EXPERIENCES IN PRODUCING A PRELIMINARY NAVIGATION OBSW PROTOTYPE FOR THE EXOMARS ROVER BASED ON EDRES

Anthony O´Dwyer (1), Raul Correal Tezanos (1), Sebastián Sánchez Prieto (2), Pablo Parra Espada (2).

(1) TCP SISTEMAS E INGENIERÍA, Fernández Caro, 7, 28027, Madrid, Spain.
(2) Departamento de Automática, Universidad de Alcalá, Alcalá Henares, Spain.
(agodwyer@tcpsi.es[1], rcorreal@tcpsi.es[1], sebastian.sanchez@uah.es[2], pablo.parra@aut.uah.es[2]).

ABSTRACT

ExoMars is a European led exploration mission to Mars including a rover that is planned for launch near 2013.

EDRES (Environnement de Développement pour la Robotique d'Exploration Spatiale) is the result of thirteen years of software development for autonomous rovers at CNES. It consists of a collection of algorithms, applications and tools covering the majority of the functions required for autonomous movement generation and execution for exploration rovers. EDRES has been built to serve as a workshop for the creation of new algorithms, tools or applications.

Related to the early developments in the navigation OBSW, TCPsi under contract from Astrium UK and in co-operation with ESA, CNES and the University of Alcalá has undertaken a project that focuses on the assessment of the flight suitability of heritage code developed by CNES, encompassing its ability to run on the flight processor, its performance and its flight-worthiness in terms of issues such as compliance with flight software standards. This involved porting developed SW from a PC and Linux environment to a more flight representative processor, a LEON2-Pender board with RTEMS.

EDRES addresses in a very comprehensive way the need to simulate, design and evaluate complex designs related to rover design and software development. It contains a rich set of resources useful for the software development of any given rover.

The ExoMars navigation flight software will meet ESA mission requirements and as such will be subject to strict development methods and standards.

Typically, the software development process for OBSW starts with no code, an engineering design and a software requirements set then iterates with testing until a flight software is available and validated. In this case, in the B1 phase and before a mature SW design has been elaborated, an evaluation navigation OBSW was produced. This task facilitated the assessment of previously developed code modules and algorithms based on (but not limited to) EDRES. The OBSW navigation program was ported to a candidate flight representative processor. An evaluation of the performance of the navigation SW running on the flight representative processor was done.

This paper details the experiences and lessons learnt during the porting process and the subsequent study to analyse the performance of the navigation SW within the scope of the early phases of the ExoMars rover project.

INTRODUCTION

The CNES ANW SW package [1] has been ported from a PC/LINUX environment to a flight representative LEON2 target processor using RTEMS. The correctness of the porting was demonstrated by detailed testing and the ported Navigation BB SW put under analysis and performance testing. The ported and modified NAV BB SW (rover prototype navigation OBSW) and documented analysis was delivered to Astrium UK. The NAV BB SW was later integrated by Astrium UK into the integrated mars rover prototype [2]. The analysis and OBSW prototyping carried out by TCPsi and UAH have served to consolidate and mature the EXOMARS rover navigation sub-system mission requirements, design and interfaces in the context of the early EXOMARS project phase (B1).

The EDRES [3] contains two principal programs, the RS (Rover Simulator) and the ANW (Autonomous Navigation Workshop). When considering the complex SW modelling of a rover and its environment, it is the RS that contains the environment simulator and the ANW represents the on-board SW with navigation logic. The RS stimulates the ANW in a full closed loop simulation and both together may be installed on a PC with Mandriva-LINUX OS or distributed on two PC machines (TCP-IP sessions are offered between programs).

The RS incorporates a 3D Mars/Moon terrain model, rover position, attitude and surface contact solver.

