

Augmenting the diagnostic power of flow-based approaches to functional reasoning

Luca Chittaro and Roberto Ranon

Dipartimento di Matematica e Informatica
Università di Udine
Via delle Scienze 206, 33100 Udine, ITALY
chittaro@dimi.uniud.it

Abstract

In this paper, we consider flow-based approaches to functional diagnosis. First, we contrast the existing approaches, pointing out the major limitations of each. Then, we choose one of them and extend it in order to overcome the identified limitations. Finally, we show how the proposed extension can be introduced into the other flow-based approaches.

Introduction

Reasoning about function for diagnostic purposes has been recently investigated by several research groups (Chandrasekaran 1994; Chittaro 1995; Hawkins et al. 1994; Hunt, Pugh, & Price 1995; Kumar & Upadhyaya 1995; Larsson 1996). Nevertheless, a lot of work has still to be done on the functional diagnosis of real complex systems. In this paper, we take into consideration *flow-based approaches* (Chittaro 1995; Kumar & Upadhyaya 1995; Larsson 1996) to functional diagnosis. These approaches propose to model a system by focusing on the flows (of mass, energy, or information) in the system and on the actions performed by components on the considered flows. From a diagnostic point of view, they typically implement diagnosis as a search in a graph structure and claim to perform diagnostic reasoning very efficiently. In this paper, we initially show that this claim is achieved at the expense of diagnostic power. Indeed, each approach exhibits at least one of the following limitations: (i) easy availability of measurements is assumed, while in real-world cases measurements are often difficult to take or too expensive, or too unreliable, (ii) a single-fault assumption is adopted, and it is thus not possible to handle multiple faults, (iii) the modeling of interactions among different physical domains is not easy or impossible.

Since we are dealing with the application of flow-based techniques to a real-world problem in the domain of marine engineering (Chittaro, Fabbri & Lopez Cortes 1996), we need to overcome these limitations. To this purpose, this paper: (i) compares the existing flow-based diagnostic engines, pointing out the major limitations of each, (ii) chooses one of them (i.e., FDef (Chittaro 1995)) and extends it in order to overcome the identified limitations, and (iii) shows how the proposed extension can be introduced into the other flow-based approaches.

Comparing flow-based approaches

This section contrasts the three main flow-based approaches to functional diagnosis. The considered approaches are MFM (Multilevel Flow Modeling) (Larsson 1996), Classes (Kumar & Upadhyaya 1995), and FDef (Functional Diagnosis with efforts and flows) (Chittaro 1995).

Flow-based approaches: a short overview

Flow-based approaches to functional representation are founded on the general concept of *flow* (Paynter 1961). Some specific instances of flow are electrical current, mechanical velocity, hydraulic volume flow rate, and thermal heat flow. Some approaches also support the general notion of *effort* (Paynter 1961), i.e., the force responsible for the flow. Specific instances of effort are voltage, force, pressure, and temperature. In flow-based approaches, function is represented by means of a set of primitives, which are interpretations of actions frequently performed on the substances flowing through physical systems. These approaches generally aim at representing function in isolation, separating it from other types of knowledge, such as behavioral or teleological, in order to increase the clarity and the reusability of models.

MFM. In MFM, functions are expressed in terms of primitives such as source, sink, storage, transport, barrier and balance. Instances of these primitives are connected together to build *flow-structures*. Functions are linked to goals (i.e., purposes of the system) by two types of means-ends relations: *achieve* and *condition*. An achieve relation connects a set of functions to the goal they are meant to achieve. A condition relation connects a goal to a function: the goal must be fulfilled in order for the function to be available.

In the diagnostic algorithm proposed by (Larsson 1996), the user starts the diagnostic process by choosing an unachieved goal. The search proceeds downwards from the goal, via achieve relations, into the connected network of functional primitives, each of which has to be investigated (by questioning the user or by sensor readings) to find if the associated function is available or not. If a functional primitive conditioned by a goal is found to be at fault (or has no means of being checked), then the connected goal is recursively investigated; if it is found to be working, the

goal is skipped.

