Guidelines for modeling hydraulic components and model based diagnostics of hydraulic applications

Sebastian Adén[1], Kenny Stjernström[2]

[1]Linköping University, Department of Management and Engineering, Linköping, Sweden
E-mail: sebad028@student.liu.se

[2]Combitech AB- Technical Information Solutions, Linköping, Sweden
E-mail: kenny.stjernstrom@combitech.se

Abstract

Model based diagnosis is a very hot topic right now in the world of system engineering, through RODON the user is provided with the necessary tools to accomplish this. With this software one can model a hydraulic system and auto-generate fmea (failure-mode-effect-analysis). The old fashioned way for solving this procedure is with pen and paper and that's a very time-demanding activity. In addition to previous attributes of traditional failure-mode-effect-analysis: The complexity-grade is increasing fast in large machinery. Due to the origin of the paper, i.e. as a product of a master thesis; a huge effort has been put in to understand how RODON works, and how hydraulics best should be modeled to provide the user with as much information and accuracy as possible.

Keywords: System, modeling, auto-generated fmea, diagnostics, non-causal interface.

1 Introduction

1.1 Background to paper and objectives

This paper is a milestone in one of the authors' master thesis. It should be considered as a method suggestion for auto-generating fmea of hydraulic machinery. The main goal of the thesis is to develop an expansion of RODONs standard component library, with aim of hydraulic and pneumatic components. The results from this paper are the same as the results from the thesis; up to the deadline-date of submission for full paper to SICFP (April 14, 2013).

1.2 Introduction to paper

FMEA is a conventional tool for the investigation; how effects of component fault modes propagating the system [1]. Often it is used in an early state of production, to find construction-critical areas, but as well for maintenance work in the field. With RODON such analysis are auto-generated, in the terms of: system behavior is simulated with activated failure modes. With this information one can evaluate construction-concept and also find the candidate failure mode that's consistent with the shown behavior.

Modeling in RODON is built on the idea that systems can be described in terms of intensity- and flow-variables. Such observations are often represented in bond graphs.

In autonomous diagnostics methods, were diagnosis is stated by sensors and microcontrollers, the diagnostic engineer got harsh requirements for constructing accurate models; off-line diagnosis can be run on much more simplified diagnosis models.

2 Background

The paper assumes basic knowledge of system engineering techniques, failure-mode-effect-analysis, logic and control theory. In some parts more, advanced theory is used, but for overall understanding basic knowledge is enough.

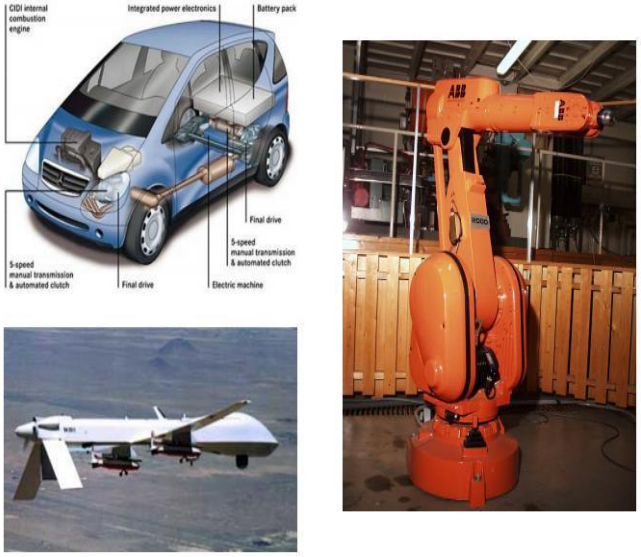

Figure 1 Different areas where diagnose is a common tool, photo: ISY, LinköpingsUniversity

2.1 Diagnosis

For the uninitiated, the term diagnosis is often only associated with medical diagnosis. But a diagnosis can also be stated for a technical system that got a headache. In this case the diagnose-algorithm is the doctor and the system is the patient.

The aim of a diagnosis is to decide whether the system is in a fault mode or not and in case of fault, identify the root cause [4]. There are many types of diagnosis and methods for generate them. This paper will focus on how to achieve diagnosis study on hydraulic applications, with help of stationary conditions. The concept is evaluated off-line. I.e. the analysis is not automated by sensor, in the sentence of; compare sensor readings with a threshold. If the sensor signal exceeds the threshold level, the system is considered faulty and the user should be warned.

Traditional diagnosis methods and the original version of fmea are to use a set of diagnostic rules, based on experience. A diagnostic rule could look something like: "If the lamp is not lit when the switch is turned on, either the switch is stuck in open position or the lamp is broken or both components are defect". The diagnosis method that's been used in this paper is of the type model-based and will be explained in the following sections.

