

# RODON - A Model-Based Diagnosis Approach for the DX Diagnostic Competition

Peter Bunus<sup>1,2</sup>, Olle Isaksson<sup>2</sup>  
Beate Frey<sup>2</sup>, Burkhard Munker<sup>2</sup>

<sup>1</sup>*Department of Computer and Information Science,  
Linköping University, Sweden (e-mail: petbu@ida.liu.se).*

<sup>2</sup>*Uptime Solutions AB, Ågatan 40, 582 22, Linköping, Sweden  
(e-mail: peter.bunus@uptimeworld.com, olle.isaksson@uptimeworld.com,  
beate.frey@uptimeworld.com, burkhard.muenker@uptimeworld.com)*

---

**Abstract:** In recent years, model-based diagnostics reasoning systems have provided a major advance in fault isolation and reduction of repair time contributing to the reduction of maintenance cost of automotive and avionics systems. In this paper we highlight the main characteristics of RODON, a commercial model-based diagnostics reasoner, used in the DX Diagnostic Competition. We present the main ideas behind RODON, and the strategies that have been employed for the ADAPT tier 1, ADAPT tier 2 and the Synthetic Track of the Diagnostic Competition.

---

## 1. INTRODUCTION

With increased system complexity in recent years, almost every system under deployment is exposed to component failures or is under the risk of suffering breakdowns under its lifetime. Maintenance and repair is an ever-increasing part of the total cost of a final product. Diagnosis techniques have been adopted by the aftermarket departments of industrial systems product developers as a fast and accurate way of finding the root causes of failures. The shortened development times and the increasing complexity of the products, as indicated by the significantly increasing electrical and electronic content, may lead to difficulties if not handled appropriately. To avoid unnecessary damage, environmental or financial, there is a need to locate and diagnose these faults as fast as possible. This can be done with a diagnostic system, which produces an alarm if there is a fault in the mechanical or electrical system and, if possible, indicates the reason behind it. Traditionally diagnosis is considered the last task in the product design chain. However, the growing importance of lowering the maintenance and repair cost demands for a closer integration of diagnostics tasks in the entire design process and reuse of product related information through all the product development cycle.

Assessment and comparison of diagnosis technologies can be difficult. To facilitate this task an Advanced Diagnostics and Prognostics testbed called ADAPT has been developed at NASA Ames (Scott Poll et al. 2007). The testbed acts as a common platform where different diagnostics tools and technologies, so called test articles, can compete against each other on equal conditions. To achieve this, ADAPT consists of a controlled and monitored environment where faults are injected into the system in a controlled manner and the

performance of the test article is carefully monitored. The hardware of the testbed is an electrical power system (EPS) of a space exploration vehicle and is located in a laboratory at the NASA Ames Research Center.

At the 20<sup>th</sup> International Workshop on Principles of Diagnosis (DX-09) a diagnostics competition has been defined by NASA Ames and PARC with the goal to systematically evaluate different technologies and to produce comparable performance assessments for different diagnostics methods. The competition consisted of three different tracks. Each track defined a different diagnostic challenge problem. Two of the tracks were using real hardware test bed data based on the ADAPT system and the third track was based on ISCAS-85 and 74X-series benchmark combinatorial circuits proposed by Hansen et al. 1999. Those interested in the details, rules and set-up of the competition may wish to consult the description provided by Kurtoglu et al. 2009b and Kurtoglu et al. 2009a.

In this paper we present a model-based diagnosis approach that has been employed in the framework of the diagnostic competition to address the challenges of the competition tracks. Our approach was based on RODON, a commercial model-based reasoning (MBR) system available from Uptime Solutions AB. The authors of this paper are members of the RODON development and application team.

The rest of the paper is organized as follows: In Section 2, we give a brief description of the principles of model-based diagnosis then, in Section 3, we describe the principles of RODON, the way in which diagnosis models can be created and the ideas behind the diagnosis algorithms employed by RODON. Section 4 presents the models developed for the competition tracks together with some diagnostics results that

have been performed on the available failure scenarios. Finally, Section 5 presents a summary of the paper, the future work and our conclusions.

## 2. PRINCIPLES OF MODEL-BASED DIAGNOSIS

In the last decade, model-based technology in diagnosis matured so far that it was transferred from academic research into real applications. It provides an alternative to more traditional techniques based on experience, such as rule-based reasoning systems or case-based reasoning (CBR) systems. Early model-based diagnosis tools include MDS (Mauss et al. 2000), RAZ'R (Sachenbacher et al. 2000), Livingstone 2 (Hayden et al. 2004), LYDIA (Feldman et al. 2006), DSI Express (Gould 2004), TEAMS-QSI (Deb et al. 1998) or RODON (Lunde et al. 2006).

The basic principle of model-based diagnosis consists in comparing the actual behavior of a system, as it is observed, with the predicted behavior of the system given by a corresponding model. A discrepancy between the observed behavior of the real system and the behavior predicted by the model is a clear indication that a failure is present in the system. Diagnosis is a two-stage process: in the first stage, the error should be detected and located in the model, and in the second stage, an explanation for that error needs to be provided. Diagnoses are usually performed by analyzing the deviations between the nominal (fault free) behavior of the system and the measured or observed behavior of the malfunctioning system.

