AUTONOMOUS AND/OR INTERACTIVE CONSTRAINTS-BASED SOFTWARE RECONFIGURATION FOR PLANETARY ROVER

Giuseppe Montano[(1)], John McDermid[(1)]

[(1)]The University of York, Heslington, YO10 5DD, York, United Kingdom, Email:{napo, jam}@cs.york.ac.uk

ABSTRACT

Following our experience in the development of ExOS, the RTAI/Linux-based real-time operating system for the planetary rover ExoMaDeR, we realized that the management of onboard software configuration requires more powerful capabilities than those currently available, especially to react autonomously and in real-time to unexpected events, e.g. faults. Autonomous dynamic software reconfiguration is a powerful fault management technique used in real-time safety-critical systems (e.g. military aircraft) to mitigate the effects of unexpected events.
In this paper we propose a novel autonomous software reconfiguration system for the planetary rover based on the Constraint Programming paradigm. Our approach is able to dynamically merge heterogeneous information that is relevant for this critical but powerful operation. Key parameters are faults, rover operating mode, mission objectives, redundancy requirements and safety requirements. We also extend our system framework to interactive scenarios, where a human operator expresses preferences on the reconfiguration and fault management processes. This feature is particularly useful during rover teleoperation. In order to produce situational awareness for the operator, our system is able to automatically generate explanations of autonomous reconfiguration and fault containment actions in real-time.
Our research is addressing ways of making the explanations readily understood by the operator.

1. INTRODUCTION

Rover autonomy is achieved by a combination of several qualities that make the system independent from human control. Fault tolerance is one of these qualities, whose role is to guarantee the survival of the system, even if in a degraded operating mode, rather then failing completely when some components of the system are subject to a fault.
Generally speaking, a fault management procedure is made up of three consecutive phases: Fault Detection (FD), Fault Identification (FI) and Fault Containment (FC). FD and FI are enabled by deterministic and accessible information, e.g. type of fault, specifics of the hardware components, fault propagation. FC is much more complicated since it requires a higher degree of knowledge, usually of a heterogeneous nature, which is not always easy to gather and elaborate during system operation, e.g. mission objectives and faults impact on them, strategies of fault masking and their effects on the system behaviour, optimal recovery paths, operators' preferences (in case of remotely-controlled rovers), etc.
The problem with representing operational knowledge is exacerbated by the unpredictability of a planetary rover operating scenarios. In fact, for unstructured environments, it is unrealistic for engineers to foresee all the possible *combinations* of unexpected events, operating conditions, rover operating mode, environmental factors, mission objectives, etc, and devise an optimal recovery strategy for each combination of factors – the number of permutations would be too great.
The ECSS Q-30B standard [1] states:"*Contingency analysis shall be performed to identify all contingencies arising from failures of the system. The analysis shall identify the means to prevent, contain and limit each contingency, and detect and diagnose it to recover the system to a nominal or degraded state. […]A fault tree analysis (FTA) shall be used to verify that the design conforms to the failure tolerance requirements for combinations of failures.*"
The kind of offline analysis that the standard prescribes is extremely difficult to achieve in the light of the arguments presented before. In order to augment rover autonomy, a dynamic fault management approach is essential, capable of *adapting to changing conditions* and devise an ad-hoc containment strategy *in real-time.*
In Section 2 we show how the problem of dynamic FC is successfully tackled in modern aircraft by means of dynamic system reconfiguration. Section 3 shows how aircraft dynamic reconfiguration techniques can be ported with good effect to planetary rovers in order to augment their autonomy. We present the framework of our autonomous dynamic reconfiguration system for planetary rovers in Section 4 together with a system implementation for the ESA ExoMaDeR rover. In Section 5 we extend our autonomous approach to remotely-controlled rovers, showing how our reconfiguration system can support the operator by automatically generating explanations of its actions in order to augment her situation awareness. Finally, we present conclusions and future directions of our research work in Section 6.

