ARPHA: AN INNOVATIVE ON-BOARD FDIR REASONING ENGINE FOR AUTONOMOUS SYSTEM

Andrea Guiotto[(1)], Luigi Portinale[(2)], Daniele Codetta-Raiteri[(3)], Yuri Yushtein[(4)]

[(1)]*Thales Alenia Space Italia, Strada Antica di Collegno 253, 10146 Torino, Italy, Andrea.guiotto@thalesaleniaspace.com*
[(2)]*Università del Piemonte Orientale "A. Avogadro", Via Michel 11, 15121 Alessandria, Italy, luigi.portinale@di.unipmn.it*
[(3)]*Università del Piemonte Orientale "A. Avogadro", Via Michel 11, 15121 Alessandria, Italy, dcr@di.unipmn.it*
[(4)]*ESA - European Space Agency, Keplerlaan 1, PO Box 299, 2200AG Noordwijk, The Netherlands, yuri.yushtein@esa.int*

ABSTRACT

In the frame of the European Space Agency (ESA) studies, Thales Alenia Space has carried out a research – VERIFIM - in collaboration with Universita' del Piemonte Orientale, implementing a software prototype called ARPHA for on-board diagnosis, prognosis and recovery. It is an innovative on-board FDIR (Failure Detection, Isolation and Recovery) reasoning engine for autonomous systems, based on the inference techniques that use Dynamic Probabilistic Graphical Models. It started in June 2010 and ended in December 2011.

1. INTRODUCTION

Currently employed FDIR operations are based on the design-time analysis of the faults and failure scenarios (e.g. FMEA, FTA) and run-time observation of the system operational status (health monitoring). It has the main objectives to timely detect the faults and to initiate the corresponding predefined recovery actions. If no corresponding action could be found, FDIR proceeds by executing the actions to put the spacecraft into a known safe configuration and transfers control to the Ground operations for troubleshooting and planning the recovery. This approach could be not adequate because the reaction time needed does not always allow waiting a Ground recovery. Besides, traditional FDIR cannot provide and utilize prognosis for the imminent failures. Automated FDIR procedures cannot leverage specific course of recovery based on the on-board evaluation of the causal knowledge of the system and its environment status. It is impossible to estimate on-board the impact of the occurred faults and failures on the operational capabilities of the system.

2. THE ARPHA APPROACH

A new approach to on-board FDIR is needed, with the capability to reason about the anomalous observations, based on the global knowledge of the system and its capabilities, system environment, and system-environment interaction in the presence of uncertainty. It has to provide the system with prognosis on the operational status to be taken into account for autonomous operational planning, and to allow preventive recovery actions.

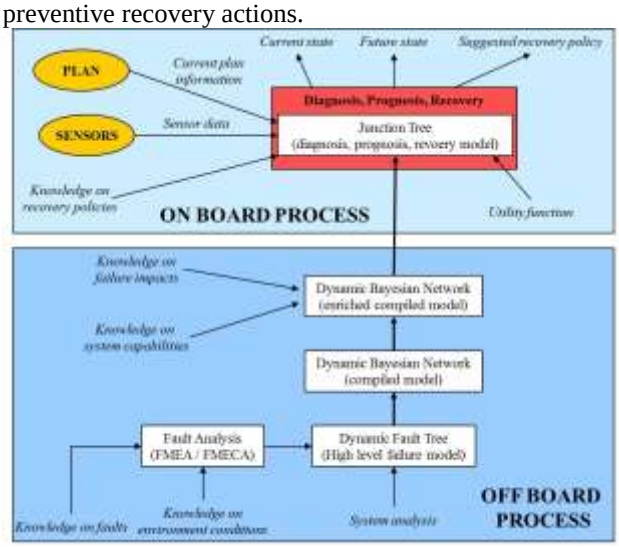


Figure 1. ARPHA on-board reasoning plus off-board process

The approach developed by Thales Alenia Space and Università del Piemonte Orientale provides a unified modeling and operational framework that integrates a high level modeling formalism (Dynamic Fault Tree - DFT [1]), a low level modeling formalism (Dynamic Bayesian Network - DBN [2]) and an inference oriented formalism (Junction Tree- JT [3]).
ARPHA prototype puts emphasis on the on-board software capabilities, but an off-line (off-board) processing phase is necessary, in order to provide ARPHA with the operational model (figure 1).
The DFT, constructed through the System/RAMS engineering, is automatically transformed to the DBN and then to the JT by an off-line (off-board) processing phase. The on-board analysis of the JT, conditioned by the sensor data and the recovery actions outcomes, allows evaluating the system current and future state, and choosing the recovery policies if necessary, in automatic way, without the assistance of the Ground

control. The recovery policy is selected on the base of a utility function that assigns a coefficient (utility value) to all combinations of utility variables (chosen between DBN variables).
Environmental aspects of the space mission can be modelled in the DBN used by ARPHA to perform inference. It is possible to take into account the failure causes by inserting them in the utility function computation used to select recovery. ARPHA evaluates failure impact on the currently executing plan. This approach increases the achievable level of autonomy.