2.2 RODON

Combitech develop and market products and services in model based diagnosis, on the platform of their own software; RODON. In RODON it is possible to create system models from existing component-libraries and from those generate different kinds of failure-analysis, like Failure-Mode-Effect-Analysis (FMEA) and diagnosis in form of Decision Trees (DT). RODON can also do the opposite; state diagnosis based on symptoms given by the user.

For mechatronic systems, typical in automotives, there are well developed experiences in terms of methods and component-library for how these systems best should be modeled in RODON and to get a FMEA that's satisfies the demands from the industry. An important aspect for successfully find the right method of modeling a system for auto-generated FMEA, is to investigate on which "level of detail" previous work is done. Shortly; the failure-analysis in RODON should generate the same system-information, as a manual made FMEA but as a considerably less time consuming activity and with an increased quality.

Today RODON is used in the automotive industry, with the work performed in this paper; the ambitions are that RODON should be able to perform diagnosis on hydraulic machinery as well.

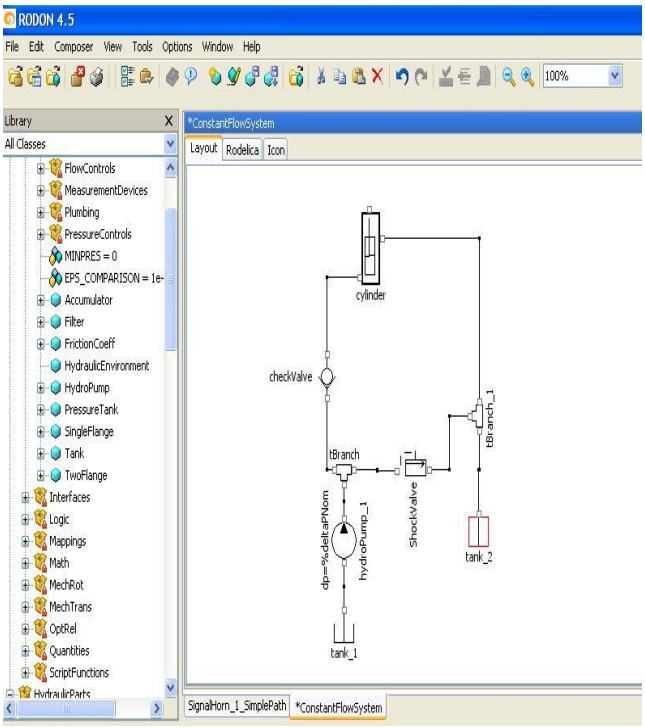

Figure 2 RODON's Graphical User Interface, Simple hydraulic system

RODON provides an equation-based object orienting language called Rodelica [2]. The syntax is related to Modelica [3]. Some of the Rodelica features are:

- It is based on interval-arithmetic, instead of sharp values.
- It supports non-causal interface, i.e. the user is not in need of propagate calculations order.
- Simulation routine with activated failure modes.
- Or-statement. This operator compare a set of conditions during simulation and executing only the one that could be valid for the whole system.

The auto-generating fmea-routine in RODON can be dived into some main steps:

- The Composer (figure 2). Where the user creates a class of the system.
- The Analyzer (figure 4), for investigation of implemented equation.
- SDBView for generate fmea
- DtGen, evaluation of decision trees.

Editing and creation of a class can be constructed in a graphical way (drag-and-drop from a component-library) or in pure Rodelica-code, as for the case in figure 3.