2. FAULT CONTAINMENT AND SOFTWARE RECONFIGURATION FOR MODERN AIRCRAFT

In recent years aviation systems have been developed as *federated* systems, with each major function, or application, in a separate hardware unit (Fig. 1a). As a consequence, aviation applications are physically separated from one another, each module is designed for the specific function it must deliver, the communication among the modules is usually one-to-one and its specifics are usually agreed with the suppliers of the communicating modules.
The aviation industry is moving towards a new approach for the development of avionic systems: *Integrated Modular Avionics (IMA)*. IMA, in brief, is a term used to describe an airborne real-time computer network consisting of a number of computing modules (Line Replaceable Modules - LRM) capable of supporting numerous applications of differing criticality levels (Fig. 1b). Every application can run on every LRM and all the airborne devices are connected to a shared data-bus (e.g. ARINC-664/AFDX). There are several standards dealing with IMA architectures, mainly distinguished by their application field, i.e. military or civil. We refer to the IMA architecture presented in the ARINC 653-1 standard [3].
The IMA concept is already applied, with different degrees of adherence to the standards, by the biggest aircraft manufacturers in their latest products, e.g. the Common Core System (CCS) by Boeing on the B787, the OpenIMA by Airbus on the A380, the Modular Data Processing Unit (MDPU) by Thales on the Rafale (more details are given in [2]).
The degree of modularity and flexibility of the architecture enables advantage to be taken of the possibility to *reconfigure* the avionics of the aircraft to adapt the system functions to changing conditions, which can be planned (e.g. mode-change, alteration of mission objectives - in case of a military aircraft) or unplanned (e.g. faults, unexpected critical scenarios).

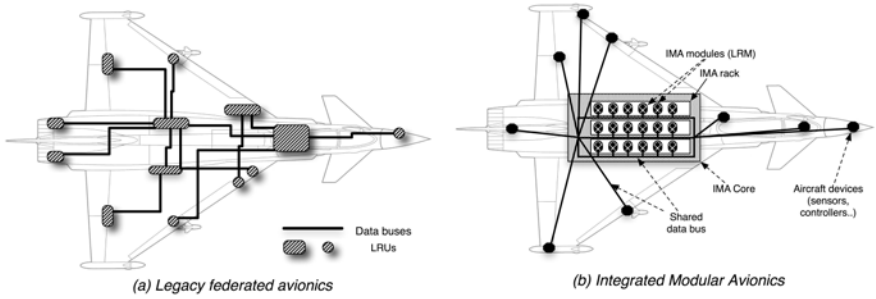


Fig. 1. Legacy federated avionics and IMA

Jolliffe and Nicholson [4] distinguish among: i) manual reconfiguration (MR), ii) static automated reconfiguration (SAR) and iii) dynamic reconfiguration (DR).
In MR, configurations (allocations of software modules to available computing resources) are manually generated offline; they are usually stored in lookup tables and accessed by the Configuration Manager (CM) when a reconfiguration "command" is issued. SAR becomes inevitable when the number of configurations increases to the extent that they cannot be treated manually, so a certain degree of automation is required. Configurations are defined automatically, but still on the ground. Because of the complexity of modern aircraft, the number of permutations of faults, operating modes, operating scenarios, architectural constraints, etc, is vast. Thus it is impossible for engineers to define configurations able to cope with all the possible critical situations.
DR is the response to this problem but it is also the most challenging type of IMA reconfiguration since it must happen whilst the system is operational (the aircraft is airborne). Configurations are generated and applied in real-time as a response to planned or unplanned events.
We show here the potential of IMA reconfiguration as a means to provide fault containment through two hypothetical, explanatory case scenarios.

Scenario 1: *the aircraft is in Battle Mode and there is a fault in the power generator* p_3. p_3 *powers* LRM_7 *on which Missile Control Service (MCS) is running.*
As a result of the fault, only a subset of all the LRM can still be powered and some applications must necessarily be shut off. Reconfiguration allows "moving" software from LRM_7 to another computing module switching off applications that are currently less critical than MCS, e.g. Gear Control Service.
Scenario 2: *because of fire onboard an A320, two LRMs are lost along with some sensors, including the main air speed sensor. The pilot is going to attempt an emergency landing so the real-time data concerning the speed is critical. The A320 is protected from low-speed stall (the flight control systems commands the engines to increase thrust without pilot intervention) which can be particularly helpful during an emergency landing. The air speed data can actually be retrieved from other sensors installed on the engines but additional controlling software needs to be loaded.*

In this scenario, the loss of two LRMs means that the system cannot run all the applications required by the current aircraft mode, so some of them must be shut off. A reconfiguration would allow shutting off non-critical applications and loading the drivers of the emergency speed sensors to help the pilot attempting the emergency landing.