2.1 Basic notions about DFT, DBN and JT

Fault Tree (FT) [4] is the most diffused and popular model in Reliability analysis. A FT is a direct acyclic graph (DAG) representing how several combinations of Basic Events (BE) lead to the occurrence of a particular event called Top Event (TE). Each BE (component failure) has a certain probability to occur according to its failure rate. So, it is possible to compute the probability of occurrence of the TE (system failure). In FT, BEs are assumed to be independent and the combinations of BEs leading to the TE can only be expressed by means of Boolean gates (AND, OR). Therefore the modeling power of FT is rather limited, so several extensions have been proposed in the literature, such as the Dynamic Fault Tree (DFT) [1]. DFT introduces dynamic gates representing dependencies: a dependency arises when the failure behavior of a component depends on the state of another component or subsystem. Dynamic gates represent several kinds of dependencies: functional dependencies, dependencies concerning the events order, and the presence of spare components.
Due to the presence of dependencies, the analysis technique for FT [4] are not suitable to analyze DFTs. Recently the analysis of DFT models has been faced by resorting to Dynamic Bayesian Networks (DBN) [2]. The way to convert a DFT into DBN is described in [5]. Besides the computation of the system unreliability, DBN allow computing predictive or diagnostic measures conditioned by observations about the system or components state during the mission time. The Radyban tool [6] for DFT analysis is based on the DBN approach. After the DFT conversion into DBN, it is possible to relax some constraints, such as the hypothesis that BEs are binary (working/failed). Furthermore, through the DBN, new knowledge and new dependencies can be included in the model.
Bayesian Networks (or Belief Networks - BN) [7] have become a widely used formalism for representing uncertain knowledge in probabilistic systems. BN are defined by a DAG in which discrete random variables are assigned to each node, together with the conditional dependence on the parent nodes. In particular, each node has associated a Conditional Probability Table (CPT) specifying the probability of each value of the node, conditioned by every instantiation of parent nodes. Root nodes are nodes with no parents, and marginal prior probabilities are assigned to them. In this way, it is possible to include local conditional dependencies, by directly specifying the causes that influence a given effect. This allows computing the probability distribution of any variable given the observation of the values of any subset of the other variables.
DBN extend BN by providing an explicit discrete temporal dimension. A DBN is essentially the replication of a BN over two time slices, $t-\Delta$ and t, where Δ is the time discretization step. In other words, each variable has two instances, one for each time slice. If a variable is characterized by a temporal evolution, its instance in the time slice t depends on its instance in $t-\Delta$. Moreover, it is possible to establish intra-slice dependencies involving different variables in the same time slice, or inter-slice dependencies involving different variables in different time slices. Given a set of observations up to the current time t, it is possible to compute the probability distribution of a variable at t, in the future, or in the past.
A way to efficiently compute conditioned probabilities on a BN or DBN, consists of generating and analyzing the Junction Tree (or Join Tree – JT [3]) according to the procedures detailed in [8]. A JT is an undirected unrooted tree where each node corresponds to a set of nodes in the original BN or DBN (also called a *cluster*).

3. ARPHA IMPLEMENTATION

ARPHA is implemented as one periodic process. ARPHA will run in parallel to other processes of the on-board software (figure 2). In particular, the management of the policy event, triggered by recovery and managed by Event Handler of Event Action Service, is assumed to run concurrently to ARPHA. In this way, ARPHA does not have to wait the conclusion of the active recovery policy execution to perform a new diagnosis or prognosis on the system, but ARPHA can consider also the changes performed during execution of a recovery policy to perform a new diagnosis or recovery.

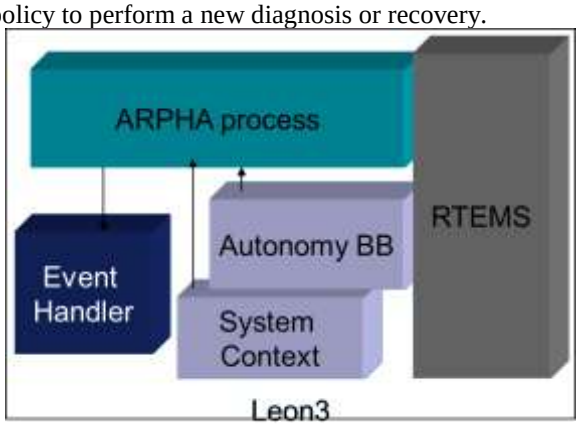


Figure 2. ARPHA process

A cycle of ARPHA starts with a diagnosis inference. In case the inference result is nominal state, ARPHA performs a prognosis inference; in case diagnosis result is anomaly or failure, ARPHA performs a reactive recovery inference. In case the prognosis inference result is not nominal, ARPHA performs a preventive recovery inference (see figure 3).