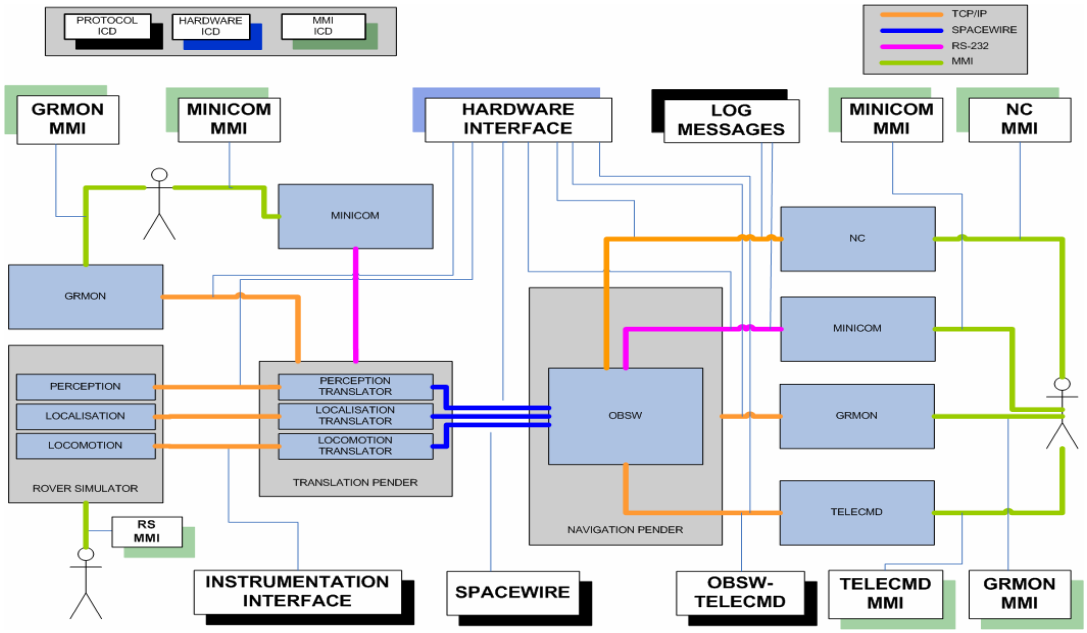

Fig. 1. Two Pender board configuration with ported NAV BB SW/OBSW.

The ANW is composed of vision, navigation/ path planning, execution and localisation routines.

The ANW Navigation sub-system may be further decomposed to SW modules and functions corresponding to the major on-board processing steps of image acquisition, degradation of resolution, lens distortion correction, stereo, disparity map and filtering, local navigation map, global navigation map and path planning. The navigation maps show unknown, navigable and obstacle zones in a rover based reference system. The inputs to the ANW, Navigation SW (OBSW) element are image pairs from the perception sub-system (that has been previously stimulated by the terrain model or real images) and the output of the navigation SW is an array of via and waypoints ending in the target position that is located on the global navigation map.

The Post-Porting HW-SW simulation environment set-up is illustrated in Figure 1. The two LEON2 pender boards that appear in Figure1 have the following specifications:

Model: ESA Sparc V8, GR-CPCI-XC4V, manufacturer Pender/Gaisler
Frequency: 71Mhz
Operating system installed: RTEMS 4.6.5
Memory: 128Mbytes

PORTING PHASE

In the development of a space flight project, several independent components that interact must be taken into account. Usually different development groups must address decisions related to the design of those components. In order to speed up the design, implementation, testing and validation cycle, the use of simulators and synthetic environments is a must. The use of those elements on space flight projects greatly simplifies the modelling of critical elements of a mission and allows a comprehensive validation and testing of algorithms used during the development process. Once the system has been designed and validated, the selected techniques, algorithms and protocols must be ported to a real platform. Usually the target platform differs on hardware, memory size, architecture, performance, etc. to the ones used on the simulation environment.

On this project, we have ported and verified an example (B1 phase) prototype ExoMars rover navigation SW package from a simulation environment based on a Linux i386 platform to a candidate flight representative processor a LEON2 SPARC v8 platform using RTEMS real time operating system.

In the design of the (B1 phase) ExoMars rover, four different components or servers have been considered: perception, locomotion, localization and navigation. Each component can run independently, even on different computers because

they are interconnected through TCP/IP links. The perception server is responsible of providing two pair of images from a synthetic 3D environment, based on the relative position of the rover, to the navigation software. The locomotion server simulates the physical characteristics of the rover and is responsible of the displacement of the rover taken into account the characteristics of the terrain. The localization server provides the relative position of the rover in the 3D environment. Finally, the navigation software is responsible of the navigation of the rover based on the information provided by the perception and localizations servers and acting on the locomotion server.

EDRES software originally runs on a simulation environment based on a Linux i386 platform. In the early stages of development of the navigation software, one of the key points was to be able to test and analyse the performance of the different algorithms of EDRES on a candidate flight representative processor, such as the LEON2 processor, executing the RTEMS real time operating system. LEON2 is a 32-bit processor based on the SPARC V8 architecture. It was developed by Gaisler Research and the European Space Agency (ESA). RTEMS is specifically designed for embedded systems. It has been ported, apart from the LEON2 processor, to numerous platforms and architectures, such as ARM and Intel i386. This section details the adaptation process and experiences and lessons learnt, the following section addresses the subsequent study to analyse the performance of the navigation software.