Reconfiguration is a powerful FC means but, as we show in [5], it is a complex combinatorial problem going far beyond "simple" scheduling and resource allocation issues. A problem solution (an avionics configuration) is found by processing information of a heterogeneous nature, e.g. task scheduling and resources allocation constraints, fault impact, hardware/software characteristics, aircraft operating mode, operating scenario, pilot's preferences, etc. Next we explore these issues in the context of the planetary rover.

3. DYNAMIC SOFTWARE RECONFIGURATION FOR PLANETARY ROVER

As with aircraft, the capability of the rover to reconfigure dynamically in order to contain the effects of unexpected events could be a powerful "enabler" for autonomy. We present a simple scenario in order to show the potential of this capability and make a comparison with the aircraft case.

Scenario 3: *a planetary rover is equipped with an onboard computer made up of, amongst other components, two LEON processors, an FPGA co-processsor, volatile and non-volatile memory (this architecture is inspired to the ExoMars onboard computer presented in [6]). The rover is carrying out a scientific experiment on Mars. Suddenly a severe electrical breakdown causes the loss of one battery, some banks of volatile memory and one the FPGA coprocessors. The applications running on the FPGA, responsible for wheel motor control, cannot execute anymore and the reduction of central memory makes it impossible to run all the scientific applications (in execution on the LEON CPUs) required for the experiment. Furthermore the experiment site is located in the bottom of a ravine and this position prevents the rover from sending data on Earth to receive recovery/maintenance instructions from the engineers.*

The intuitive consequence of Scenario 3 is mission loss: the rover is stuck in an experiment site, has no control over its wheels, there is no possibility to receive data from Earth, the energy available and the computer status prevents from executing all the applications required to finish the experiment and to move to a better location.

Dynamic reconfiguration could be a possible solution to avoid total mission loss. In fact, what is really important for a planetary rover is the capability of the computer to guarantee that only those functions that are *critical* in the *current* operating conditions are available. We identify this quality as system *functional survivability*.

Hypothetically, in Scenario 3, the system should: *a)* identify the faults; *b)* set new constraints on the computing resources that are actually available in order to circumvent the affected components; *c)* shutoff all the non critical applications and devices, and try to reconfigure the software in order to fit the functions which are required to move the rover away from the hostile location using the reduced set of resources available. Point *c* should be performed by processing information about the current operating environment, the mission objectives and the safety requirements.

Besides the literature in the aviation domain already mentioned, several studies on Failure, Detection, Isolation and Recovery (FDIR) for space systems (e.g. [7]) have highlighted that the knowledge required to define the recovery phase following a fault is one of the biggest challenges. Specifically, [7] reports: "*The main limit, as for each former aspect, stays in the necessity of a good knowledge of the system the FDIR has to operate on. Both a physical and a functional knowledge is needed in order to correctly settle the states and attributes involved in each high level activity and to define the global transition matrix, fundamental for the consistent path definition*".

Back to Scenario 3, like with avionics, points *a* and *b* of the recovery strategy previously mentioned can be implemented with the current state of technology but the organization and the utilization of the information required to solve *autonomously* point *c* constitutes the real research challenge.