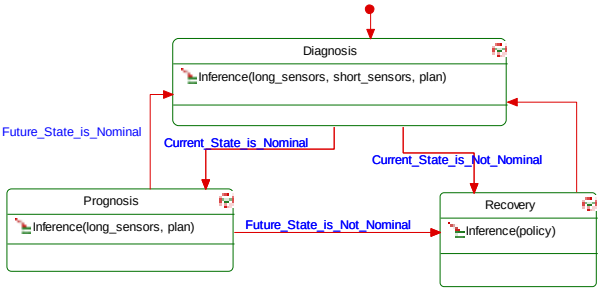


Figure 3. ARPHA cycle

4. THE ARPHA IMPLEMENTATION AND VERIFICATION

The developed ARPHA prototype has been evaluated on the space embedded target (running under RTEMS on the LEON3 processor) using as a case study the scenario and the simulation of a rover mission for planetary exploration. Several simulations have been performed in order to evaluate the suitability of the approach w.r.t.:

- The rover mission characteristics such as unpredictable local conditions (slope of terrain, presence of dust, type of soil)
- Damage to rover component such as battery
- Current state diagnosis
- Future state prognosis
- Selection of recovery
- Development methodology

ARPHA has also been run in order to characterize the approach with regard to the following parameters: reliability (ability to perform diagnosis and prognosis to select a recovery strategy in case of anomaly), adequacy (failure impact verification, autonomy requirements, critical system requirements), effectiveness (dependency on the component fault rate, precision of diagnosis, prognosis and recovery), availability (duration of the ARPHA cycle vs. the frequency of observations collection, schedulability analysis), processing power and memory requirements.

5.1 Description of case study

The selected space system to be used as a case study has been a rover for planetary mission. Because of the limited per day Ground contacts and of the communication delay mainly due to the Earth-Planet distance, on-board processes have to be designed and implemented to manage autonomously potential anomalies and threats.
In addition to the classical anomalies and failure sources that are normally considered on a satellite, the planetary rover is subject to a number of mission threats that are due to the close interaction of the rover with the Planet environment, that is, in some cases, impossible to predict by ground and requires an onboard monitoring and reaction capability.

A typical example of threat to the rover vehicle is when the rover is travelling on a slope. In this case, if the inclination of the rover is exceeding a given security threshold then an FDIR reaction is immediately needed to stop the locomotion system. When a threat is detected, this will trigger an alarm causing a reboot of the OBC, leading the system to Standby Mode. Other errors depending on slope of terrain are operational errors in the orientation of the solar panels, influencing the power generated by solar arrays. Moreover, power generated by solar arrays is also influenced by the presence of dust, while power generated by the battery can decrease as a consequence of a damage of the battery itself. Other threats can be finally related to drill and subsurface material.
In order to evaluate ARPHA with realistic mission threats, the following scenarios have been identified:

- Slope of terrain (S1): the presence of a terrain slope increases sun aspect angle by causing lower power generation of solar array
- Presence of dust (S2): the presence of dust increases optical depth and reduces power generated by solar arrays.
- Problem during drilling (S3): we simulate an unexpected high request of energy by drill.
- Damage to battery system (S4): we simulate a damage to battery that reduces battery charge level.

Hereafter the table reports description of recovery policies.

Policy ID	Description	Scenario
1	Transition to STAND_BY mode	S1
2	Bring solar array horizontally	S1
3	Move out from shadowed area	S2
4	Move SA panels to horizontal position Retract DRILL Transition to STAND_BY mode	S2 S3 S4
5	Retract Drill Transition to STAND_BY mode	S3

Identified scenarios can be represented by considering the electrical power subsystem of the rover, and the top event (TE) considered in DFT analysis is the unavailability of power to generic equipment when needed.

For each failure, there is one or two recovery policies to be selected. The recovery policy is selected on the base of a utility function (see table 1). In this case, the interest of utility function is to perform a sequence of recovery actions and to have a positive balance (generated power is greater than consumed power). Utility function assigns a coefficient to all combinations of actionID and balance. For example, the utility to perform any actions with negative balance is zero. Utility values different from zero are assigned to combination between recovery action and not negative balance.