In Figure 1, a model of a real system (an airplane passenger seat system) is depicted at the lower left corner. It might contain, for example, the behavior of the mechanical components incorporated in the seats, or the behavior of the in-flight entertainment system, or both. Note that, like all models, the model is only an abstraction of the real system (depicted at the upper left corner) and can be incomplete. The granularity of the model and the amount of information and behavior captured into it will directly influence the method employed by the reasoning engine as well as the precision of the diagnostic process.

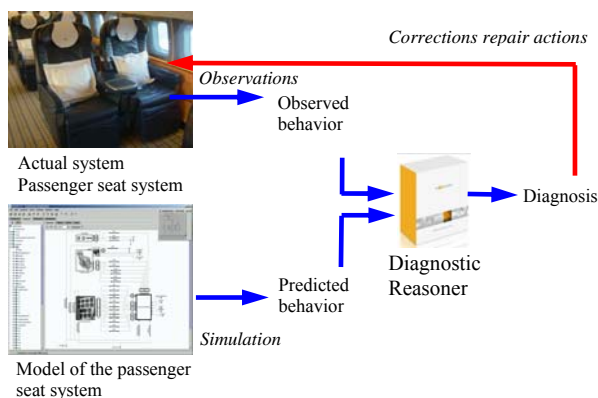


Figure 1. Illustration of the basic principle of model-based diagnosis.

As a general rule, the models are built to enable the identification of failed Line Replaceable Units (LRUs). Once

a model of the real system is built, simulation or prediction can be performed on the model. The predicted behavior, which is the result of the simulation, can then be compared with the observed behavior of the real system. This comparison is usually done by a reasoning engine (in our case RODON) that is able to detect discrepancies and also to generate and propose corrective actions that need to be performed on the real system to repair the identified fault. We should note that the process of diagnosis (incorporated in the diagnostic reasoner) is separated from the knowledge about the system under diagnosis (the model). This ensures that the model can be reused for other purposes as well, such as optimization and reliability analysis (Lunde 2003), model based FMEA support (Zampino and Burow. 2002) and BITE coverage.

## 3. RODON – A MODEL-BASED DIAGNOSIS ENGINE

RODON's Model Based Diagnostic (MBD) engine is based on "Diagnostic Reasoning Based on Structure and Behavior" as its very foundation and on the principles of the General Diagnostic Engine (GDE) as described by de Kleer and Williams 1987 and the G+DE by Heller and Struss. 2001. These principles have been further advanced and tailored to the RODON applications to increase efficiency and diagnostic power.

RODON uses contradictions (conflicts) between the simulated and the observed behavior to generate hypotheses about possible causes for the observed behavior. If the model contains failure modes besides the nominal behavior, these can be used to verify the hypotheses, which speed up the diagnostic process and improve the results. But even if not all imaginable fault modes of a component can be included in the model, it is possible to define an unknown fault mode which summarizes all other possibilities of the component to fail. Thus RODON can find unexpected or unknown faults as well.

The basic fault detection and isolation method is conflict-driven. All the available signals are fed into the model. They are then propagated across the model from the points of "value injection" or stimulation to simulate or predict the nominal behavior of the system. If during this simulation contradictions between the observed and the simulated behavior are detected, then the system is no longer consistent with the model and is assumed to be defective. The contradictions between calculated or observed values are called conflicts. They are the starting point for the diagnostic process. This process first generates candidates which may explain the deviation between the observed symptom vector and the behavior predicted by the model. If the simulation result with such a candidate is consistent with the observed values we have a possible explanation for the malfunction or a diagnosis. The results of this diagnostic process are printed to a RODON console and in addition the suspect components are highlighted in the RODON system model if the graphical user interface is activated.

RODON's general problem solving architecture is briefly depicted in Figure 2.

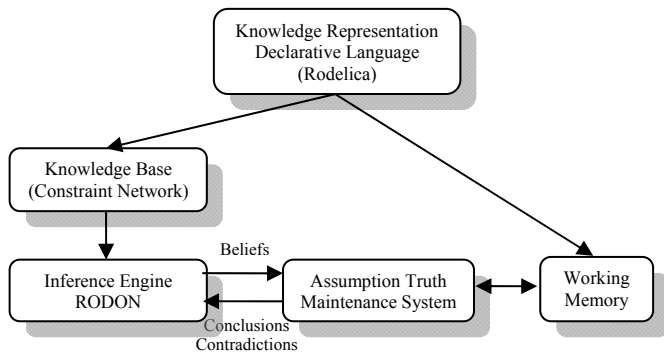


Figure 2. RODON'S general problem solving architecture.

The knowledge representation language provides the means for describing how a particular problem can be solved. In RODON the knowledge representation language is called Rodelica and has a clear syntax and semantics. The application models needs to be specified with this language. The usability requirement for such a language is that it should be easy to be used by application engineers that build the models. More details about the Rodelica language are given in the following subsections.

The syntax and semantics used at the Knowledge Representation Language Level is very high level and there is a need to translate this knowledge into the Knowledge Base in a form that can be used by the Inference Engine in the problem solving process. In RODON the Knowledge Base consists of a constraint network that is automatically extracted from the model described in the Knowledge Representation Language. A constraint network is a set of variables and constraints that inter-relate and define the valid values for the variables.