Information Required for Effective Dynamic Reconfiguration

The following types of information (or constraints) need to be handled simultaneously to solve autonomously the combinatorial problem associated with rover dynamic reconfiguration:

- *Task scheduling:* all the applications must meet their execution deadlines to guarantee coherent real-time execution;
- *Hardware resources:* the active set of application should be chosen in respect of the set of available hardware resources, e.g. memory, CPU, bus bandwidth, functioning sensors and devices;
- *Operating mode:* for each operating mode, a set of running application is strictly necessary, whilst others are less important. For example, in Cruising Mode, steering control is mandatory but it is not mandatory during the execution of a scientific experiment that doesn't require any movement of the rover. Furthermore, the degrees of *allowed degradation* of each operating mode should be defined and this information should drive the

reconfiguration process, e.g. a specific reconfiguration action must be issued under certain degrees of degradation.

- *Mission objectives:* like with operating modes, each mission objective has its set of mandatory applications, e.g. each experiment can be carried out only if its supporting applications are executed.
- *Faults:* each fault has a specific impact on the system and one or more paths to recovery are associated with it. Apart from the set of applications required to accomplish the recovery operation, some issues concerning the transition process between two configurations after a fault must be considered. First of all, the transition is not always instantaneous but sometimes, depending on the nature of the fault, it must go through several mandatory intermediate stages, e.g. after a fault to a non-volatile memory block, some procedures that provide for isolating the faulty block need to be executed before proceeding to the application of a new software configuration. Furthermore, depending on the nature of the fault, not every configuration is "reachable" from the current system state, e.g. in a bi-processor system, after a fault to one of the CPU, any configuration that relies on processor redundancy (either as part of the end state or in transition) cannot be reached.
- *Dependability requirements:* dependability is critical for a planetary rover. This quality is made up of several system attributes, amongst which availability, reliability, safety, integrity and survivability. All these attributes must weigh on the definition of the characteristics of the next configuration. For example: configurations C_i and C_j are two candidates to be applied after a fault; C_i offers double bus redundancy and C_j offers no bus redundancy but provides some functions that are advisable but not strictly required by the operating scenario; C_i is a better candidate since redundancy increases most of the dependability attributes previously mentioned.

Organizing and harmonizing such heterogeneous information in a way that it can be efficiently treated by the system in real-time to devise suitable system configurations is one of the main issues to be addressed in developing a rover reconfiguration system. We evaluated several approaches to Multi-Objective Optimization (e.g. aggregating functions, utility theory, fuzzy logic, goal programming, lexicographic approaches, simulated annealing, tabù search, genetic algorithms) but all of them share a weakness in the inability to express and manage highly heterogeneous data.
In the next Section we put forward an approach based on Constraint Programming (CP) that solves the issue. Before that, we need to define the reconfiguration problem in more details.

The Rover Dynamic Reconfiguration problem

The strong similarity of the dynamic reconfiguration problem for avionics and planetary rovers allows generalisation of some definitions that we put forward in [5] to the rover case.
Definition 1. *A System Behavioural Requirements Specification (SBRS) is a complete description of the behaviour of the system. The requirements are of heterogeneous nature, e.g. they cover safety, redundancy, performance, human-system interaction (in the case of remotely-controlled systems), degraded modes of operation. The system should have multiple SBRS to cope with different operating scenarios.*
Definition 2. *A System Configuration (SC) is an allocation of software functions onto available hardware resources, satisfying a specific SBRS.*
Definition 3. *A System Dynamic Reconfiguration (SDR) is the process through which the system, at runtime, halts operation under its current source SC and begins operation under a different target SC as a consequence of a SBRS variation.*
The problem of dynamic reconfiguration is usually associated with an application relocation problem (scheduling and resource allocation are the main issues). As we show in Fig. 2, a change in the software allocation (Configuration Change – CC) is only the result of a "higher level" Requirements Change (RC). An effective CC can be applied only with a good understanding of the RC, and this requires mastery of all the information described in the previous Sub-section.