Classes. Classes represents function of a component in terms of the ports of the component, i.e., function is a relation between input and output of energy, matter or information. A set of functional primitives, called *classes* (producer, consumer, data, store, control, and address) is defined. Every causal flow in the system is called a *signal*; a *signal-line* is the sequence of components along the path from the origin to the use of a signal. Signal-lines can be of different types (power, clock, control, address, and data) with respect to the port of the component to which they provide input.

The diagnostic technique proposed by (Kumar & Upadhyaya 1995) starts from an incorrect system output S. Signal-lines that merge at output S are chosen for investigation, and ordered using heuristic criteria. Signal-lines are investigated following the order until a signal-line that contains a fault is found. Components in the currently investigated signal-line are ordered exploiting heuristic criteria, and tested in the obtained order until the faulty one is found. If a suspect component is connected to another signal-line, that signal-line is recursively investigated.

FDef. FDef (Chittaro 1995) adopts a limited version of the functional model proposed in Multimodeling (Chittaro et al. 1993), where functional primitives (called *roles*) are defined as interpretations of the physical equations describing the behavior of components. This interpretation is carried out using the Tetrahedron of State (TOS) (Paynter 1961), an abstract framework that describes a set of generalized equations which are common to a wide variety of physical domains. When the TOS is instantiated in a specific domain, the ordinary physical variables and equations are obtained. Functional roles are interpretations of the generalized equations of the TOS, and are of four types: generator, conduit, barrier, and reservoir.

The FDef diagnostic technique is based on the identification of the so called *enabling sets* and *disabling sets*. An enabling set is a set of roles which are all allowing the passage of flow or effort. A disabling set is a set of roles where there is at least one impediment to the passage of flow or effort. These sets are derived starting from the given measurements, using general axioms as those provided in (Chittaro 1995). They are then used both for exoneration purposes (identify normal roles), and to generate conflicts (i.e., sets of roles, each one containing at least a faulty role). A simple candidate generation algorithm (de Kleer & Williams 1987) uses the set of conflicts to produce the minimal diagnoses, and a minimum entropy (de Kleer & Williams 1987) prescription mechanism suggests the best measurement to discriminate among them.

Comparison

This section compares the three flow-based approaches in terms of the assumptions they make on the availability of measurements, their diagnostic output, and how they represent interactions among different physical domains.

Required availability of measurements. (Larsson 1996) uses questions (or sensor readings) in order to find if the currently investigated MFM functional primitive is faulty (or not), and in order to decide which further parts of the model have (or have not) to be investigated by the search algorithm. In order to guarantee the progress of the diagnostic algorithm, it is thus necessary that the highest number (possibly all) of the functions in the system is measurable (by diagnostic question or sensor readings). On the contrary, the scarce availability of measurements is a typical problem in real-world systems (e.g., because some of them are too costly, or too unreliable, or it is physically impossible to take them). In these cases, the diagnostic capability of the approach is impaired, leading to a partial, incomplete diagnosis or to a stuck reasoning process. It should also be noted that since MFM handles only flows, observations about efforts cannot be represented.

In Classes (Kumar & Upadhyaya 1995), components have to be testable in order to proceed in the investigation of signal-lines. This can lead again to a stuck reasoning process, when it is not possible to test some inputs and outputs of components. Unlike MFM, Classes tries to focus first on most probable diagnoses, by applying its heuristic ordering rules (e.g., a power signal-line has precedence over a control signal-line) to determine an order of investigation for signal-lines, and an order of testing for components inside the currently investigated signal-line.

FDef (Chittaro 1995) fully performs its diagnostic activity, regardless of the number of measurements given, returning the complete set of minimal diagnoses that are physically consistent with the given measurements. After generating this set, it ranks all the possible additional measurements from the most to the least informative, following an entropy-based strategy (de Kleer & Williams 1987). In this way, it aims at isolating the real diagnosis, using the least number of measurements.

Diagnostic output. The three approaches differ also in the type of diagnoses they produce. MFM produces just one diagnosis, including all the functions which have been measured to be faulty in the parts of model which have been explored. In order to help the user in the interpretation of this output, it classifies faulty functions into primary and secondary (the secondary could be an effect of the primary). Fault masking cases can thus be identified only if specific evidence is obtained, i.e., after a measurement pinpoints that a function involved in the masking is faulty.