Autonomous Navigation Workshop

One of the main components of EDRES is the Autonomous Navigation Workshop (ANW). ANW is a software application developed as a test bed for the different algorithms involved in the autonomous navigation process, such as stereovision or path planning. ANW has: 1) a front-end, which allows the execution of the different algorithms and shows the results in a human-readable way; 2) the code of the algorithms and 3) the interfaces needed to obtain the data from the instruments and to command the different mechanisms of the autonomous rover (whether it is a real or a simulated one).

The front-end uses OpenMotif graphical libraries to display the content windows. The interfaces with the different instruments of the rover are designed to allow the abstraction of the code of the algorithms and the front-end from the real data source. By doing this, the ANW can be connected to a real rover platform or to a simulator that emulates the rover instruments. Another component of EDRES is the Rover Simulator. The Rover Simulator (RS) is a complex piece of software that simulates the locomotion mechanisms of the rover on an emulated environment, as well as the vision and localization instruments. The original setup of EDRES then was the ANW application connected to the RS simulator using TCP/IP connections. A schema is shown on Figure 2.

The code of the application is a monolithic block that contains the front-end, the interfaces and the algorithms. The connection between these three different blocks is done via direct calls to the code functions.

Adaptation of ANW: the On-Board Software for porting

The monolithic design of the ANW made it difficult to perform a quick adaptation to an embedded platform, such as one compatible with the RTEMS operating system and the LEON2 processor. The approach chosen to perform the adaptation was to separate the front-end from the actual execution of the algorithms. A schema of this adaptation is shown in Figure 3a.

The original ANW was divided into two applications. One piece of software, called Telecmd, contains the front-end (MMI) while the other, named On-Board Software (OBSW), is the one that actually computes the algorithms and is executed on the embedded LEON2 platform. A new message-passing communication protocol between the new fronted and the embedded application was established. The physical connection was made using TCP/IP over Ethernet.

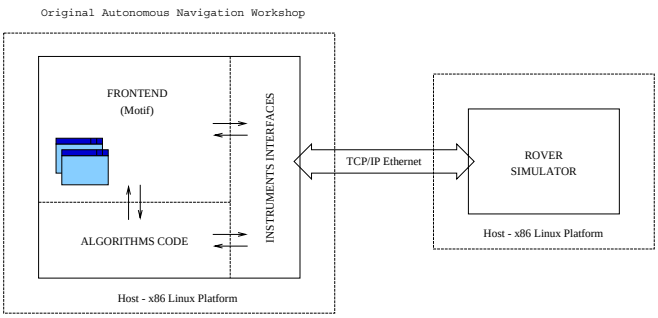

Fig. 2. Original setup of the Autonomous Navigation Workshop and the Rover Simulator.

