

Software Architecture For Diagnosing Physical Systems

Samir Touaf – Stéphane Ploix – Jean-Marie Flaus

Laboratoire d'Automatique de Grenoble, INPG, UJF, UMR 5528,

BP 46, F-38402 Saint Martin d'Hères Cedex, France

Phone: 33 4 76 82 62 44, Fax: 33 4 76 82 63 88

e-mail : Samir.Touaf@inpg.fr, Stephane.Ploix@inpg.fr, Jean-Marie Flaus@inpg.fr

Abstract: This paper discusses about a general software architecture suitable for the design of diagnostic systems based on logical reasoning. This type of architecture should be able to support different kinds of consistency tests and different kinds of triggering strategies. Different steps, such as data acquisition from a physical system, batch or online consistency tests, diagnostic reasoning and management of consistency tests have been isolated. Software dedicated to the diagnosis of a system of two cascading water tanks is also presented as an illustration.

Keys-words: Fault detection and isolation, Fault Diagnosis reasoning, Static and dynamic systems, software architecture, Diagnostic strategy.

1 Introduction

Fault diagnosis systems become very complex when the physical systems to be diagnosed are no longer elementary. To manage the diagnosis of complex dynamic and static systems, general concepts have to be developed in order to define a general architecture suitable for any software implementation. The proposed architecture relies on the object-oriented paradigm because of its main advantages in developing robust software in the shortest time possible, in re-using existing software wherever possible (Rumbaugh, 1990) and therefore in promoting software maintainability.

An object-oriented software architecture has to be as natural as possible i.e. each object has to be clearly identified in terms of semantics, role and responsibility. The following main object-types have been identified:

- The data provider. Its role is to provide present or past data from the actual physical system on request.

- The consistency test. Its role is to provide symptoms by checking if the data coming from a physical sub-system is consistent with a reference behavior related to a given state (which may contain normal assumptions and specific fault assumptions). Note that a reference behavior may be composed of different kinds of models, in particular parity relations or state observers.

The diagnostic machine. Its role is to analyze symptoms so that diagnoses i.e. possible states of the actual physical systems can be inferred.

The test manager. Its role is to manage the whole diagnosis process. For example, what consistency tests should be triggered ? and When? When does the diagnosis machine need to analyze symptoms? and so on.

This decomposition is unusual both in the Automatic Control community and in the Artificial Intelligence community. In the Automatic Control community, most of the algorithms combine all

these roles. For example, detection and isolation (consistency tests and symptom analysis) are performed at the same time when using a bank of state observers (Franck, 1990). In the AI community, the logical reasoning used for analyzing symptoms does not consider that there are many ways to perform a consistency test and that there are also many ways to schedule consistency tests. Moreover, each community uses its own method for analyzing symptoms: the logical reasoning developed by the AI community may yield several plausible diagnoses, sometimes an exoneration assumption is considered (Dague,2001) sometimes not (Leyval, 1994), the FDI reasoning developed by the AC community usually assumes that the number of possible faults is finite and that all of them are modeled by simple additive terms. It is not our purpose here to compare these approaches: interesting elements for comparison can be found in (Cordier et al,2000) but to point out that there are several ways of performing consistency tests, and there are also several ways of analyzing symptoms, managing a diagnostic system and acquiring data from a physical system.

The first section of this paper describes in detail general software architecture which can handle all the different diagnostic procedures. The second section deals with the management of a diagnosis system, particularly with the management of consistency tests. The third section presents a diagnostic system dedicated to a water tank system, which has been developed in Java from the general architecture.

2 A general software architecture for diagnosis systems

Diagnosis of a physical plant can be seen as a cyclical process. Figure 1 shows the different tasks, which have to be handled during each cycle. According to the architecture presented in the introduction, each object-type has the responsibility of carrying out one of these tasks. This kind of design is very robust because each object may extend its responsibility to new contexts or use cases without any repercussion on the rest of the software, provided that the interface between each object-type is fixed.

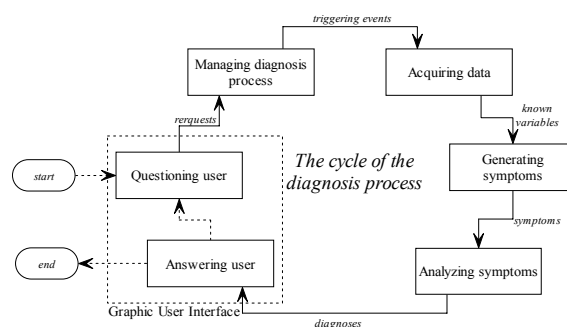


Figure 1 – Tasks in a diagnosis process

The general characteristics of the different main object-types will be studied in the following sub-sections.