Classes relies on the single-fault assumption, and produces an ordered set of single faults, which depend on the signal-line that has been currently reached by the investigation process. Multiple faults and fault masking cases are thus not covered.

FDef produces the set of all the minimal diagnoses that are physically consistent with the given measurements. It does not rely on the single-fault assumption, and each diagnosis is a minimal explanation of the observations. In this way, it covers also all the minimal fault masking cases consistent with the observations, without needing to obtain

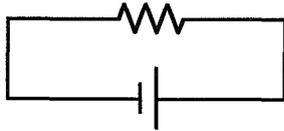


Figure 1. A simple circuit.

specific evidence first. However, producing this more detailed and complete diagnostic output causes FDef to be less efficient than MFM and Classes.

Representation of influences. All flow-based approaches aim at modeling separately the different flows in a system, by organizing the model into different networks (often called *flow-structures*) of functional primitives. Each flow-structure operates in a single physical domain (e.g., electrical, thermal,...). This modeling strategy is meant to allow: (i) the production of clear, easier to understand, models that modularly represent the different physical aspects of system functioning, and (ii) the focusing of reasoning on a selected physical domain. As a consequence, a component that works in more than one physical domain has to be represented by more than one functional primitive. For example, consider a simple circuit composed by a resistor connected to a battery, with the goal of producing heat (Figure 1). The resistor has a function both in the electrical domain (to conduce current), and in the thermal domain (to generate heat). In flow-based approaches, this system is represented by an electrical and a thermal flow-structure, each one including a function associated to the resistor. Since the two functions belong to the same physical component, the modeling approach should also provide a way to represent the relation between them. In general, we call *influence* the relation between two interacting primitives belonging to different flow-structures: the state of one of the two (called the *influencing* one) has the capability to influence the state of the other (called the *influenced* one). In the resistor example, the electrical function of the resistor influences its thermal function: an heat flow is generated by the resistor if and only if the resistor is conducting electrical flow.

MFM represents influences using means-ends relations. A condition relation connects a goal to the influenced function (i.e., the goal must be fulfilled in order for the function to be available). Then, the flow-structure containing the influencing function is connected to the goal

by an achieve relation. It is interesting to note that representation of influences in MFM requires to switch to the teleological level of representation, and then return to the functional one. Furthermore, from a practical point of view, the modeler is required to define a specific goal for every interaction he/she needs to model. An MFM model of the resistor circuit is shown in Figure 2(a). The diagnostic use of the condition relation in this example is the following: if the source associated to the resistor is measured to be malfunctioning, goal "keep electrical flow through resistor" is investigated and thus the electrical flow-structure is checked; if the source associated to the resistor is functioning, the goal is not investigated.

Classes represents a component that works in more than one physical domain by assigning it different functional primitives with respect to its inputs and outputs. Influences are implicitly represented in the model. From the Classes point of view (Figure 2(b)), the resistor has an electrical input, and a thermal output: it is a consumer (i.e., it consumes flow) with respect to the electrical input, and a producer (i.e., it produces flow) with respect to the thermal output. The interactions between the two primitives that represent the resistor are not explicitly expressed in the model. During diagnosis, if the resistor becomes suspect in one signal-line, then the other signal-line can be recursively investigated.

In Multimodeling (Chittaro et al. 1993), influences are defined as follows: a role FR_i, which refers to a physical equation PE_i, influences a role FR_j, which refers to a physical equation PE_j, if a physical variable of PE_i is (or concurs to determine) a parameter of PE_j. In the example, the conduit role associated to the resistor in the electrical domain influences the generator role associated to the resistor in the thermal domain. The resulting model is depicted in Figure 2(c), where the influence states that presence of flow in the electrical conduit is required to activate the thermal generator. Although Multimodeling considers influences, FDef does not support them. As a consequence, FDef can only diagnose a flow-structure instantiated in one physical domain.