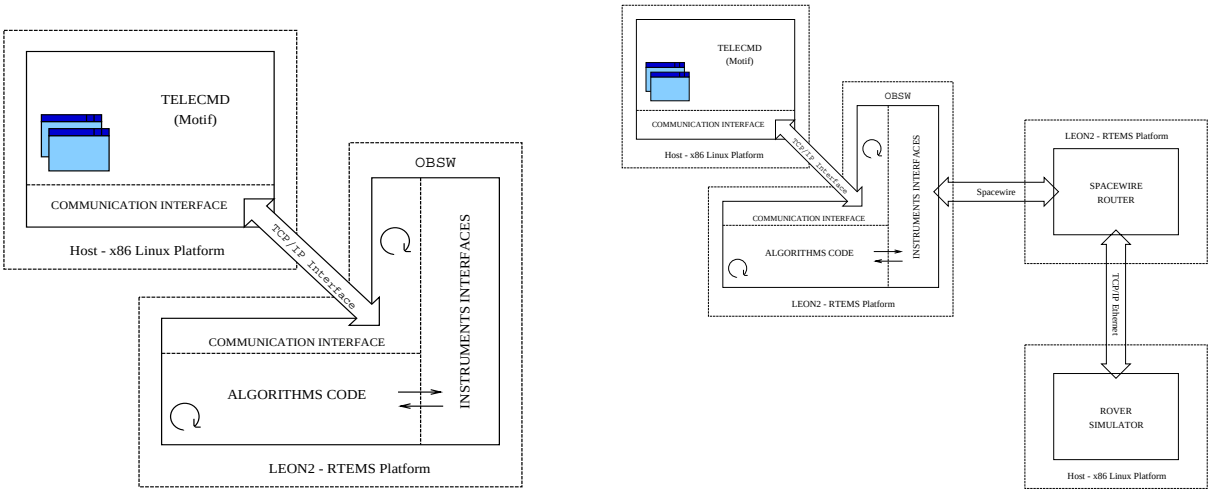

Fig. 3. a) Embedded application schema, b) final setup of the test bed.

The original ANW was divided into two applications. One piece of software, called Telecmd, contains the front-end (MMI) while the other, named On-Board Software (OBSW), is the one that actually computes the algorithms and is executed on the embedded LEON2 platform. A new message-passing communication protocol between the new fronted and the embedded application was established. The physical connection was made using TCP/IP over Ethernet.

OBSW execution is divided into two threads. One thread of execution is in charge of maintaining the communication between the embedded application and Telecmd. It receives the commands from the user and stores them into a queue. The second thread extracts the pending commands from the queue, executes them and returns the results to Telecmd application.

The communication protocol between Telecmd and the OBSW is basically a command-response mechanism. The front-end codifies, requested by the user, the operations to be performed by the OBSW on the board.

The whole porting process was verified by comparing the results obtained by the execution of the different steps of the navigation software on both the original platform and on the embedded one. Both platforms were initialized with the same input data and the results of the different steps were traced for later comparison. When all the steps were performed, the results were compared and checked that they were the same except with negligible differences.

Added to the original TCP/IP protocols, we included support for using Spacewire connections between the OBSW and the simulator. Spacewire is a high-speed communication network specifically designed for space applications. To minimize the impact on the RS simulator side, the connections between the OBSW and the RS are done in a two stages way. Figure 3b shows the connections schema. The LEON2 board executing the OBSW is connected to another LEON2 board in charge of routing the connections to the simulator. The RS side thus remains unchanged, since it receives and sends the application data using the same TCP/IP links.

Challenges encountered in porting

As mentioned, EDRES and, by extension, ANW is a complex piece of software and a result of many years of study and research. Many researchers have worked on its development. This fact explains the complexity of the internal structure of the application. The initial work that had to be done was similar to a reverse engineering process in order to understand the very nature of the application and to find the location of the different blocks shown in Figure 2.

Another challenge was encountered when performing the comparison between the results obtained by the execution of the algorithms on the original and the embedded platforms. The important issue that complicated the performance of those comparisons was the different machine epsilon showed by the two platforms. The machine epsilon measures the relative error due to rounding of floating point numbers. A different machine epsilon of two platforms implies that the same floating point operations with the same input data performed on the two platforms could lead to different results. This fact almost made impossible any data comparison. However, with a special compilation option, the machine epsilon on the Linux host could be reduced and adjusted to the one of the embedded platform, so both platforms shows the same behaviour. By doing this, it severely reduced the differences, which in the end were only negligible and due to the fact that we had to use different versions of the mathematical library to perform the complex operations of the algorithms.

ANALYSIS PHASE

After porting and modification of the ANW to form the NAV BB SW for execution on the Navigation Pender board and development of a translation interface (Spacewire to TCP-IP & visa versa) for the three sub-systems perception, localisation and locomotion, the characterisation of the NAV BB SW by analysis and test began. Based on the experience and knowledge gained, recommendations to raise TRL level and improve the NAV BB SW as a flight SW were made.

The analysis and test covered the following areas:

Multi-tasking:

Cache misses & flushing as percentage of scheduler allocated duration
varied memory use by dummy task
Cache flush frequency adjusted

Scenario and Configuration Variation of terrain characteristics:

Variation of key parameter values of the terrain
Variation of navigation SW parameters
Comparison of robot physical configuration data sets (CNES/ASU)
Navigation, maximum path length
Latency
Navigation performance Vs camera resolution

Resources usage:

Program size
Theoretical memory analysis
Dynamic memory resources
Dynamic memory resources top level functions
Processor performance
Global processor load
Top level functions processor load
Navigation comparison to locomotion
Bottlenecks
Bottlenecks vs. configurations

Worst case:

Maximum duration to complete a calculation cycle
Maximum duration to complete first panorama and merge
Maximum duration to complete a reference test path

Various SW path coverage tests and estimations.