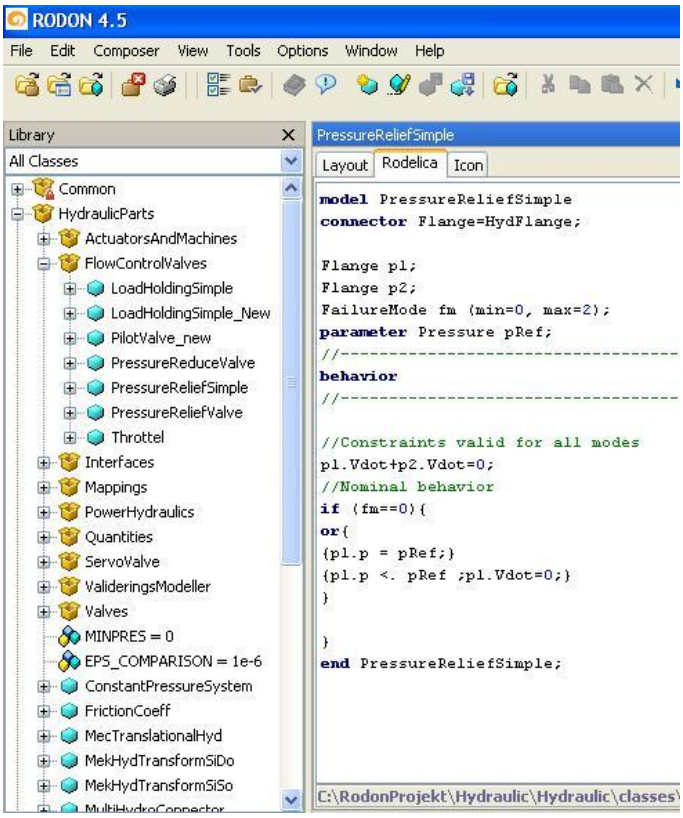

Figure 3 RODON's Graphical User Interface, Rodelica code for pressure relief-class

In figure 3 the Rodelica-interface for a pressure relief-valve is presented. Development of a component in general, consists basically of two steps; Instructions for nominal behavior and component behavior when a fault-mode is active. For the case in figure 3, no behavior for defect mode has been implemented yet.

The next step in the fmea-routine is to import the model into the Analyzer, like in figure 4.In Analyzer-mode, the user can simulate the system and evaluate the implemented equations. But foremost; in the Analyzer-mode, the user have access to what RODON call AutoSim. With this tool a simulation-routine can be initiated and for every single/multiple fault, the effect can be evaluated by the RODON- engine. This comes in handy when a diagnostic engineer, wants to find out candidates to debilitating/defect behavior. Symptoms of this kind are implemented in Composer-mode on system-level. To draw a parallel to the situation in figure 4 (horn system); a symptom could be implemented based on the statement "the horn is not sounding, although the nominal value of switch-position is turned to position *on*". This is realized by the following Rodelica code-snippet.

```
//-----------------------------------------------------------------
behavior
//-----------------------------------------------------------------
 //At SystemLevel
//Implemented condition for the case of considering
symptom signalHornNoSound to be  true:

signalHornNoSound:=(switchOnOff1.posNom==1 &
signalHorn1.sound==0);

   connect(gnd.pHarness, wireGnd.p2);
   connect(wireGnd.p1, signalHorn1.p2);
   connect(signalHorn1.p1, switchOnOff1.p1);
   connect(switchOnOff1.p0, wireBattSwitch.p2);
   connect(wireBattSwitch.p1, batterySimple1.p);
```

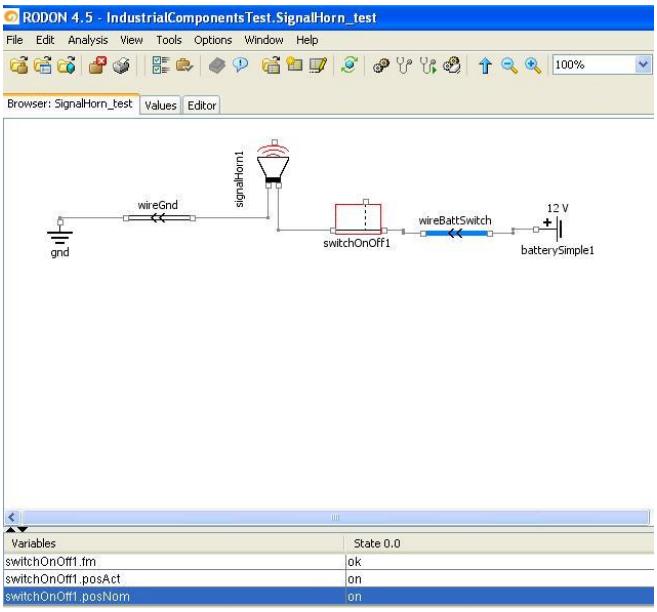

Figure 4 Graphical User Interface, Analyzer-mode of a horn-system

During AutoSim-routine, every combination of component-status, for which the system generates outputs that satisfy the symptom-condition; RODON will store information like this in a special database. Concrete; RODON have found candidates, which can explain system behavior and state a diagnosis.

As been said before, RODON can present diagnosis in form of Decision Trees (DT). In the case of defect horn-system, this could look something like figure 5.
From this view a user can navigate through the remaining candidates to find the root cause of the debilitating behavior.