Figure 3 depicts a constraint network extracted from ADAPT tier 1 model representation in which the nodes represents variables or equations (constraints) from the model. In the zoomed view depicted in the top right corner of Figure 3 the variables are represented with yellow or green color while the constraints are represented with gray color. A connecting line between a variable node and a constraint node means that the variable is present in that constraint.

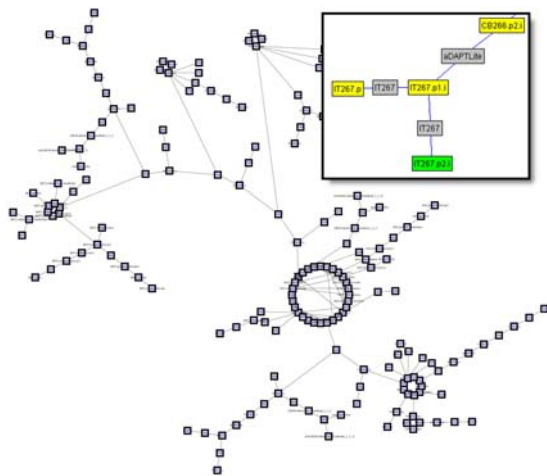


Figure 3. Constraint Network extracted from the ADAPT tier 1 model representation.

Relations can be represented extensionally by tuple sets or tables as well as intentionally using for example algebraic equations, inequalities, Boolean formulas, and splines. Discrete variable domains are supported as well as continuous variable domains. In RODON variables can be represented by intervals. This is especially useful when working with data that is subject to measurement errors or uncertainties.

Typical tasks for an Inference Engine (constraint solver) are checking whether a solution exists, computing one, some, or all solutions, or consistently extending partial instantiations. Inference strategies transform the constraint network into equivalent networks which describe the set of solutions in a more explicit way. Often they do not solve the problem directly, but they reduce the search space by problem reformulation. Transformations include reduction of variable domains and addition, removal, or modification of constraints. It is worth noting that in RODON due to interval type of the variables the inference engine used interval arithmetic for propagating the values of the variables in the constraint network. The basic arithmetic operations for two intervals  $[a, b]$  and  $[c, d]$  are given below:

$$[a,b] + [c,d] = [a + c, b + d]$$

$$[a,b] - [c,d] = [a - d, b - c]$$

$$[a,b] \times [c,d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

$$[a,b] / [c,d] = [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)]$$

The inference engine calls the Truth Maintenance System to obtain data needed in reasoning, to write conclusions into the working memory and to write data into the working memory. RODON uses a special type of truth maintenance system called Assumption Truth Maintenance System (ATMS) as the one described by de Kleer 1986. An ATMS is using an efficient labeling technique to compile dependency information between assumed and derived data.

### 3.1 The Rodelica Modeling Language

The existence of a modeling language that can be used for capturing model knowledge for diagnostics purposes is central for model-based-diagnostics. Early model-based diagnosis systems used traditional general purpose programming languages for specifying models. However, in some aspects, general purpose programming languages are inadequate to the task of formalizing significant domains of engineering practice. They lack the expressiveness and power of abstraction required by engineers. For specifying the diagnostics models RODON uses a declarative equation-based language called Rodelica which is strongly related to the equation-based object-oriented modeling language Modelica (Modelica Association 2007). Rodelica similar to Modelica has the following main characteristics:

- Acausal modeling based on mathematical equations to describe the behavior of the system.
- Multi-domain modeling capability, which gives the user the possibility to combine electrical, mechanical, ther-

modynamic, hydraulic, etc., model components within the same application model.

- A general type system that unifies object-orientation, multiple inheritance and generic templates within a single class construct. This facilitates the reuse of components and evolution of models.
- A strong software component model, with constructs for creating and connecting components.

At the lowest level of the Rodelica language, equations are used to describe the relations between the quantities of a model and to define the behavior of the class. As mentioned before, one of the distinctive features of Rodelica is the acausal programming model. The computation semantics does not depend on the order in which equations are stated. The acausality makes the library classes more reusable than traditional classes containing assignment statements where the input-output causality is fixed, since the language classes adapt to the data flow context in which they are used. The data flow context is defined by stating which variables are needed as *outputs* and which are external *inputs* to the simulated system. From the simulation practice point of view this generalization enables both simpler models and more efficient simulation. The declarative form allows a one-to-one correspondence between physical components and their software representation.

Programs in Rodelica are built from classes like in any other traditional object-oriented language. A class is intended to describe the structure of the object generated from the class. The main difference compared to traditional object-oriented languages is that instead of functions (methods), equations are used to specify behavior. A class declaration contains a list of variable declarations and a list of equations preceded by a keyword, usually `behavior`. The class concept of Rodelica is identical to the class concept of Modelica. However, we added set-valued data types as well as a new behavior description with semantics which differ from the equation-based behavior description in Modelica.