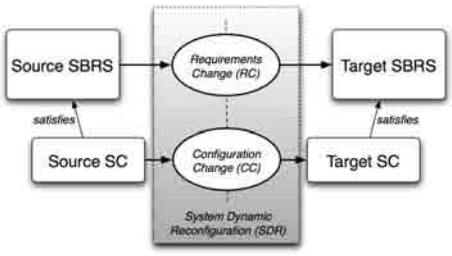


Fig. 2. The Rover Dynamic Reconfiguration problem

4. A SDR SYSTEM BASED ON CONSTRAINT PROGRAMMING

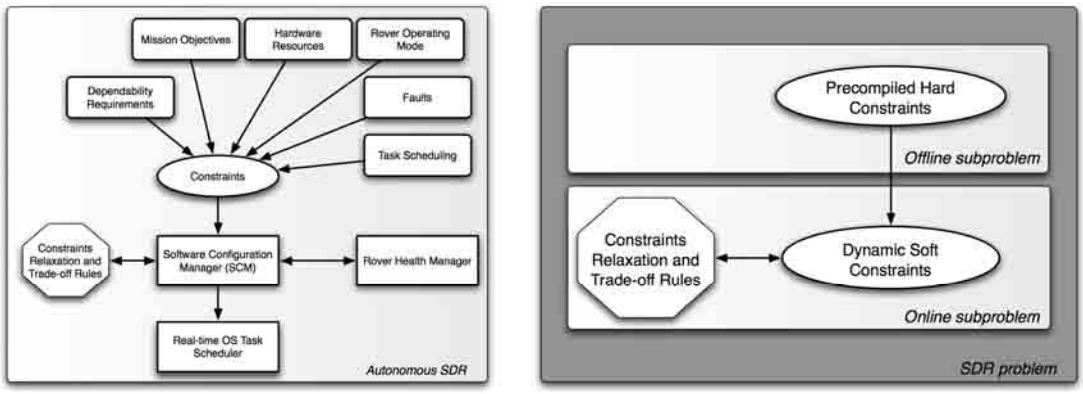
The Rationale for an Approach Based on Constraint Programming

The biggest issue that we found with most of the Multi-Objective Optimization approaches we evaluated is the inability to treat large amount of heterogeneous information. The flexibility of expression in Constraint Programming (CP) makes this paradigm particularly suitable for treating the SDR problem. In more details, some very good aspects of CP for combinatorial problems like SDR are [8]:

- *Declarativity*: variables, domains and constraints are simply described;
- *Genericity*: CP is not a problem-dependent technique, general rules are mechanically performed during the search;
- *Adaptability*: each constraint can be considered as independent and a model could be simply extended by merging different constraints;
- *Non-parametric ability*: no sensitivity to initial parameters (cf. temperature and cooling for SA), no training, easiness of extensions of the method to new models;
- *Completeness*: if there is no solution, the CP algorithm is able to prove it. This attribute is a valuable asset for dependable systems.
- *Utilization*: CP has been effectively used for a large range of combinatorial problems;
- *Performance*: CP has proved to perform just like (or even better than) SA in real- time scheduling and resource allocation [8], which is a fundamental part of the SDR problem.

The SDR System Framework

In Fig. 3a we present the framework of our SDR system for the planetary rover. The Software Configuration Manager (SCM) is responsible for the overall reconfiguration process. It collects information on the current operating scenario, translates it to constraints and uses them to devise new effective configurations. The SCM must work in tight collaboration with the system unit(s) responsible for fault detection and identification, i.e. usually the System Health Manager (SHM). The outcome of the SHM is critical to devise an effective recovery strategy since its action is propaedeutic to the fault containment phase. Here we will not analyse the FD and FI phases but a comprehensive study of the need for a tight collaboration between the SCM and the SHM in reconfigurable control systems, and how it should be structured, can be found in [8].



(a) The autonomous SDR system framework (b) The SDR problem

Fig. 3. Autonomous CP-based rover dynamic reconfiguration system

As SDR is usually applied in adverse operating conditions, the optimization objectives to satisfy at the same time are usually conflicting, e.g. dependability requirements and mission objectives. Given this motivation, there will not always be an optimal solution to the complex combinatorial problem associated with SDR and *trade-off decisions* must be made.

However the relaxation cannot be applied to any constraint independently: relaxing constraints like max memory available and bus bandwidth could result in the impossibility of running the onboard software. In order to solve this problem, we use an approach based on Interactive Constraint Satisfaction Problems (iCSP) theory. The overall problem is divided into two sub-problems (Fig. 3b): hard constraints (those that cannot be relaxed) are precompiled offline, before system operation, and the resulting solution space is stored in non-volatile memory using some form of symbolic representation (e.g. Binary Decision Diagrams). Soft constraints are applied on top of the precompiled solution space. Soft constraints can be added and retracted dynamically and the solution space is coherently modified in real-time.