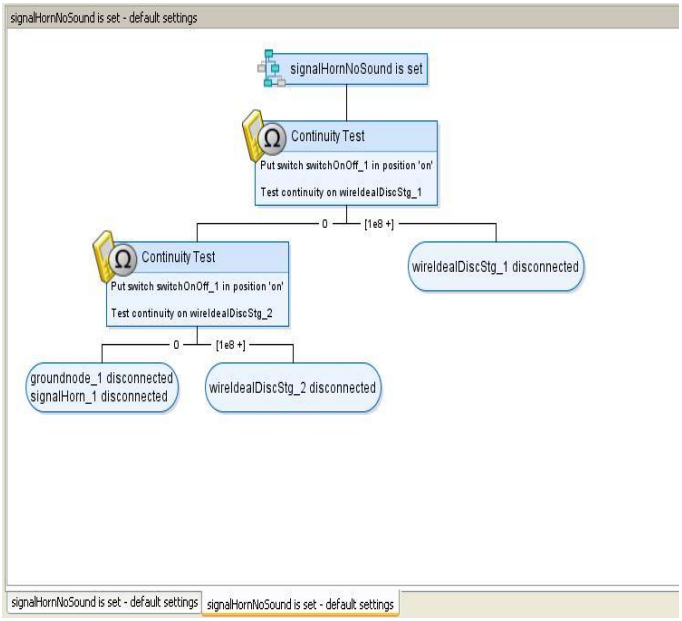

Figure 5, Decision Tree for guided diagnostic of horn system

2.3 Diagnosis system

Diagnosis systems are implemented in a wide area of technical system [4]. For example:

- Electrical motors
- Industrial robots
- Automotive

The reason and argument for implementing a diagnosis system are foremost:

- Safety
- Environment/Machine protection
- Maintenance work

With incorporate diagnosis system of an off-line approach, an ease of maintenance work is likely the goal of implementation. This is how RODON is used today; from generic component models, join into large mechatronic systems (with multiple physical domains), make statements of possible candidate(s) which is/are consistent with observed symptoms.

2.4 Diagnosis model

The definition of model-based diagnosis is: the diagnosis is based on an explicit formal model of the system. The first research reports of the subject were published in the beginning of the 1970s. [4].

A model that is used for diagnosis analysis is called a diagnosis model. It is used for evaluating presence of fault, given observation(s). In practice that means: A diagnostic model doesn't necessarily need to calculate the exact physical behavior, as long as it represents the behavior of interest for making statements regarding component health.

The simplest form of model based diagnostics is to model the system for nominal behavior and compare with observations. Results from such test are then evaluated against a threshold, in diagnose theory terms; one have created a residual, i.e. the numerical difference between system output and model output based on the same input, that's ideal zero (no fault). The computation routine is explained graphically in figure 6.

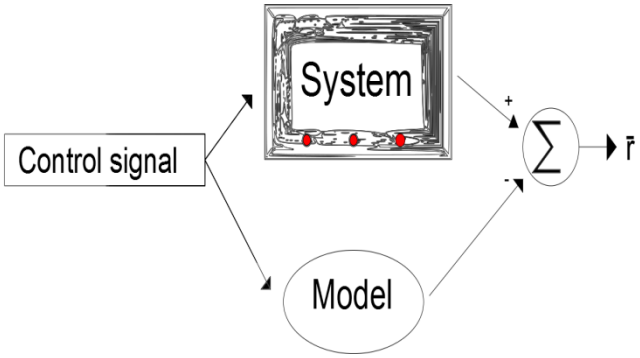

Figure 6 generating residual for diagnosis purpose

The size of absolute residual (r) -value is compared against a threshold (J).

In the case of r is larger than the threshold, the system is considered faulty. Due to the well-known nonlinear nature of hydraulics as well as the diagnostic engineers possibilities to accurate measure flow and pressure in arbitrary areas, adequate models for online diagnosis are rather hard to establish.

In a component-based reasoning approach (like the one in RODON), outputs from a faulty system can not have completely arbitrary values. It is the application engineer's assignment to construct components with well-defined nominal and faulty behavior.

A simple system is shown in figure 7, it contains two adders and three multiplies and illustrates the model-based reasoning for evaluating of error.

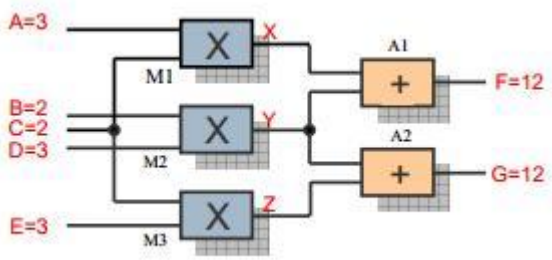

Figure 7 Multiplier- and Adder circuit, photo: Kleer and Kurien [6]

Inputs to the system are A, B, C, D and E, while F and G are the outputs. X, Y and Z can be considered as probing points. In the case of hydraulic environment, this corresponds to pressure and flow transducers. If the system is provided with the input of A=3, B=2, C=2, D=3, E=3, the expected output signal should become F=12 and G= 12, i.e. for system in nominal behavioral mode. A conflict is every value that differs from the nominal.