We illustrate below a class corresponding to a resistor (Resistor) modeled in Rodelica.

```

model Resistor extends TwoPin(fm (max = 2) );
  public      Resistance rNom(final min = 0);
  protected Resistance rAct(final min = 0);
behavior
  // Basic constraints for nominal case:
  if (fm == 0){
    // Actual resistance is nominal resistance
    rAct = rNom;
    // Ohms law for voltage drop between pins
    p1.u - p2.u = rAct * p1.i;
  }

  // Extensions for failure mode "disconnected":
  if (fm == 1){
    // Infinite resistance between pin 1 and 2:
    rAct = INF_PLUS;
  }

  // Extensions for failure mode "short circuit
  // between pin 1 and 2":
  if (fm == 2) {
    // Same potential at pins 1 and 2:
    p1.u = p2.u;
  }

```

```

    // No resistance between pin 1 and 2:
    rAct = 0.0;
  }
end Resistor;

```

It also should be noted that the `Resistor` class is a specialization through the inheritance mechanism of a special class called `TwoPin`. The natural inheritance mechanism in the Rodelica language works by extending classes with new equations and variables. The `TwoPin` class that defines electrical components that have two pins is defined as follows:

```

partial model TwoPin
  Pin p1;
  Pin p2;
  FailureMode fm(max =1);
behavior
  // current balance that defines the
  // nominal behavior
  p1.i + p2.i = 0;
  // constraints for the failure mode
  // "disconnected"
  if (fm == 1){
    p1.i = 0;
  }
end TwoPin;

```

The `partial` keyword before the model definition means that the `TwoPin` class does not contain enough equations to completely specify its physical behavior. The notion of partial class in Rodelica corresponds to the notion of abstract class in traditional object-oriented languages such as C++ or Java.

Compared to a Modelica representation of a `TwoPin` component, it can be noticed that the Rodelica `TwoPin` component, besides the nominal behavior, defines the behavior of the component when it is disconnected. The disconnected failure mode will have the effect that the current in `pin1` will be zero (`p1.i = 0`). The alternative behavior of the component is specified with the help of a type variable `FailureMode` that acts like a switch between the two operation modes of the component. In our case, the failure mode behavior is enclosed between the brackets of the `if(fm=1)` statement.

It should be noticed that a `Resistor` component will inherit all constraints, and thus all failure modes, from the `TwoPin` component. By extending `TwoPin`, the resistor class has the possibility to add new constraints, thus extending the inherited failure modes or even adding new failure modes. By default, nominal behavior is assigned to the failure mode `fm == 0`. In the example, it is extended by specifying that the actual resistance will take the value of the nominal resistance, and by specifying Ohm's law for the voltage drop between pins. Note that constraints which are not enclosed in any `if`-statement (like Kirchhoff's law in the `TwoPin` class) are valid in all behavior modes. It should be also noticed that the `Resistor` has an extra failure mode that captures the situation when there is a short circuit between `p1` and `p2`. In this case, `p1` and `p2` will have the same potential (`p1.u = p2.u`), and due to the short circuit the resistance of the `Resistor` will be equal to zero (`rAct = 0`). The short-circuit current is not specified within the resistor class.

This `TwoPin` class instantiates twice the class `Pin` which is a special class called connector. Such connectors declare variables that are part of the communication interface of an object defined by the connectors of that object. Thus, connectors specify the interface for interaction between a component and its surroundings. Our connector `Pin` class uses two Interval variables: one for the current (*i*) and one for the voltage (*v*). In an electrical circuit the current should always be summed up when connecting two components, and according to Kirchoff's law, the variable *i* defined in the `Pin` class will have the prefix `flow`.

```
connector Pin
  Interval u;
  flow Interval i;
end Pin;
```

Besides the instantiation of two pin interface objects, also called ports or connectors, some extra equations are provided to define the behavior of the objects such as the voltage drop along the component ( $v = p.u - n.u$ ) or the current inside the component ( $0 = p.i + n.i; i = p.i$ ). In a similar way a Ground component would be defined as follows:

```
model Ground
  Pin p;
behavior
  p.u = 0;
end Ground;
```

Connections between objects can be established between connectors of equivalent type. Rodelica supports equation-based acausal connections, which means that connections are defined as special equation forms. A connection equation form such as `connect(pin1, pin2)` with `pin1` and `pin2` of connector class `Pin`, connects the two pins so they form one node. This is equivalent to, and is eventually expanded into two equations, namely:

```
pin1.u = pin2.u; pin1.i + pin2.i = 0;
```

The first equation says that the voltages of the connected wire ends are the same. The second equation corresponds to Kirchoff's current law saying that the currents sum up to zero at a node (assuming positive value while flowing into the component). The sum-to-zero equations are generated when the prefix `flow` is used. Similar laws apply to flows in piping networks and to forces and torques in mechanical systems.