An example of some of the most important numerical results of the default system design were:

- Pender NAV BB SW Image size: 2.7 Mbytes
- Static memory use: 1.4 Mbytes
- Minimum memory for RTEMS: 0.5Mbytes
- Heap memory allocated: 266Mbytes, uses 26 Mbytes
- Dynamic memory needed: 27.7 Mbytes (covers peek memory users of disparity filter and path planning)
- CPU load is at 71% for Navigation element of OBSW and 78% for OBSW with communications SW
- Peek CPU loads: stereo algorithm drives CPU to 99.3%, the path planning to 93%
- Bottlenecks of path-planning, degradation and filter disparity were identified
- For more detailed analysis and data see [4]

An example of some of the most important design drivers and system characteristics noted:

- Camera Resolution (and thus the number of pixels to treat) has a major impact on many function execution durations and system memory use. Image acquisition, degradation, distortion correction, and stereo algorithms generally proportionally affected

- Duration to complete the filtered disparity map is a function of the complexity of the image pair and is not fixed. (Many complicated obstacle shapes and forms various distances provokes much longer function execution time than flat obstacle-less terrain)
- Extreme sensitivity to the “discontinuity planification parameter” value (largest unknown area that can be included in the path plan)
- “Planification Perimeter” dictates the number of way-points in the path plan. It is critical to tune and has a great duration and energy use impact
- Increasing light intensity decreases the time to traverse a reference path. However, with sun at 90% (no shadows) a peek run duration is noted
- Present rover system design in (NAV BB SW) is “typically” stopped for 60% of the time and in motion 40% of traverse duration
- Present rover system design in (NAV BB SW) is doing SW processing and environment sensing when stopped and CPU load is practically idle when in motion
- Perceived Obstacle resolution dimensions and shape are strongly linked to the rover radius parameter and affect the probability to make a narrow pass between obstacles

Some of recommendations for improvement reported:

The EDRES (of July 2007) is a design environment and offers a rich set of resources for rover development; it was requested by contract to evaluate what would be needed beyond the ANW to produce a fully functional flight rover OBSW.

- Inclusion of operation based MODES in the OBSW design: e.g. initialisation mode, SW maintenance mode, direct commanding mode, autonomous mode, safe mode, TM/TC earth link mode. And a mode state transition machine, (mode change permission table). Mode changes may be by Auto, TC, or FDIR transition
- Inclusion of Telemetry and Telecommand processing modules compliant with ESA TM/TC packet format standards
- Inclusion of a Failure Detection Isolation and Recovery module. This covers data value replacement, sensor/actuator health checks and replacement, logic to do processor reset and switch to redundant processor
- Addition of a Real Time scheduler. Respecting the real time RTEMS OS, the Linux based high level application call chain may need modification to produce a “round robin” repetitive/deterministic call chain in an RTEMS compatible scheduler
- Various optimisations of algorithms w.r.t CPU, memory and duration use
- Proposed modified functionality to vary the waypoint to waypoint maximum distance as a function of determined terrain characteristics. This to optimise autonomous navigation speed
- Proposed new function to convert rover coordinate reference system to a planetary based (latitude, longitude) system for scientific target and team coordination. Possible Fine Sun Sensor or Star-Tracker use to determine latitude and longitude

For further details see [5]

CONCLUSIONS

The ANW of CNES has been successfully ported from a PC-LINUX environment to a flight representative LEON2-RTEMS environment. Many issues, limitations and characteristics of the OBSW/NAV-BB-SW and relevant hardware have been studied, reported and documented. Valuable experience has been gained. The work has been used later by Astrium-UK in the production of the integrated Mars Rover Prototype.

REFERENCES

[1] CNES Autonomous Navigation Basic Description and Preliminary Requirements Issue 1 Rev. 2
[2] Selecting and Combining Navigation and Locomotion “Breadboards” to complete an Integrated Mars Rover Prototype, IAC-08-A3.3.A8 Boyes, Soper, Patel, Clemmet, Renouf. Astrium 2008
[3] “EDRES General.doc” document, CNES September 2007
[4] ExoMars Rover Navigation Software Breadboard, Navigation SW BB test Report (P2), EXM-RM-TPR-TCP-002 V3
[5] ExoMars Rover Navigation Software Breadboard, Assessment of Areas for Improvement. EXM-RM-TN-TCP-004