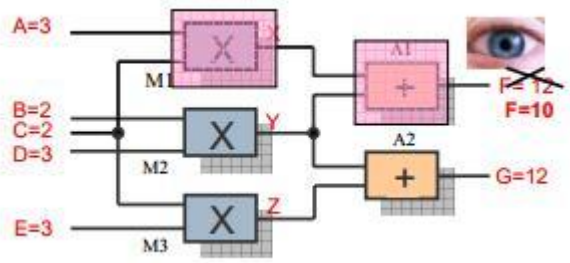

Figure 8 Multiplier- and Adder circuit with inconsistent behavior, photo: Kleer and Kurien [6]

Possible candidates to explain the behavior shown in figure 8 are: multiplier M1 or adder A1 is defect. This diagnosis-statement is based on logic reasoning, due to output G still acting consistent with predicted behavior. From figure 8, one can evaluate that G is decoupled from M1 and A1, and for the case of only single fault (defect M2 and A2 can also explain above observations) there could not be any other candidates than M1 or A1. For isolating one candidate and state a minimal diagnosis, the probing points needs to taking into account as well.

2.5 Level of detail

From above discussion about reasoning approach to diagnosis, the natural extension of that subject is; on which level of detail should components be developed, so they could be used both for nominal- as well as for faulty-behavior in a generic way. It should be noted that the main task of a diagnosis model is not to perform a simulation.

For example: When modeling a relay, from a diagnosis point of view. It's not relevant that equations corresponding to current-flow through the coil are implemented in the model (if not the interesting symptoms expects to depend on that specific behavior). An easier approach is to represent the relay with a resistance and a discrete switch, the function of the "simple-relay" is then emphasized from knowledge according to effect flow in the resistance-element.

2.6 Simple hydraulic modeling

The interface between components in Rodelica [2] is built on the idea that many systems can be described in terms of flow- and intensity-variables. Such observations are often represented in bond graphs [5]. For modeling work in RODON, no account needs to been taken to maintain conflict-free causality. The solver deals with this during its simulation-routine.

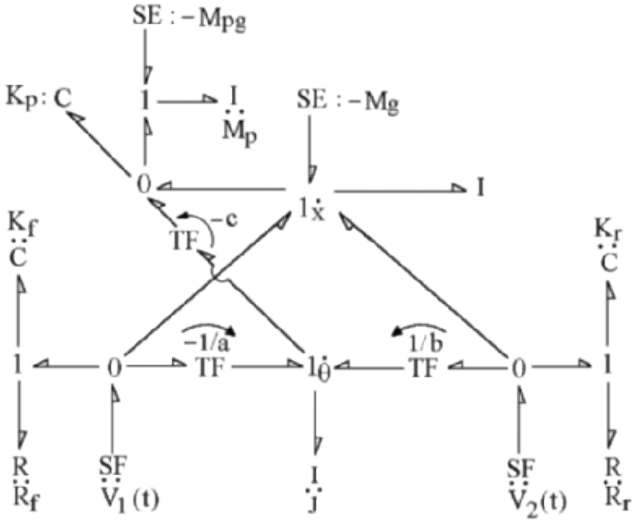

Figure 9 Bond graph of a mechatronic system, photo: bondgraph.org

For hydraulic environments these flow- and intensity variables are directly translated to flow and pressure respectively.

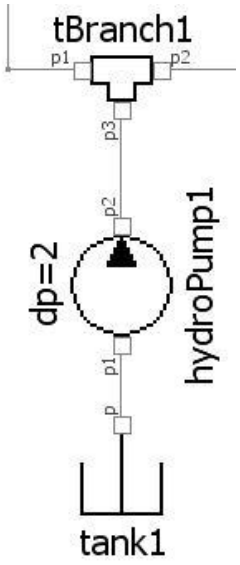

Figure 10, component-interface in RODON

Since Rodelica is built on interval-arithmetic, the only sharp value that has been declared for a hydraulic environment is tank-pressure (apart from parameters). Value-assignment for the rest of the hydraulic circuit is done by simulation-routine.

3 Results

In the following subsections results from the thesis to present day will be presented.

3.1 Components

As been described; the master thesis that this paper is based on, aims to develop a standard component-library in RODON.