Besides enabling trade-off reconfiguration decisions through soft constraints, the iCSP-based approach allows to improve the performance of the SDR process because part of the computation is done offline and only those decisions that required online knowledge are left for real-time elaboration. We made some preliminary tests using Binary Decision Diagrams as a symbolic representation to solve the reconfiguration problem on avionics systems and we obtained satisfactory results [5].

System Implementation

We implemented the SCM as a software module for the kernel of ExOS [10], the RTAI-Linux based real-time operating system for ESA ExoMaDeR rover (Fig. 4).

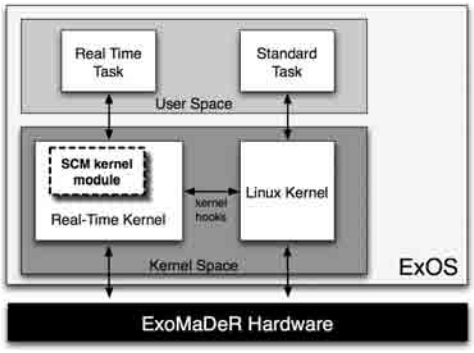


Fig. 4. SCM software architecture

The choice of implementing the SCM as a *kernel module* is mainly motivated by the following reasons:

- the SCM can directly access information concerning application deadlines, periods, computational time, etc. from within the kernel space, with no need to issue system calls and transfer information to the real-time user space, where all other scientific applications run. This is critical to verify the schedulability of each software configuration before its application;
- it can pre-empt the execution of other real-time and standard tasks. This guarantees low latency of the software when reacting to system events that issue a reconfiguration;
- the SCM can communicate directly with the rover hardware and retrieve efficiently low level information concerning faults, devices status, etc.

The SCM generates new system configurations using a model of the system (onboard computer and devices) based on the Constraint Satisfaction Problem (CSP) paradigm. A CSP is defined by a triple $R=(X,D,C)$ where:

- X is a finite set of variables;
- D is the domain of the variables in X;
- C is a set of constraints on the variables in X.

In line with our previous implementations for avionics [5], we modelled the rover SDR problem in CSP in the following way:

- hardware resources (e.g. buses, CPU, FPGA), software features (e.g. tasks computational time, memory required, number of messages), faults, etc, are modelled as variables of X;
- the domain D represents unary constraints on the variables (e.g. task deadlines, maximum memory available, bus bandwidth);
- C contains generic n-ary constraints, e.g. services required by each rover operating mode, execution dependencies among applications, dependability constraints.

We chose Explanation-based Constraint Programming (eCP) [11] techniques to solve the CSP associated with the SDR problem. The eCP-based SCM has been specifically developed to operate on the ExoMars onboard computer architecture presented in [6]. We used the Choco [11] eCP solver to implement the CP solution engine.

We simulated 25 different types of hardware fault and dynamically reconfigured the system more than 100 times, obtaining a Worse Case Execution Time of roughly 7 milliseconds on a computer equipped with a 2.16 Ghz Intel Dual Core processor and 1 Gb RAM.

These results demonstrate the efficiency of our method and give proof of applicability to real-time systems although a more extensive evaluation would be needed before the system could be deployed.

5. HUMAN CONTROLLER'S INVOLVEMENT

So far, the potential of our reconfiguration system has been presented for autonomous rover control. However, the utilisation of remotely-controlled rovers in future planet exploration missions is perfectly realistic, e.g. rover controlled by humans from the inside of settlements on the target planet.