Introducing influences in FDef

The previous analysis has shown that while FDef exhibits interesting diagnostic capabilities, it imposes an

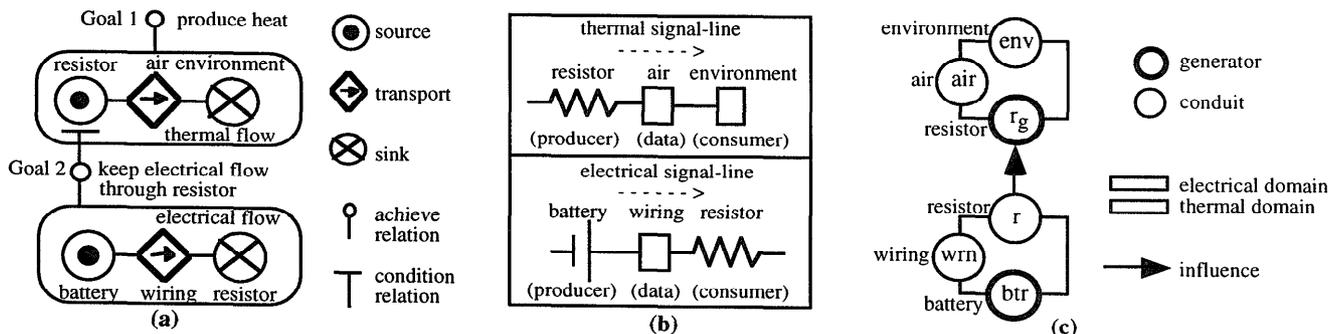


Figure 2. Models of the resistor system.

unacceptable restriction to a single physical domain. In this section, we extend FDef in order to remove that restriction, allowing a scaling up in the complexity of the functional models to be handled.

In addition to what is defined by Multimodeling, we further characterize influences as follows.

With *transduction influences*, the influenced role is a generator, and the state of the influencing role determines if the influenced generator is active. For example, the electrical conduit associated to the resistor causes (if it is traversed by current) the activation of the thermal generator associated to resistor.

With *regulation influences*, the influenced role is not a generator, and the state of the influencing role regulates the state of the influenced one. For example, the mechanical reservoir role associated to the screw of a tap regulates the passage of flow in the tap viewed as an hydraulic conduit.

The relay system case study

In the following, we consider a diagnostic example proposed by (Holtzblatt 1992), where the main component is a single pole, double throw relay. Holtzblatt presents two different cases: in the first, two components (both sensors) are connected to the relay; in the second, sensors are substituted with bulbs. In order to show how we handle both situations, we consider the case in Figure 3, where both a sensor (sns) and a bulb (b) are connected to the relay. The relay can work in two different states: an energized state (V_c is greater than a given threshold) in which current is allowed to flow only between P_{common} and sns, or a de-energized state (V_c is lower than the given threshold), in which current is allowed to flow only between P_{common} and b. Hereinafter, we suppose that three observations are given: (i) $V_c > \text{threshold}$ (the relay is thus expected to be in the energized state), (ii) sns is off, and (iii) b is lit.

Figure 4 depicts the functional representation of the relay example, considering the same domains taken into account by (Holtzblatt 1992).

Reasoning with influences

This section presents the extension of FDef, which is structured in three phases: (i) generation of Influence Assigners, (ii) application of influences, and (iii) generation of candidates. First, we characterize influences in more detail. Then, we provide a general treatment of the three

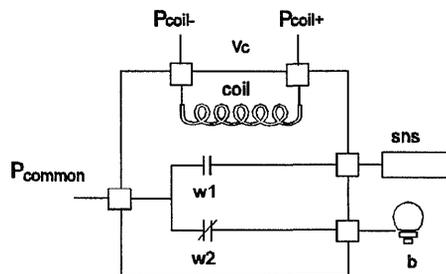


Figure 3. The relay system.

role	normal	abnormal
generator	produces effort and causes flow	does not allow passage of flow and effort, does not produce them
conduit	allows passage of flow and effort	does not allow passage of flow and effort
barrier	does not allow passage of flow and effort	allows passage of flow and effort

Table 1. Normal and abnormal functioning of roles.

role	positively influenced	negatively influenced
generator	unchanged	becomes normal barrier
conduit	unchanged	becomes barrier
barrier	unchanged	becomes conduit

Table 2. Functioning of influenced roles.

phases, also showing the results on the relay example for each of them.