Hydraulic components can rather rough be divided into following classes:

- Hydraulic machines (pumps and motors)
- Actuators
- Directional valves
- Pressure-control valves
- Flow-control valves
- Check-valves
- Pipes

For which objects can be instantiated. With different attributes e.g.:

- Proportional control
- Servo control
- Linear/non-linear characteristics

3.2 Hydraulic modeling in RODON

In the world of interval arithmetic and diagnostic approach; programming of component-classes can become quite confusing at first glance. Since one not normally propagates calculation order in RODON, the function-structure differs from a conventional simulation-program (e.g. MATLAB/Simulink). The easiest way of illustrate above statement is to look in to the Rodelica-code of a throttle valve.

```
model Throttel
/** Hydromechanical connectors*/
connector Flange=HydFlange;
Flange p1, p2;
parameter FrictionCoeff kFriction = 1;
FailureMode fm (max = 1, mapping = "ok, blocked");
// ----------------------------------------------------------------------
behavior
//----------------------------------------------------------------------
  // Constraints valid for all modes:
  // Volume flow balance:
  p1.Vdot + p2.Vdot = 0;
   // Definition of pressure drop:
  deltaP = p1.p - p2.p;

   // Basic constraints for nominal case:
  if (fm == 0)
 {
        deltaP = kFriction * p1.Vdot;
 }
// Constraints for failure mode "blocked":
  if (fm == 1)
{       // No volume flow through pin 1:
        p1.Vdot = 0;
 }
end Throttel;
```

In the Rodelica-code for a throttle; no propagation is taken place for input/output signal, nor which of the ports that acting input/output. Depending of components, more or less sophisticated failure-modes have been implemented. A failure-mode can be considered as a state-machine; whenever a fault is present, the component should act in a predefined way.

As a first approach in component-modeling of hydraulics; only trivial failure modes are implemented, e.g. clog (as in the case of throttle-model) or pressure drain to atmosphere.

Verbal defined functions:

- If the clog- failure mode is active, then set flow through component to zero
- If component suffer from leakage to atmosphere, then set component pressure to atmosphere pressure.

Due to diagnosis purpose, modeling can be done with different abstraction-level (2.5 level of detail). Consider a load holding-valve like the one in figure 11.

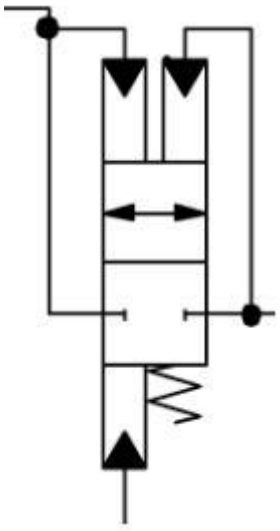

Figure 11 Schematic view of a conventional load holding-valve, photo: used with permission from Volvo CE

A Newtonian force balance of the component in figure 11 exhibit at least four well-defined forces. These are all relevant for simulation purpose, due to the study of dynamic/static behavior. Suppose now that the purpose of modeling is to investigate the health of the load holding-valve; an abstraction of component-function becomes:

- Depending on flow direction; No effect-losses on the high-pressure side, i.e. free flow. On the low-pressure side, a small pressure build up is propagated, i.e. component acting like a throttle. If a local or global fault is active, then the behavior will change.

An implementation of an abstracted load holding-valve in RODON is presented in figure 12. One demand in the master thesis is that all components should be constructed in a generic way; when the component classes are created, additional features are implemented as well. For the case of the load holding-valve in figure 12, an extra port (p3) has been added. Due to this one can e.g. let the valve function depend on a pilot valve or some other external components. For diagnosis this is very important; Consider the load holding-valve in figure 11, internal failure modes could be {always open, spring broken}. But system behavior will also be affected if a pilot valve suffer from leakage or stucks in closed postion, the effects of these modes are spread to load holding-valve through port p3. In this case the pilot valve should be considered as an information port, i.e. the simplest way for modeling a pilot valve is to assign it carateristics of a discrete switch.