#### 4. THE RODON DIAGNOSTICS MODELS

The ADAPT system from NASA Ames consists of three major modules: a *Power Generation Unit*, a *Power Storage Unit* and a *Power Distribution Unit*. The *Power Generation Unit* can charge the batteries located in the *Power Storage Unit* with the help of two battery chargers and a photovoltaic unit (solar panel) with charge controller. The power generation unit is divided into six subsystems: the solar panel unit (not currently used), the battery charger panel, the protection and enable panel and three battery-charge selection panels. The power storage unit contains three battery packs and several relays that control the connections between the load bank and the batteries. Circuit breakers protect the power distribution unit from dangerously high currents coming from the batteries. The *Power Storage* unit is divided

into two major subsystems: the battery cabinet and the battery-load selection panel. The *Power Distribution* unit consists of two identical load banks. Each load bank is connected to the *Power Storage* unit and powers two DC loads and six AC loads. The testbed is controlled by a number of relays and monitored by a large set of sensors. Consequently, it is possible to detect an injected fault and recover from it if the correct action is taken

A complete description of the ADAPT system can be found in NASA 2006 and a complete RODON model implementation has been reported by Isaksson 2009. The complete RODON model of the ADAPT system is available upon request from the authors.

Several diagnostics systems such as HYDE (Narasimhan and Brownston ), FACT – Fault Adaptive Control Technology (Manders et al. 2006) from Vanderbilt University and TEAMS-RT – Testability Engineering and Maintenance System Real Time (Deb et al. 1998, Deb et al. 1998) have been reported to be integrated and tested on the ADAPT system. For the Diagnostics Competition we have created two different models for the ADAPT Tier 1 and for ADAPT Tier 2. For each combinatorial circuit from the synthetic track a separate model has been created. The same diagnostic algorithm has been used for all the models. The following sections will give a more detailed explanations of the models created in RODON.

##### 4.1 The ADAPT Tier 1 Model

The ADAPT Tier 1 (ADAPT Lite) model is depicted in Figure 4 and contains a battery connected through a series of circuit breakers and relays to an inverter, a phase angle transducer and a load consisting of a large fan. The rotation speed of the fan is measured by a speed transmitter component. A series of six AC or DC voltage sensors and three current transmitters measure the voltage and current in different probing points of the circuit. The circuit breakers CB236, CB136 and CB336 can be commanded externally to be closed or open and their position is monitored with the help of a position sensor connected to them. For each component type a separate library component has been created which has been instantiated in the model. The ADAPT Tier 1 and ADAPT Tier 2 models are using the same library. The implementation details of the models are not relevant for the discussion in the paper. Let us just mention that for each component, the nominal behavior was modeled and augmented with the relevant failure modes. Some of the library components models are very detailed and the models were built by following the supplier specifications of the real component. A detailed description of those models can be found in Isaksson 2009.

Each sensor data frame was sent to the diagnostics engine via an application programming interface API and a diagnosis has been computed. The diagnostics algorithm, for the ADAPT Tier 1 model was configured to find diagnostics up to triple faults. The evaluation results performed on the available 59 experiments are given below:

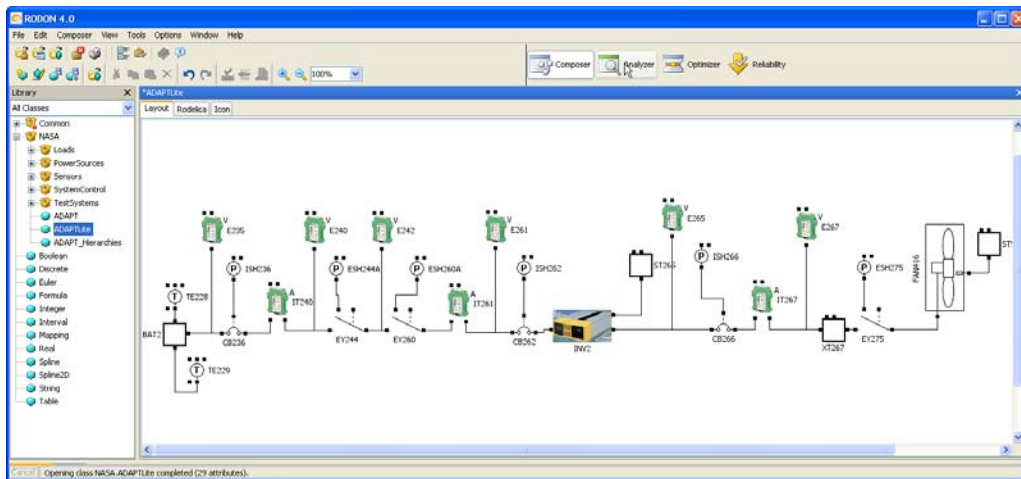


Figure 4. The RODON model for the ADAPT Tier 1 competition challenge.