	BALANCE		
ACTIONID	gen=cons	gen>cons	gen<cons
standby	0,2	1	0
drill	0	0	0
move	0	0	0
pancam	0	0	0
mast	0	0	0
wisdom	0	0	0
tilt	0,1	1	0
retract	0	1	0

Table 1. Example of Utility function

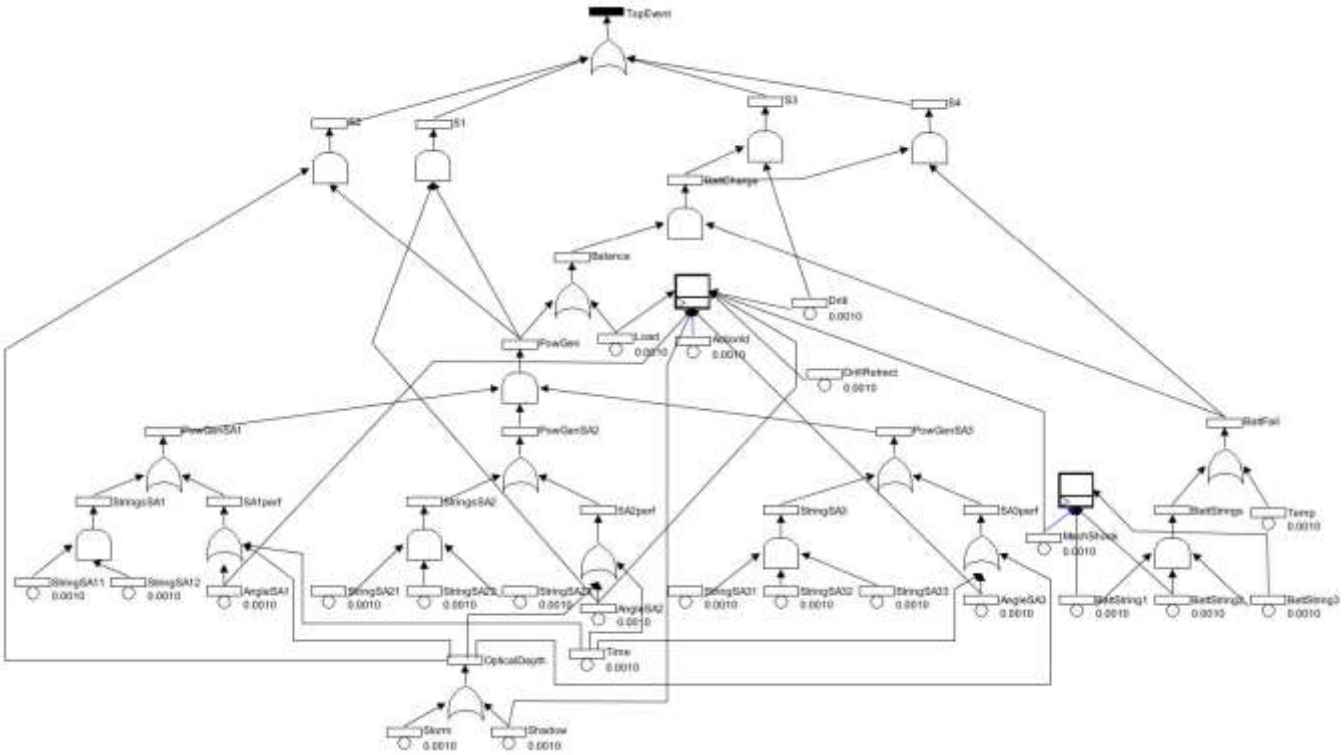


Figure 4. DFT model of the case study

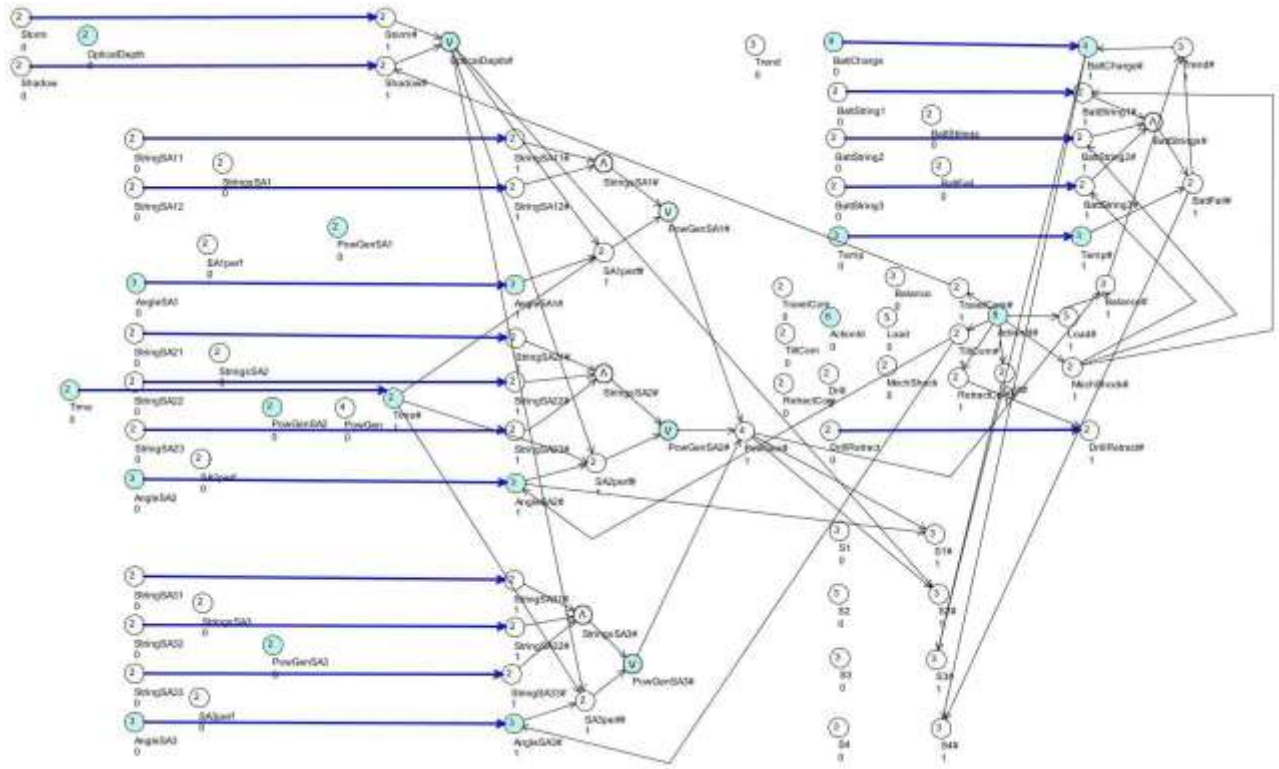


Figure 5. DBN model of the case study

The DFT model of the case study represents the combinations of events or states leading to TE corresponding to the anomaly or failure of the whole system (figure 4). TE is the output of an OR gate and occurs if the event S1, S2, S3, or S4 happens. The event S1 represents the scenario S1 and is the output of an AND gate. S1 occurs if both the events PowGen and AngleSA2 occur. They represent an anomaly/failure about the power generation (for instance, a low level of generated power) and a non optimal sun aspect angle for SA2 (we assume that the sun aspect angle of SA2 is similar to the angle of SA1 and SA3). PowGen occurs if all the events PowGenSA1, PowGenSA2 and PowGenSA3 happen. Each of them represents the fact that a solar array is not producing energy. For example, PowGenSA1 concerns SA1 and happens if StringsSA1 occurs (all the strings of SA1 are failed) or SA1perf occurs (the time, the optical depth and sun aspect angle of SA1 do not allow the generation of energy). The optical depth is not optimal in case of storm or shadow.

The event S2 occurs if both PowGen and OpticalDepth happen. S3 occurs if both BattCharge and Drill occur; they represent an anomaly/failure about the level of charge of the battery, and the drill actions in execution respectively. BattCharge in turns occurs if both the events Balance and BattFail happen. Balance represents the fact that the use of the battery is necessary: Balance happens if both PowGen and Load occur. The second event represents the presence of a load (consume of energy). The event BattFail models the damage of the battery because of the failure of all its strings (event BattStrings) or a low temperature (event Temp). Finally, S4 occurs if both the events BattCharge and BattFail happen.

The model contains two functional dependency (FDEP [1]) gates. The first one represents the influence of ActionId on other events, such as Load, Drill (in case of drilling actions), DrillRetract (drill in or out), AngleSA1, AngleSA2, AngleSA3 (in case of tilting actions), Shadow (in case of travelling actions), and MechShock (possibility of mechanical shock damaging the battery strings, in case of drilling or travelling actions). MechShock influences in turns the events BattString1, BattString2, BattString3 by means of the second FDEP gate.