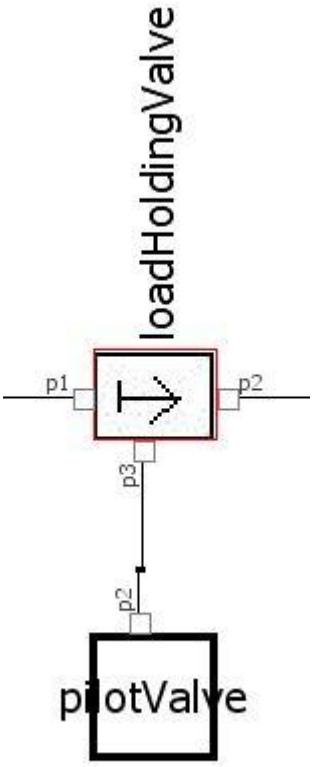

Figure 12 Schematic representation of a load holding-valve

In summary the discussion according to the load holding-valve ends up in the following Rodelica-code:

```
model LoadHoldingValve
/** Hydromechanical connectors*/
connector Flange=HydFlange;
/**Connector prepared for connection with pilot valve*/
connector actPos= Discrete;
```

```
/**Declare connector for pilotvalve*/
actPos p3;

  FailureMode fm (max = 0, mapping = "ok, stucksClosed, stucksOpen");
  Discrete modeNom (min=0, max=1, mapping = "closed, open");
  Discrete modeAct (min=0, max=1, mapping = "closed, open");
/**user defined parameters*/
  parameter Interval fricNom;
  parameter Interval fric2;
  parameter Interval fricLeak;
  parameter Interval fric1;
//-----------------------------------------------------------------------
behavior
//-----------------------------------------------------------------------
  deltaP = p1.p - p2.p;
  p2.Vdot + p1.Vdot = 0;
  or
  {
        {
          //Behavior is not propagated from pilot valve
          p3=0;
          modeNom = 0;
        }
        {
         //Behavior is  propagated from pilot valve
          p3> 0;
          modeNom = 1;
        }
 }
   if (modeAct==0)
 {
        p1.Vdot * fric2 = deltaP;
 }
 if (modeAct==1)
 {
        //Nominal behavior
        if(p3==1){
        p1.Vdot * fric1 = deltaP;
        }
        // Leakage over pilotValve
        if(p3==2){
        p1.Vdot * fricLeak = deltaP;
        }
        // PilotValve stuck in closed position
        if(p3==3){
        //only leakage flow over loadholding valve
        p2.Vdot= 0;
        }
 }
 // nominal case:
 if (fm==0)
 {
        modeAct = modeNom;
 }
end LoadHoldingValve;
```

3.3 Effects due to failure

Throughout the standard library that has been developed in the thesis, a certain level of abstraction has been used. In this section some of the component will be used for modeling the working hydraulics in a wheel loader and auto-generated decision trees for guided diagnostics will also be presented.

Figure 13 Volvo Wheel loader, photo: used with permission from Volvo CE

The working hydraulics of the tilting function in a wheel loader is represented in figure 14.

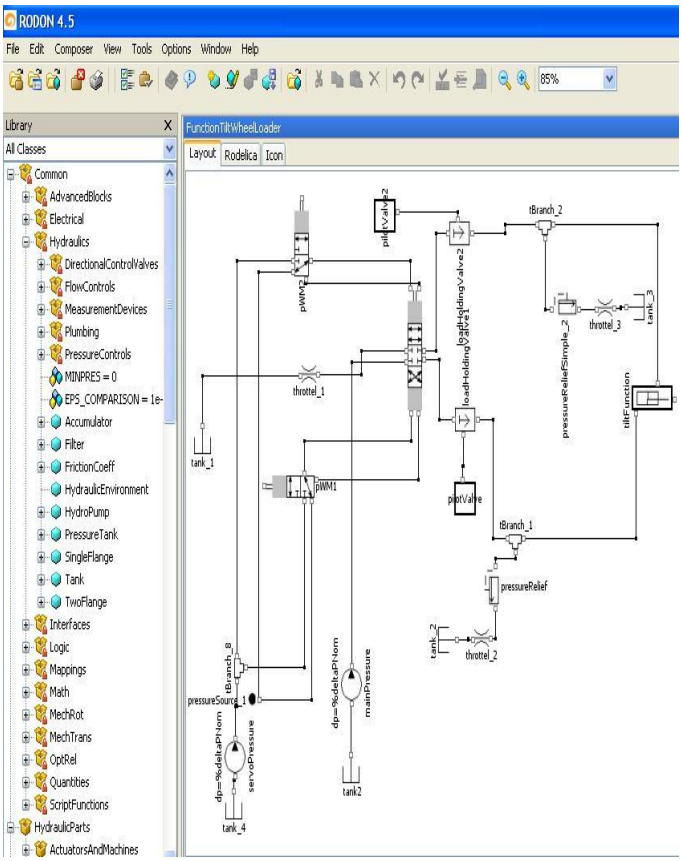

Figure 14, Tilt function in a wheel loader, modeled in RODON

Assume the diagnostics engineer got interest in finding all candidates that can explain performance problems within the tilting cylinder.

A symptom-variable is implemented, as explained in section (2.2), in this case:

- lowTransmissionPower (Boolean)
- tiltFunctionOutOfService (Boolean)

Conditions to satisfy the symptoms -->

lowTransmissionPower:=(tiltFunction.p1.Vdot>0 & tiltFunction.p1.p< 1.42) | (tiltFunction.p2.Vdot>0 & tiltFunction.p2.p< 1.95);

tiltFunctionOutOfService := (pWM1.PWMposNom == 0 & pWM2.PWMposNom == 0 & tiltFunction.isMoving != 0) |

(pWM1.PWMposNom == 1 & pWM2.PWMposNom == 0 & tiltFunction.isMoving != 2) |

(pWM1.PWMposNom == 0 & pWM2.PWMposNom == 1 & tiltFunction.isMoving != 1);

Based on the symptoms in the Rodelica Code; stated above, a decision tree has been computed (figure 15). The decision tree contains all the possible failure modes that can explain the faulty behavior; also it shows a guided diagnostic to ease maintenance work. Within the decision tree, the thought is to do the suggested tests. The final goal is to find the candidate on component level that can explain the debilitating behavior on system level.