Characterization of influences. For clarity purposes, Table 1 first summarizes what is assumed by FDef as normal and abnormal functioning of roles (e.g., in their normal functioning, conduits allow passage of both flow and effort). FDef also qualitatively characterizes the status of functional roles in this way: a role is *uncrossed* (*crossed*) if the flow associated to it is (is not) zero, a role is *unpushed* (*pushed*) if the effort associated to it is (is not) zero. Hereinafter, an influenced role is said to be *positively influenced* if the influencing role is crossed, or *negatively influenced*, if the influencing role is uncrossed. Table 2 characterizes the functioning of positively influenced and negatively influenced roles.

Generation of Influence Assigners. The diagnosis of systems with influences requires to consider different alternative situations: if the given observations do not allow to derive the status of an influencing role with respect to flow (crossed or uncrossed), the functioning of the influenced role is undetermined. For example, while the three given observations in the relay case study allow to conclude that influencing role b is crossed by electrical flow (because bulb b is lit), they do not allow to conclude anything about neither influencing role sns (the observation that the sensor is off does not mean that current is not flowing through it, because the sensor could be failing in

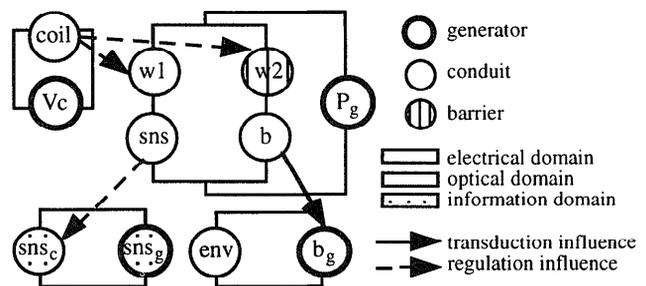


Figure 4. FDef model of the relay system.

communicating information) nor influencing role coil (the observation $V_c > \text{threshold}$ does not allow to conclude anything about current through the coil). In order to handle this, we introduce the concept of *Influence Assigner* (IA). An IA is a set of observations from which it is possible to univocally derive the status of every influencing role. When the set of observations currently given on the system is not an IA (i.e., it is not possible to derive the status of at least one influencing role), we need to consider multiple possibilities (the alternative would be to prescribe and take additional measurements until there are no more ambiguities, but this solution would lead to the first problem pointed out in the Comparison section). We thus build a set of IAs, by assuming additional observations concerning some undetermined influencing roles. More specifically, the algorithm that builds the set of IAs is the following (GivenObs denotes the set of observations given on the system, IAS the set of IAs produced by the algorithm, and the predicate $\text{obs}(\text{role}, \text{observation})$ describes observations on roles):

```

let IAS =  $\emptyset$ ;
let SetsOfObs = {GivenObs};
repeat
  foreach set of observations  $S \in \text{SetsOfObs}$  do
    derive all the consequences of the observations in  $S$  and
    let Undg be the set of undetermined influencing roles;
    if Undg =  $\emptyset$  then
      remove  $S$  from SetsOfObs;
      add  $S$  to IAS;
    endif
  enddo
  if SetsOfObs  $\neq \emptyset$  then
    foreach set of observations  $S \in \text{SetsOfObs}$  do
      remove  $S$  from SetsOfObs;
      choose  $r \in \text{Undg}$ ;
      add the set  $S \cup \{\text{obs}(r, \text{crossed})\}$  to SetsOfObs;
      add the set  $S \cup \{\text{obs}(r, \text{uncrossed})\}$  to SetsOfObs;
    enddo
  endif
until SetsOfObs =  $\emptyset$ .

```

Generation of IAs is obviously a possible source of combinatorial explosion, especially when very few measurements are given. However, the propagation activity (carried out both backward and forward before a new observation is assumed) typically allows to determine the status of a number of influencing roles which need not to be considered in the generation of assumptions, and also ensures that only physically feasible IAs are considered and generated. Moreover, inferences are cached and not performed more than once, e.g., when a set S is used (in the second part of the algorithm) to generate two sets that differ just for one assumption, they both inherit the inferences already performed with S .