Dynamic reconfiguration is a highly invasive operation, with significant impact on the behaviour of the "tele-operated" rover. In the case of remotely-controlled operation, excessive authority and automation of the SCM could have an unexpected and negative impact on the mission. Consider the following hypothetical scenario:

Scenario 4*: an engineer is remotely controlling a rover on Mars providing life support functions. A mistake could result in serious injuries to astronauts. Suddenly an electrical fault happens and the autonomous SCM, in order to protect the rover from further severe damage, decides to shutdown the rover and wait for external intervention. The time needed by the human controller to understand the reason for the rover shutdown, restart the system and go back to the safety critical task could be determining for the survival of the astronauts.*

The decision of the SCM is perfectly reasonable from a self-protection standpoint. However, because of the criticality of the operation, the human controller could decide to ignore the onboard electrical fault and, even risking the loss of the rover, try to save the astronauts' lives. In certain unpredictable environments, arbitrary human decisions, made on information which is impossible to fully codify in a computer, can possibly avoid mission failures or severe accidents.

In this regard, our SCM allows interaction with the human controller during the reconfiguration process. We accomplish this by translating the operator's preferences into soft constraints (as such, they can be issued and retracted during rover operation) as Fig. 5 shows.

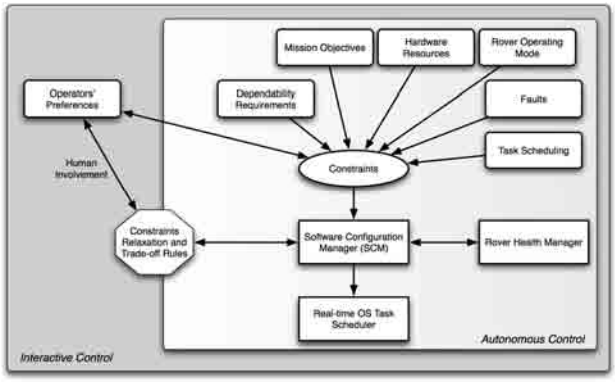


Fig. 5. Interactive CP-based rover dynamic reconfiguration system

Through a system of weights we give priorities to reconfiguration constraints, so that their influence on the reasoning of the system can be controlled and human preferences can be accommodated. The constraints that represent the controller's preference have higher priority so that the operator can *override* automated decisions. A characterisation from a cognitive standpoint of the human involvement during dynamic system reconfiguration can be found in [5].

Besides the possibility of accommodating operator preferences, the importance of intelligent systems having the ability to explain their reasoning has been recognized by several studies of cognitive science (e.g. [12]) as a means to improve situational awareness in the human controller. Explanations *(why?)* of system action and their implications *(what if..?)* can be very helpful for remotely-controlled rover operators in critical situations. In fact, in an interactive decision-making process, the quality of explanations affects: transparency, scrutability, trust, effectiveness, persuasiveness, efficiency and satisfaction [12]. For rover interactive SDR (iSDR), *effectiveness* (making good choices) and *efficiency* (making decisions quickly) are the most important aspects.

We chose an eCP-based approach to solve the SDR problem because, beside the motivations already given in the previous Section, eCP allows to automatically generate explanations of system actions during the solution of the problem. In eCP the search for solutions is guided by explanation-sets, i.e. *sets of constraints that justifies a domain reduction*. Translated into rover reconfiguration terms, an explanation of a reconfiguration action is information that explains *why* the system is suggesting a certain action, e.g. why, after a fault, the rover is not able to continue the current experiment anymore.

Explanation-sets are used by the eCP solver to efficiently drive the search for configurations but they use a low level representation of the problem, so they are not easily understandable by the operator at present. A translation to a higher level of representation is required. In [13] a method to accomplish this task is presented. This is one of our current research interests, together with the automatic generation of *implications* of reconfiguration actions.

Apart from the application to rover control, our approach to support the operator by automatically generating explanations of system actions during fault management processes can be generalized and applied to many reconfigurable manned space systems.

6. CONCLUSIONS

The ECSS Q-30B standard for space product assurance prescribes that contingency analysis "shall identify the means to prevent, contain and limit each contingency, and detect and diagnose it to recover the system to a nominal or degraded state". Unfortunately, because of the huge number of permutations of unexpected events that can happen onboard a rover and the unpredictability of the operating scenario, this kind of analysis is particularly difficult (if not impossible) to perform offline.