The DBN of the case study reported in figure 5 has been derived from the faul-tree model by following two steps: 1) the DFT has been “translated” into the DBN. 2) Then, the DBN has been enriched by increasing the size (number of possible values) of several variables and expressing more complicated relations among the variables inside the Conditional Probability Tables (CPT) of the variables. For instance, the level of power generation, battery charge, or load needs to be represented with a variable with more than two values, if the model has to be enough accurate to capture the aspects of the system behaviour causing its state. For

this reason, the DBN resulting from the DFT conversion has been enriched in this sense:

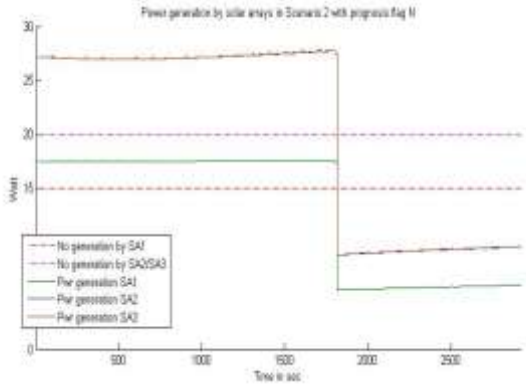
- The variables representing the sun aspect angle of each solar array and the variable Temp are ternary (good, discrete, bad).
- The size of PowGen and BattCharge is 4 (we can represent 4 intermediate levels of power generation and battery charge).
- The size of Load is 5 (5 levels of consume of energy).
- The size of ActionId is 8 in order to represent 8 actions of interest in the model.
- The variables S1, S2, S3, S4 are ternary in order to represent the states Normal, Anomalous and Failed in each scenario (the Normal state indicates that the scenario is not happening).

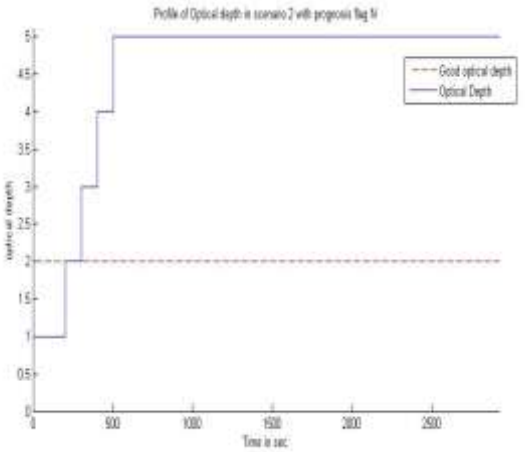
The structure of the DBN reflects the structure of the originating DFT: each event in the DFT corresponds to the variable in the DBN with the same name, while the DFT gates determine the influence arcs in the DBN. However, in the DBN we added some support variables in order to reduce the number of entries in the CPT of the non binary variables by applying the so-called "divorcing" technique [6]. The support variables are: TravelCom, DrillCom and RetractCom depending on ActionId, and Trend depending on Balance and BattFail.

In the DBN, each variable has two instances, one for each time slice (t, $t+\Delta$). If a variable has a temporal evolution, its two instances are connected by a "temporal" arc appearing in blue colour in figure 5. Still in figure 5, the observable variables are put in evidence (blue nodes); the values coming from the sensors will become observations for such variables during the ARPHA cycles and the inference analysis of the model.

5.2 Evaluation results

In order to perform an empirical evaluation of the approach, ARPHA has been deployed in an evaluation platform composed by a workstation linked to a PC via Ethernet cable. A rover simulator has been installed on the workstation. On the PC we installed the TSIM environment, emulating the on-board computing hardware/OS environment (LEON3/RTEMS). This paper reports results of scenario 2 simulation (presence of dust increases optical depth and reduces power generated by solar arrays).





Diagnosis at mission time 9 (663 sec) and 10 (728 sec) gives as result, nominal current state, while prognosis at mission time 9 and 10 gives as result, anomalous/failed future state (see table below). No preventive recovery is requested because prognosis flag is set to N that inhibits the recovery for prognosis.

***** Diagnosis ***** STATE SYSTEM **"N"**
Prognosis ## FUTURE STATE SYSTEM "F" (2)
Future System state anomalous/failied but prognosis flag set to 'N'
Elapsed Time Prognosis and or Recovery: 43.500000 sec

At mission time 27 (1835 sec) also diagnosis detects the failure of S2 by confirming the prognosis of the previous steps. Policy 3 and 4 are selected for reactive recovery. Policy 4 is saved as the best one (utility function=0.8773) and Policy 3 is discharged (utility function=0.09183467).