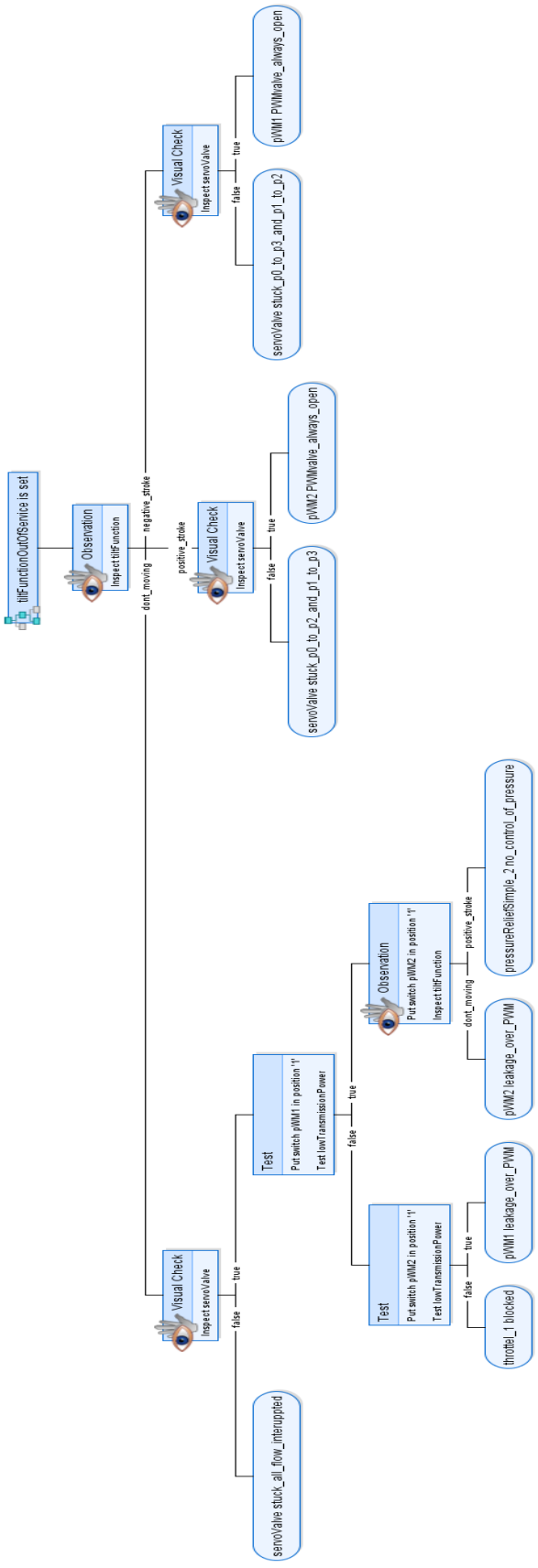

Figure 15, an auto generated Decision Tree for defect tilt-function

4 Conclusions

By the work of this paper; a component-library has been developed, for diagnostics use. It has been shown that the root cause for debilitating behavior can be found, through a model based diagnostic approach. By the end of the thesis, the expectations are that more sophisticated symptoms could be diagnosed; as well as more attributes should be implemented in the beta models, so that the decision trees will be more informative. Such additives could be an ECU. This work can be considered as a method evaluation, with the ultimate goal to find an artificial intelligence method for locate arbitrary failure modes in hydraulic machinery.

References

[1] *National Aeronautics and Space Administration (NASA), 2007, NASA Systems Engineering Handbook, Tech.Rep,* *http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301_2008008500.pdf*

[2] *Peter Bunus, Olle Isaksson, Beate Frey, Burkhard Munker (2009), RODON- A Model-Based Diagnosis Approach for the DX Diagnostic Competition, Uptime Solutions AB.*

[3] *The Modelica Association (2007). Modelica- a unified object-oriented language for physical systems modeling*

[4] Mattias Nyberg, Erik Frisk, 2012, Model Based Diagnosis of Technical Processes

[5] Lamb, J.D., Woodall, D.R. Asher, G.M (1997c), Bond graphs ii: Causality and singularity

[6] Johan de Kleer and James Kurien (2003). Fundamentals of Model-Based Diagnosis. In Proceedings of IFAC SafeProcess, Washington USA, June 2003