In this paper, we introduce SDR as an effective countermeasure to this problem. Our approach is inspired by fault containment and reconfiguration techniques for modern aircraft. Through SDR, the effects of unexpected events (e.g. faults) can be mitigated in real-time, guaranteeing the survival of the system in adverse situations.

In order to guarantee high performance for real-time execution, our approach divides the SDR problem into two components: an offline part and an online part. The results of offline contingency analysis are translated in constraints and codified in an efficient form of symbolic representation (e.g. Binary Decision Diagrams). Online analysis generates dynamic constraints that are applied on top of the solution space produced by offline analysis. New configurations are devised by a solution engine based on the eCP paradigm.

We presented an implementation of an explanatory SDR system for the ExoMaDeR planetary rover. The software is able to efficiently generate new system configurations in real-time, merging information of a heterogeneous nature, i.e. task scheduling, hardware resources, rover operating mode, mission objectives, faults and dependability requirements.

We extended our system with support for interaction with an operator, in case of remotely-controller rovers. Besides allowing the operator to override system decisions and express preferences on the characteristics of the next configuration, the system can automatically generate low level explanations of its actions in real-time in order to augment the situational awareness of the human operator. Our current research interests are making explanations readily understood, generating implications of system actions and extending the applicability of the approach to manned space systems in general.

7. REFERENCES

[1] European Cooperation for Space Standardization, "ECSS-Q-30B Space Product Assurance – Dependability", ECSS Secretariat, ESA-ESTEC, Requirements and Standards Division. Noordwijk, The Netherlands. 8 March 2002.

[2] G. Montano, "Reconfiguration Management for Integrated Modular Avionics", PhD Qualifying Dissertation, Department of Computer Science, The University of York, York, UK. September 2007.

[3] Airlines Electronic Engineering Committee, "ARINC Specification 653-1", Aeronautical Radio Inc., 2551 Riva Road, Annapolis, Maryland 21401. 2003.

[4] G. Jolliffe, M. Nicholson, "Exploring the Possibilities Towards a Preliminary Safety Case for IMA Blueprints". MSc Project, Department of Computer Science, University of York. 2004

[5] G. Montano, J. McDermid, "Human Involvement in Dynamic Reconfiguration of Integrated Modular Avionics", 27th Digital Avionics Systems Conference (DASC 2008). St.Paul - Minnesota (USA). 26-30 October 2008

[6] M. Roe, "ExoMars Rover Vehicle", European Industry Day, TAS-I, Turin, 29 May 2008 (http://esamultimedia.esa.int/docs/exomars/TurinMay2008/IndustryDayRV_euro_3.pps)

[7] Guidotto A., Martelli A., Peccagnini C., Lavagna M., "Smart-FDIR: Use of Artificial Intelligence in the Implementation of a Satellite FDIR", Proc. of DASIA 2003, Prague, Czech Republic. p. 71.1. 2-6 June 2003

[8] P.E. Hladik, H. Cambazard, A.M. Déplanche, , N. Jussien, "Solving a real-time allocation problem with constraint programming", The Journal of Systems & Software, Vol.81, n.1, Elsevier, pp. 132-149. 2008

[9] M. Nicholson, "Health Monitoring for Reconfigurable Integrated Control Systems", Proceedings Of The Thirteenth Safety-Critical Systems Symposium, Southhampton, UK, 8-10 February 2005, 2005.

[10] G. Montano, "A Linux-based Real-Time Operating System for ExoMaDeR", European Space Agency, ESA/ESTEC, Noordwijk, The Netherlands. January 2006.

[11] N. Jussien, "The versatility of using explanations in constraint programming", Reseach report, École des Mines de Nantes, Nantes, France. 2003.

[12] N. Tintarev, J. Masthoff, "Survey of explanations in recommender systems", 23rd IEEE International Conference on Data Engineering, pp.801-810. Istanbul, Turkey. 2007.

[13] N. Jussien, S. Ouis, "User-friendly explanations for constraint programming", ICLP'01 11th Workshop on Logic Programming Environments (WLPE'01), Paphos, Cyprus. 2001.