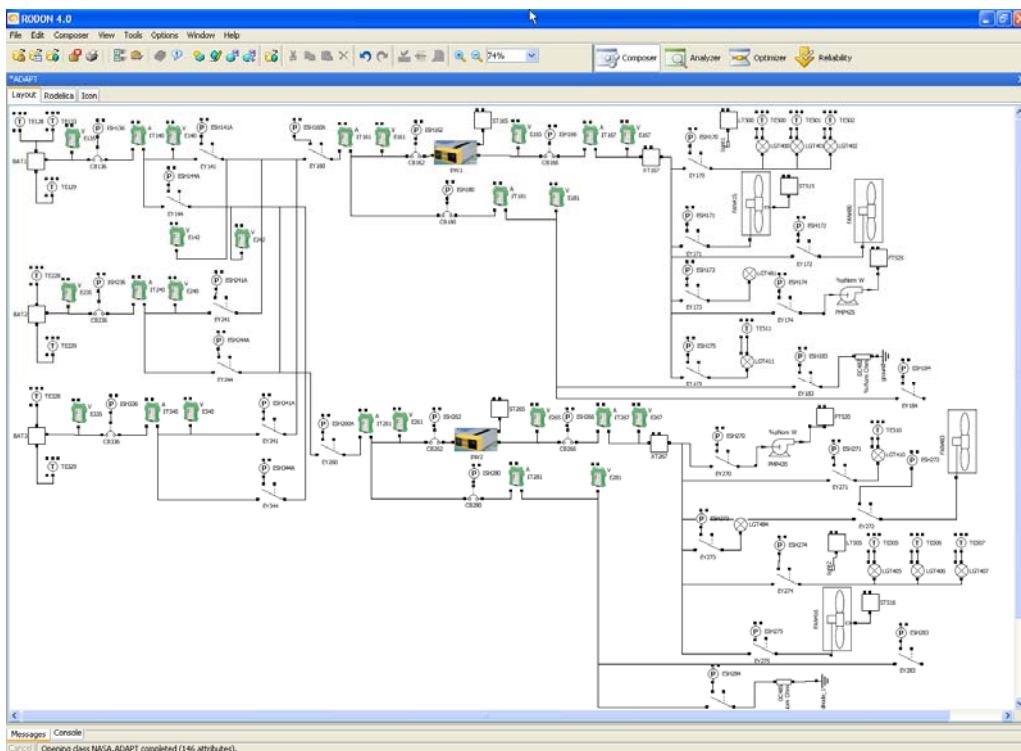


Figure 5. The graphical view of the ADAPT Tier 2 RODON model.

Classification Errors	4.0
Detection Accuracy	0,9492
False Negatives Rate	0,1111
False Positives Rate	0,0
Mean Cpu Time (ms):	1753 ms
Mean Time to Detect	315 ms
Mean Time to Isolate	5766 ms
Mean Peak Memory Usage	27032 kb

All the experiments have been performed on a laptop computer with an Intel Core™2 Duo CPU 2.2GHz processors, and 2GB of RAM running Microsoft Windows XP Professional. A formal description of the diagnostics metrics is described by Kurtoglu et al. 2009a.

It should be noted the high detection and isolation accuracy of the algorithm. The positive number of False Negative Rate

is due to Exp\_578\_064\_pb\_t1f and Exp\_578\_107\_pb\_t1f. In both experiments the injected fault is a stuck voltage sensor E242 and E265 respectively. Both voltage sensors are stuck in the nominal range (see Figure 6a and Figure 6b) and there is no other negative effect on the system. For this reason the diagnostics algorithm was not able to detect these faults. Similarly, in experiment Exp\_593\_pb\_t1 a failure mode was injected by introducing a parasitic load to the battery which caused the voltage levels in the line to decrease slightly from the levels prior to fault injection. The voltage level in the battery after the fault injection was still in the nominal range and therefore remained undetected by the diagnosis algorithm. However these faults could have been detected by providing a preliminary sensor data filtering that could capture the stuck behavior.

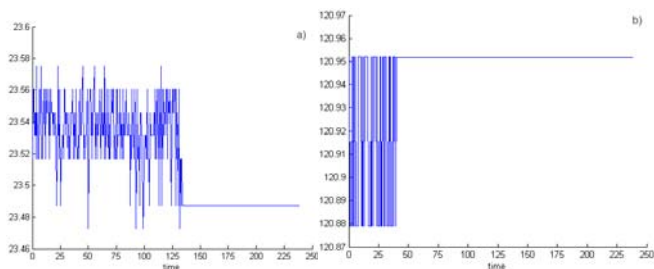


Figure 6. a) In Exp\_578\_064\_pb\_t1f the voltage sensor E242 is stuck at time 133,141s in the nominal admissible voltage ranges. b) In Exp\_578\_107\_pb\_t1f the voltage sensor E265 is stuck at time 39,46s in the nominal admissible voltage ranges.

The mean time to detect (315 ms) is below the sampling rate of the sensor data. Due to the simplicity of ADAPT Tier 1 model and the availability of sensors, the time to detect and to compute the diagnosis candidates is very fast. The power of the RODON inference engine was not fully used for this particular model. The high value of the Mean Peak Memory Usage is due to the fact that the whole RODON inference engine was used and loaded when executing the experiments. We tried to keep the modification on the engine itself to a minimum for the competition and used almost the same configuration of the on-board diagnostics engine that is deployed for other industrial applications.