********** ROSEX VALUES **********
(**1835**) At step 27 read opticaldepht = 5.00000
(**1835**) At step 27 read pwrsa1 = 5.53303
(**1835**) At step 27 read pwrsa2 = 8.77638
(**1835**) At step 27 read pwrsa3 = 8.79704
****
***** Diagnosis *****

NO FAILURE Pr{S1#=2} = 0.00000000 (0.99000000)
(Criticality level 3) **Failure 2 save**
[Pr{S2#=2} = 1.00000000] >= 0.59000000

NO FAILURE Pr{S3#=2} = 0.00000000 (0.99000000)
NO FAILURE Pr{S4#=2} = 0.00000000 (0.99000000)
Anomaly 1 excluded because under recovery or minor criticaly (no check)
Anomaly 2 excluded because under recovery or minor criticaly (no check)
NO ANOMALY Pr{S3#=1} = 0.00000000 (0.99000000)
NO ANOMALY Pr{S4#=1} = 0.00000000 (0.99000000)
STATE SYSTEM "F" (2)
Elapsed Time for diagnosis: **8.630000 sec**
Reactive Recovery
Policy 3 save as best (0.09183467)
Policy 4:
Utility Function= 0.8773
Policy 4 save as best (previous 0.09183467)
Best policy for Reactive Recovery is the 4
BEST POLICY for the failure 2 is: 4
Failure 2 under recovery
Policy 4 running
Elapsed Time Prognosis and or Recovery: **174.320000 sec**

From mission time 0 to mission time 3 all the observations from sensors indicate a normal situation about the monitored parameters, and this determines the detection of the normal state of the system by both the diagnosis and the prognosis.
From mission time 4 to mission time 8, there is only one change in the observations. Such change concerns the optical depth which is not optimal any more. Since we still observe that the power generation is performed by all the solar arrays, S2 is not detected by the diagnosis.
The prognosis does not detect S2 in the future five steps because the plan does not contain in such steps any movement (GNC) action by the rover. This means that the position of the rover will not change, and as a consequence the situation about the optical depth and the power generation may be the same. The future probability of S2 is not high enough for the detection of S2 by the prognosis.
At mission times 9 and 10, the observations from the sensors are the same as before, so S2 is not detected by the diagnosis. However the plan contains now a future travel (GNC) action. Such action will change the rover position increasing the probability of the scenario S2 which is detected by the prognosis as a consequence.
From mission time 11 to mission time 26 we are in the same situation observed in the mission times from 4 to 8, about the sensors and the plan. Therefore we obtain again that the current and future state is normal.

The size of ARPHA application is 320 Kbytes. ARPHA performs inference within 32MB of RAM. Using case study model, an ARPHA cycle (diagnosis and prognosis) takes about 1 minute. A single inference cycle takes 9 seconds. In case of recovery the ARPHA cycle can take 4 minutes. The time spent for inference could appear as a weak point of the approach, but it is due to microprocessor performance used to run ARPHA. It is obvious that with a more powerful hardware also ARPHA performance will improve. The duration of ARPHA cycle should be comparable to frequency of sensors updating in system context, in case of the best performance by the processor. In any case, ARPHA has been developed as a potential Building Block to be reused in on-board software. ARPHA is robust to stack overflow. ARPHA does not use unbounded depth recursive functions. It does not use dynamic memory allocation.

5.3 Industrial prospective

Evaluation and characterization of ARPHA have demonstrated that ARPHA could be suitable to be used in the context of the current space applications and available on-board, but some gaps must be still filled. In particular, computing time for analysis could be reduced by improving the onboard computing power especially in the prognosis functions. It could be interesting to run ARPHA on a co-processor. Another aspect to be improved is related to modelling. The DFT formalism is rather simple, so the design of the DFT model does not require a modeller with particular skills in stochastic modelling. The DBN can be obtained in automatic way, but its enrichment actually requires a modeller with a specific experience in Bayesian modelling. In particular, the editing of CPTs needs a particular attention in order to consider any possible case and avoid cases not compatible with observations.
In order to avoid the manually enrichment of DBN, DFT formalism can be extended into the EDFT [9] formalism introducing the modeling mentioned above. If an automatic translator from EDFT to DBN was developed, the effort to enrich the DBN would be less relevant because several features may be directly modelled in EDFT form, and translated into DBN in automatic way.
In order to design an accurate stochastic model, knowledge about probability parameters (i.e. component failure rates to be associated with the basic events in the DFT model) has to be provided. Such values may not be immediately available; in this case, they must be estimated or investigated
At this point, another not negligible aspect is the link between computing time and model complexity. The time necessary to analyze the model is influenced by the model size. The complexity of the DBN model depends on the number of entries in the CPTs of variables. The size of CPT depends on the size of variables (number of possible values) and the number of parents of the variables. In this sense, a simple model may not take into account system features relevant to the system reliability, while a complex model may require long computing time on the on-board hardware. It is necessary to perform a trade-off between the model accuracy and the computing time.
The on-board use of ARPHA requires a dedicated software development process to be integrated with the on-board software development showed in figure 6.