Considering the relay example, GivenObs contains three elements: $\text{obs}(V_c, \text{pushed})$ (voltage is produced by generator V_c), $\text{obs}(\text{sns}_c, \text{uncrossed})$ (information from the sensor is

off), and $\text{obs}(\text{env}, \text{crossed})$ (light is flowing in the environment around the bulb). As discussed previously, these observations do not allow to determine the status of all the three influencing roles (coil, sns, and b), and thus they are not an IA. Only the status of b can be determined by propagation: since env is crossed, then generator b_g must be positively influenced, i.e., b is crossed. In this case, the following four IAs are generated:

V_c	coil	sns	sns_c	env
pushed	crossed	crossed	uncrossed	crossed
pushed	crossed	uncrossed	uncrossed	crossed
pushed	uncrossed	crossed	uncrossed	crossed
pushed	uncrossed	uncrossed	uncrossed	crossed

These four IAs are to be considered as different diagnostic situations, and thus handled disjunctively.

Application of influences. For each generated IA, the functional model is transformed according to the definitions of influences in Table 2 (e.g., a negatively influenced conduit is replaced by a barrier, positively influenced roles remain unchanged,...). For example, the application of the fourth generated IA to the model in Figure 4 results in three changes: conduit w_1 becomes a barrier, barrier w_2 becomes a conduit, and conduit sns_c becomes a barrier.

Generation of candidates. Generation of candidates for an IA and the corresponding model (i.e., the result of the transformation described above) is performed as follows.

First, generation of local minimal candidates is performed for each flow-structure in the model, by locally running the plain FDef engine (as described in (Chittaro 1995)) only on that single flow-structure, feeding it with the observations (contained in the considered IA) concerning roles of that flow-structure. An interesting feature of this procedure is that it focuses diagnostic reasoning only on small sets of components (i.e., those belonging to the currently considered flow-structure). In the case of the fourth generated IA, locally running FDef only on the electrical flow-structure composed by V_c and coil produces the enabling set $\{V_c\}$ and the disabling set $\{V_c, \text{coil}\}$. The minimal candidate for this flow-structure is then $\{\text{coil}\}$, while the local consideration of the other flow-structures does not produce any conflict (and thus no candidates): the coil is faulty, and the remaining part of the relay behaves as expected (the relay is actually in the de-energized state).

Second, global candidates for the currently considered IA are simply obtained by cartesian product of the sets of local candidates generated for the single flow-structures (for the fourth IA, $\{\text{coil}\}$ is thus the only minimal candidate). The generation of consistent global candidates is guaranteed, because the IA and the model transformation ensure that the flow-structures in the model are in a mutually consistent state, and thus the local candidates are also mutually consistent and can be globally combined.

The two activities above are performed for each generated IA, and then the complete set of candidates is obtained as the union of the sets of global minimal candidates obtained

with the different IAs. In the relay example, the complete set of minimal candidates generated is $\{\{coil\}, \{w2, w1\}, \{w2, sns\}, \{w2, sns_c\}, \{w2, sns_g\}\}$.

In order to speed up generation, the results of locally running FDef on a single flow-structure for an IA are saved, avoiding the need of repeating the computation with other IAs that make the same assumptions on that flow-structure.

Our extension of FDef preserves the entropy-based mechanism for test prescription. In the relay example, measuring the flow associated to role coil (or, alternatively, to generator Vc) is the suggested best measurement.

Extending other flow-based approaches

This section provides guidelines to implement the extended FDef reasoning strategy inside the other flow-based approaches. Firstly, enabling sets and disabling sets have to be introduced in the considered approach. FDef uses axioms for the derivation of enabling and disabling sets from observations on flows and efforts (Chittaro 1995; Chittaro, Fabbri & Lopez Cortes 1996). Their adaptation to MFM and Classes is relatively straightforward. For example, reformulating FDef axioms in a more general context, we obtain statements such as: "if in a circuit of functional primitives, we are given at least an observation of presence of flow, and no observations about absence of flow or effort, then that circuit is an enabling set", or "if in a circuit of functional primitives, we are given at least an observation of absence of flow, and no observations about absence of effort, then that circuit is a disabling set". Consider for example the MFM flow-structure in Figure 2(a), representing an electrical circuit made of battery, wiring and a resistor, and suppose to observe current flowing in the resistor. In this case, the first of the two rules mentioned above would conclude that the battery, wiring and resistor allow the passage of flow, i.e., they constitute an enabling set. On the contrary, if absence of current were observed in the resistor, the second rule would conclude that there is at least a component in the circuit that does not allow the passage of flow, i.e., they are a disabling set. The Classes case (Figure 2(b)) is analogous.

