centre.

References

1. Benkhoff, J.: BepiColombo: Overview and Latest Updates, European Planetary Science Congress, p. 7. EPSC Abstracts (2012)
2. Waugh, A., Grant, A.: Anatomy and Physiology in Health and Illness. Ross and Wilson (2004)
3. Horn, P.: Autonomic Computing: IBM's perspective on the state of information technology. IBM Research Report (2001)
4. Melchior, N.A., Smart, W.D.: Autonomic systems for mobile robots. In: Proceedings of the 2004 International Conference on AC, New York, USA (2004)
5. Schmeck, H.: Organic computing – a new vision for distributed embedded systems. In: Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. IEEE Computer Society (2005)
6. Kishi, T.: Model driven design and organic computing: from the viewpoint of application production. In: Proceedings of the IEEE International Symposium on ISORC 2009, pp. 97–98. IEEE Computer Society (2009)
7. Beer, S.: Brain of the Firm. 2nd ed. Wiley (1994)
8. Hilton, J., Wirght, C., Kiparoglou, V.: Building resilience into systems. In: Proceedings of the International Systems Conference, Vancouver, Canada (2012)
9. Ashby, W.R.: The William Ross Ashby Digital Archive (2014). http://www.rossashby.info/index.html
10. Cariani, P.A.: The Homeostat as Embodiment of Adaptive Control. International Journal of General Systems **38**(2) (2008)
11. Vassev, E.: ASSL: Autonomic System Specification Language - A Framework for Specification and Code Generation of Autonomic Systems. LAP Lambert Academic Publishing, Germany (2009)
12. Vassev, E.: Towards a Framework for Specification and Code Generation of Autonomic Systems. Ph.D. Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (2008)
13. Insaurralde, C.C., Vassev, E.: Software specification and automatic code generation to realize homeostatic adaptation in unmanned spacecraft. In: Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E 2014), pp. 35–44. ACM (2014)
14. Vassev, E., Hinchey, M., Montanari, U., Bicocchi, N., Zambonelli, F., Wirsing, M.: D3.2: Second Report on WP3: The KnowLang Framework for Knowledge Modeling for SCE Systems. ASCENS Project Deliverable (2012)