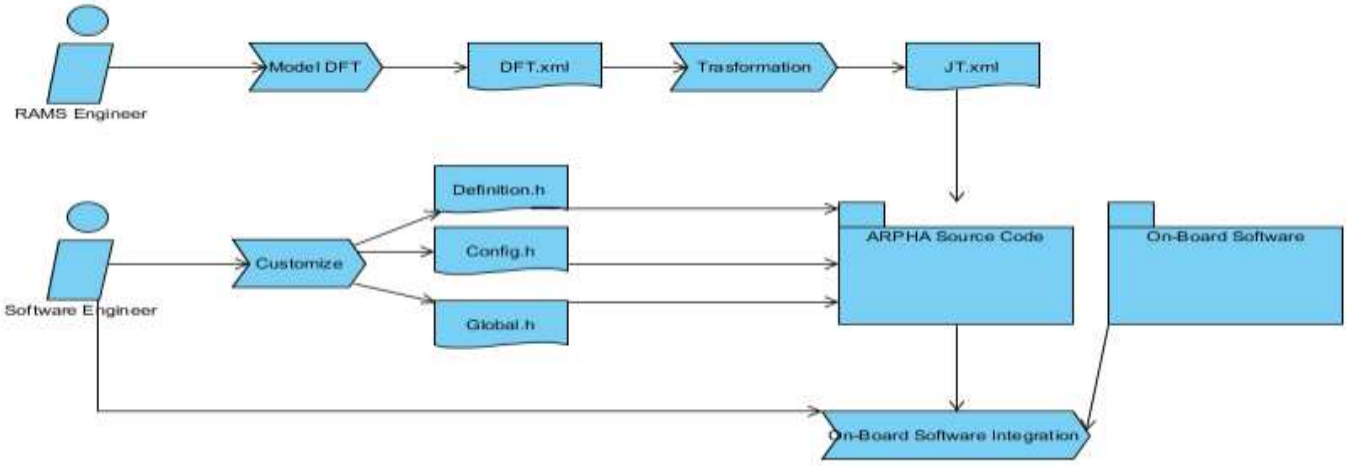


Figure 6. ARPHA software development process

5. CONCLUSIONS

The developed approach provides a unified modeling and autonomous framework that integrates an high level modeling formalism (DFT), a low level modeling formalism (DBN) and an inference oriented formalism (JT). The on-board analysis of the JT conditioned by the sensors data and the recovery actions, allows evaluating the system current and future state, and the recovery policies if necessary, in automatic way, without the assistance of the ground control. This approach increases the achievable level of autonomy. The developed prototype ARPHA represents an on-board software FDIR component suited for use in the existing spacecraft system architectures. It can perform on-board diagnosis, prognosis and recovery inference. ARPHA is able to verify the failure impact on the future state of the system.

Environmental aspects of space mission can be modeled in the DBN used by ARPHA to perform inference. It is possible to take in account the failure causes, by inserting them in the utility function used to select recovery. ARPHA can evaluate the failure impact on the currently executing plan as well.

The developed ARPHA prototype has been evaluated on the space embedded target (running under RTEMS on the LEON3 processor). The obtained performance data shows ARPHA usability in the context of the current space applications and available on-board computers.

6. REFERENCES

1. Bechta-Dugan, J., Bavuso, S.J. & Boyd M.A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability* **41**, 363-377.

2. Weber, P. & Jouffe, L. (2003). Reliability modelling with dynamic Bayesian networks. In Proc. 'Symposium on Fault Detection, Supervision and Safety of Technical Processes', Washington DC, USA.

3. Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkley.

4. Schneeweiss, W.G. (1999). *The Fault Tree Method*, LiLoLe Verlag.

5. Montani, S., Portinale, L. & Bobbio, A. (2005). Dynamic Bayesian networks for modeling advanced fault tree features in dependability analysis., In Proc. 'European Safety and Reliability Conference', Gdansk, Poland.

6. Portinale, L., Bobbio, A., Codetta-Raiteri, D. & Montani, S. (2007). Compiling dynamic fault trees into dynamic Bayesian nets for reliability analysis: The Radyban tool. CEUR Workshop Proceedings **268**.

7. Langseth, H., & Portinale, L. (2007). Bayesian Networks in reliability. *Reliability Engineering and System Safety* **92**, 92–108.

8. Huang, C. & Darwiche, A. (1996). Inference in Belief Networks: A Procedural Guide. *International Journal of Approximate Reasoning* **15**(3), 225-263.

9. Portinale, L. & Codetta-Raiteri, D. (2011). Using Dynamic Decision Networks and Extended Fault Trees for Autonomous FDIR. In Proc. 'International Conference on Tools with Artificial Intelligence', Boca Raton, FL USA.