The second step is the identification of influences in MFM and Classes. To do this, the following approach can be followed. In MFM, every condition relation can be considered as a transduction influence, that connects two different functions representing the same component in two flow-structures (e.g., the sink and the source associated to the resistor in Figure 2(a)). In Classes, influences can be introduced when a component has ports belonging to different physical domains. For example, in Figure 2(b), the resistor is represented by a consumer class in the electrical signal-line, and a producer class in the thermal signal-line, and thus they can be connected by an influence.

Once the above described adaptations have been carried out, it is straightforward to run the extended FDef engine described in this paper on MFM and Classes models, thus overcoming the shown limitations of these approaches at the expense of some efficiency.

Conclusions

This paper presented (i) an evaluation of the diagnostic power of existing flow-based diagnostic engines, (ii) a relevant and useful extension of the FDef diagnostic engine, and (iii) guidelines to implement the features of extended FDef in the other flow-based approaches.

The techniques presented in this paper are being used on the technical marine system application presented in (Chittaro, Fabbri & Lopez Cortes 1996), where they are allowing us to move from the diagnosis of the considered hydraulic system to the diagnosis of the whole set of subsystems connected to it. The evaluation of the results on this application is pointing out that the approach is good at isolating faults when they result in a loss of functionality. Since some faults in the considered domain are preceded by a slow degradation in performance before turning into a loss of functionality, one of the subjects we are considering is the introduction and exploitation of "too low"/"too high" flow and effort observations in flow-based diagnostic approaches.

References

- Chandrasekaran B. 1994. Functional Representation and Causal Processes. *Advances in Computers* 38:73-143.
- Chittaro L.; Guida G.; Tasso C. and Toppano E. 1993. Functional and Teleological Knowledge in the Multimodeling Approach for Reasoning About Physical Systems: A Case Study in Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6): 1718-1751.
- Chittaro L. 1995. Functional Diagnosis and Prescription of Measurements Using Effort and Flow Variables. *IEE Control Theory and Applications*, 142(5): 420-432.
- Chittaro L.; Fabbri R. and Lopez Cortes J. 1996. Functional Diagnosis Goes to the Sea: Applying FDef to the Heavy Fuel Oil Transfer System of a Ship. In *Proceedings of the Ninth Florida AI Research Symposium (FLAIRS)*, Key West, FL.
- Hawkins R.; Sticklen J.; McDowell J.K.; Hill T. and Boyer R. 1994. Function-based Modeling and Troubleshooting. *Journal of Applied Artificial Intelligence* 8(2): 285-302.
- Holtzblatt, L.J. 1992. Diagnosing Multiple Failures Using Knowledge of Component States. In W. Hamscher, L. Console, J. de Kleer (eds.), *Readings in Model-based Diagnosis*, San Mateo, Calif: Morgan Kaufmann, 165-169.
- Hunt J.; Pugh D. and Price C. 1995. Failure Mode Effects Analysis: a Practical Application of Functional Modeling. *Journal of Applied Artificial Intelligence* 9(1): 33-44.
- de Kleer J. and Williams B.C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence* 32: 97-130.
- Kumar A. N. and Upadhyaya S.J. 1995. Function Based Discrimination during Model-based Diagnosis. *Journal of Applied Artificial Intelligence* 9(1): 65-80.
- Larsson J.E. 1996. Diagnosis based on explicit means-end models. *Artificial Intelligence* 80: 29-93.
- Paynter H.M. 1961. *Analysis and Design of Engineering Systems*. Cambridge, Mass.: MIT Press.