#### 4.2 The ADAPT Tier 2 Model

The ADAPT Tier 2 model is depicted in Figure 5. It consists of three batteries that can be connected through two inverters to two different load banks. For developing the model we have used the same component library as the one used for the ADAPT Tier 1 model and the same diagnosis algorithm with the same settings. The diagnostics algorithm, for the ADAPT Tier 2 model was configured to find diagnostics up to triple faults. The evaluation results performed on the available 113 experiments are given below:

Classification Errors	32.66
Detection Accuracy	0.9823
False Negatives Rate	0.027
False Positives Rate	0.0
Mean Cpu Time (ms):	21029 ms
Mean Time to Detect	712 ms
Mean Time to Isolate	8168 ms
Mean Peak Memory Usage	28473 kb

The Detection and Isolation Accuracy for this set of experiments is also very high. The small False Negative Rate, as it was the case with the ADAPT Tier 1 model, is due to a voltage sensors E265 in experiment Exp\_628\_pb\_t2f and a current transmitter IT261 in experiment Exp\_639\_pb\_t2f stuck in the nominal range that were not detected by the diagnosis algorithm. Noisy data was influencing the diagnosis results especially the Mean Time to Detection. In experiment Exp\_616\_pb\_t2 a peak in the current sensor IT 167 made that the diagnostics engine reported a double fault (INV2 FailedOff & IT 167 Stuck) that increased the isolation time (see Figure 7a).

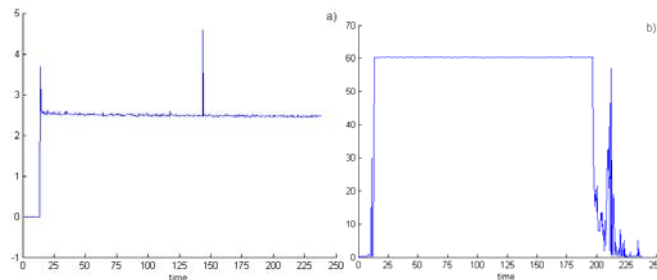


Figure 7. a) A noise in the current sensor IT 167 in experiment Exp\_616\_pb\_t2 determined the diagnostic algorithm to report a double fault. b) A noise in ST165 in experiment Exp\_620\_pb\_t2 determined the diagnostic algorithm to report a double fault.

A similar situation occurs in experiment Exp\_620\_pb\_t2 where a noise in the ST165 around frame 430 determined the diagnostic algorithm to report a double fault that caused an increase of the fault isolation time (see Figure 7b).

#### 4.3 The Synthetic Track Models

The Rodelica models corresponding for the Synthetic track were automatically generated by a translator from the provided XML specification of the combinatorial circuit. A very simple combinatorial circuit library has also been built in RODON with basic combinatorial gates. Only one fault mode “faulty” was specified for each component. For the Synthetic track a different strategy has been employed for computing the diagnosis: the diagnostics engine was configured to compute as many as possible diagnosis in the given time up to multiple faults of order 5. This strategy would guarantee a higher Detection Accuracy but for some scenarios might produce more candidates than it is necessary lowering the Isolation Accuracy and increasing the Mean CPU Time and Mean Memory Usage. The evaluation results performed on the available training experiments are given below:

Circuit name	Isolation Accuracy	Mean Cpu Time (ms)	Mean Peak Memory Usage KB
c432	0,9955	13192,05	43911,4
c499	0,9963	13233,55	34285,8
c880	0,9982	9758,5	30127,8
c1355	0,999	14277,4	37905,0
c1908	0,9992	14729,6	39918,4
c2670	0,9995	16316,2	37885,6
c3540	0,9996	19876,5	55337,0
c5315	0,9997	25914,1	51226,2
c6288	0,9998	41149,8	69054,4
c7752	0,9992	14729,6	39918,4
74L85	0,9846	1114,7	20751,6
74181	0,9882	4967,1	25882,2
74182	0,969	1853,6	20671,84
74283	0,9824	1733,5	20307,2

The Detection Accuracy of 0,95 for the for the experiments performed on circuits c1355, c6288 and c7752 is due to the fact that the diagnostic algorithm was stopped before it managed to compute the real candidates. An additional 10 seconds increase in the post scenario time-out

(POST\_SCN\_TIME\_M) specified in the Dxc.cfg configuration file would have resulted in a 1,0 detection accuracy. The bigger models c5315, c6288 and c7752 required some additional time to be loaded by the diagnostic algorithm and for this reason the diagnosis algorithm timeout DA\_TIMEOUT\_MS was extended from 15000 to 3500 in the Dxc.cfg diagnostic framework configuration file. The other synthetic combinatorial circuits were successfully loaded and executed without any modification of the framework timeout parameters.

## 5. SUMMARY AND CONCLUSIONS

In this paper, we presented an evaluation of RODON a commercial model-based diagnosis system in the context of the Diagnostic Competition hosted at the 20<sup>th</sup> International Workshop on Principles of Diagnosis (DX-09). The diagnostics capability of RODON has been demonstrated by applying several fault scenarios to the model for which diagnostics candidates have been generated. We have briefly presented the models that have been developed for the three tracks of the competition together with the results obtained from the available training experiments. The source code of the models and the binary of the diagnostic algorithm used for the competition will be made available for download at DASHLink<sup>1</sup> web site.

## ACKNOWLEDGEMENTS

We would like thank to Werner Seibold and the RODON development team at Uptime Solutions AB Sweden for valuable discussions and the feedback received for this paper.

## REFERENCES

- Association Modelica (2007) "Modelica - A Unified Object-Oriented Language for Physical Systems Modeling - Language Specification Version 3.0," September, 2007.
- de Kleer Johan. (1986) "Problem solving with the ATMS." *Artificial Intelligence*, vol. 28: 2, pp. 197-224, 1986.
- Deb Somnath, Sudipto Ghoshal, Amit Mathur, Roshan Shrestha, and Krishna R. Pattipati. (1998). "Multisignal Modeling for Diagnosis, FMECA, and Reliability." In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 11-14 October, 1998).
- Feldman Alexander, Jurryt Pietersma, and Arjan van Gemund. (2006). "All Roads Lead to Fault Diagnosis: Model-Based Reasoning with Lydia." In *Proceedings of the Eighteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC-06)*. (Namur, Belgium, October, 2006).
- Gould Eric. (2004). "Modeling it Both Ways - Hybrid Diagnostic Modeling and its Application to Hierarchical System Designs." In *Proceedings of the Annual Systems Readiness Technology Conference (Autotestcon 2004)*. (San Antonio, Texas, September 20-23, 2004).
- Hansen M., H. Yalcin, and J. P. Hayes. (1999) "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering." *IEEE Design and Test*, vol. 16: 3, pp. 72-80, 1999.
- Hayden Sandra C., Adam J. Sweet, and Scott E. Christa. (2004). "Livingstone Model-Based Diagnosis of Earth Observing One." In *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*. (Chicago, Illinois, 20 - 22 September, 2004).
- Heller Ulrich and Peter Struss. (2001). "G+DE - The Generalized Diagnosis Engine." In *Proceedings of the 12th International Workshop on Principles of Diagnosis*. (Via Lattea, Italy, March 7-9, 2001).
- Isaksson Olle. (2009). *Model-based Diagnosis of a Satellite Electrical Power System with RODON*. Master Thesis. Department of Electrical Engineering, Linköping University, 2009.
- Kleer Johan de and Brian C. Williams. (1987) "Diagnosing Multiple Faults." *Artificial Intelligence*, vol. 32, pp. 97-130, 1987.
- Kurtoglu T., S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman. (2009a). "First International Diagnosis Competition – DXC'09." In *Proceedings of the 20th International Workshop on Principles of Diagnosis (DX-09)*. (Stockholm, Sweden, June, 2009a).
- Kurtoglu T., S. Narasimhan, D. Garcia S. Poll, L. Kuhn, J. de Kleer, A. van Gemund, and A. Feldman. (2009b). "Towards a Framework for Evaluating and Comparing Diagnosis Algorithms." In *Proceedings of the 20th International Workshop on Principles of Diagnosis (DX-09)*. (Stockholm, Sweden, June, 2009b).
- Lunde Karin. (2003) "Ensuring system safety is more efficient." *Aircraft Engineering and Aerospace Technology*, vol. 75: 5, pp. 477 - 484, 2003.
- Lunde Karin, Rüdiger Lunde, and Burkhard Münker. (2006). "Model-Based Failure Analysis with RODON." In *Proceedings of the ECAI 2006 - 17th European Conference on Artificial Intelligence* (Riva del Garda, Italy, August 29 -- September 1, 2006).
- Manders Eric-J., Gautam Biswas, Nagabhushan Mahadevan, and Gabor Karsai. (2006). "Component-oriented modeling of hybrid dynamic systems using the Generic Modeling Environment." In *Proceedings of the Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*. (Potsdam, Germany, March 30, 2006).
- Mauss Jakob, Volker May, and Mugur Tatar. (2000). "Towards model-based engineering: Failure analysis with MDS." In *Proceedings of the ECAI-2000 Workshop on Knowledge-Based Systems for Model-Based Engineering*. (Berlin, Germany, 2000).
- Narasimhan S. and L. Brownston. "HyDE- A General Framework for Stochastic and Hybrid Model-based Diagnosis." In *Proceedings of the 18th International Workshop on Principles of Diagnosis*. (Nashville, TN).
- NASA Ames Research Center (2006) "Advanced Diagnostics and Prognostics Testbed (ADAPT) System Description, Operations, and Safety Manual," February, 2006.
- Sachenbacher Martin, Peter Struss, and Claes M. Carlén. (2000) "A prototype for model-based on board diagnosis of automotive systems." *AI Communications*, vol. 13: 2, pp. 83 - 97, 2000.
- Scott Poll, Ann Patterson-Hine, Joe Camisa, David Garcia, David Hall, Charles Lee, Ole J. Mengshoel, Christian Neukom, David Nishikawa, John Ossenfort, Adam Sweet, Serge Yentus, Indranil Roychoudhury, Matthew Daigle, Gautam Biswas, and Xenofon Koutsoukos. (2007). "Advanced Diagnostics and Prognostics Testbed." In *Proceedings of the In Proceedings of International Workshop on Principles of Diagnosis (DX-07)*. (Nashville, TN, May 2007, 2007).
- Zampino Edward J. and Dirk Burow. (2002). "The Application of RODON to the FMEA of a Microgravity Facility Subsystem." In *Proceedings of the Annual Reliability and Maintainability Symposium*. (Seattle, WA, USA, 28-31 Jan, 2002).

<sup>1</sup> <https://dashlink.arc.nasa.gov/member/petbu/>