2.1 Data provider object-type

Its responsibility is to provide data concerning known physical phenomena to other objects when requested. Different kinds of requests may arise depending on the context, which may be the real-time context of the diagnostic system or a batch context. This type of object may be able to provide the last updated value as well as a set of consecutive values related to a given known phenomenon. Each object should be dedicated to a given phenomenon and the way it gets information is an internal private part hidden from the other objects: information may be provided for instance by a sensor, a controller for controlled variables, a database or an operator. Nevertheless these interfaces, i.e. the kinds of requests from the other objects, need to be precisely defined. The objects called *Consistency tests* have to interact with the *data providers*.

2.2 Consistency test object-type

The responsibility of this object-type is to provide symptoms to the diagnosis machine. This generation is based on consistency tests that check if the actual behaviors are consistent with reference models. For the same subsystem, many possible consistency tests based on different techniques (parity relation, state observer,... for analytical models) may exist.

A test should be based on a set of behavioral models, on a validity condition, which determines when the results of the test are valid and a set of assumptions on the states of the components involved. For example, a consistency test may yield any one of the following results:

- The result is consistent
- The result is inconsistent
- There is no result because the test is invalid in the present context.

2.3 Diagnosis machine object-type

The responsibility of this object-type is to analyze the available symptoms provided by consistency tests in order to determine the possible abnormal states. Any kind of diagnostic reasoning may be possible, such as reasoning based on signature tables (Gertler,1988), causality (Montmain,2000), logic (Reiter,1987), (de Kleer,1987),... However, all the diagnostic reasoning procedure are not equivalent. They differ from implicit assumptions.

A reasoning based on a signature table assumes that there is no fault when nothing is detected and that the system to be diagnosed is never affected by unmodeled faults. The causal reasoning only assume that there is no fault when nothing is detected. Finally, the logical reasoning has no implicit assumptions.

Whatever the way of reasoning, to analyze symptoms, information on the component states checked by each test is required. For example, a test to check the value of an electrical resistor may be based on the assumption: the resistor state, the current sensor state and the voltage sensor state are normal. Using the circumscription theory formalism (Mac Carthy,1986), it may be written as:

$$\neg AB(resistor) \wedge \neg AB(current\ sensor) \dots \\ \wedge \neg AB(voltage\ sensor)$$

where $\neg AB()$ means *not abnormal*.

These expressions qualify the symptoms; they provide a meaning to the test results: if the previous resistor test provides an inconsistency, it means that the related test assumption is false:

$$AB(resistor) \vee AB(current\ sensor) \dots \\ \vee AB(voltage\ sensor)$$

Namely, the state of the resistor is abnormal or/and the state of the current sensor is abnormal or/and the state of the voltage sensor is abnormal.

2.4 Diagnostic manager object-type

The responsibility of this object-type is to manage the diagnostic process i.e. to trigger the other objects with respect to the user requests. It has to be able to interpret the user requests in order for selecting a relevant triggering strategy. It may trigger data providers, consistency tests and to select a relevant diagnostic reasoning.

The next section is dedicated to the diagnosis manager. It shows how selecting of a strategy relevant to a user request.

3 Diagnostic Strategy

Developing a strategy for a diagnostic system mainly consists in selecting a triggering mechanism of consistency tests. Decomposing a strategy into two steps is natural as for medical diagnosis. The first step corresponds to the analysis of symptoms provided by tests exclusively based on knowledge of normal behavior. The second step relies on tests assuming specific abnormal behaviors. It is indeed natural to check what is behaving normally before looking for specific faults. It is a matter of time. The number of specific tests may be very large and many of them usually become irrelevant after considering the results of the Normal Behavior Oriented (NBO) tests.

3.1 Normal-behavior oriented strategy

Two strategic steps are usually distinguished during a normal-behavior oriented diagnostic procedure. The first one is based on permanent tests, which are the tests of level 0, and the other one is composed of tests of level 1 triggered by inconsistencies detected by these permanent tests. Even if this rare, higher levels may also exist (see strategy 4 in figure 4).

An example of system architecture is presented in figure 3. Physical variables are represented by empty circles, known variables by full circles, and models by solid lines. There are tree possible NBO tests: these are represented by dotted lines.

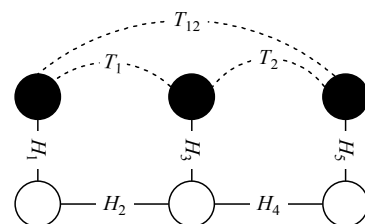


Figure 3 – Model architecture and consistency tests

The number of possible NBO tests may be very large. For simple architectures as shown in figure 2, the number of possible tests for normal behavior is equal to C_n^2 where n is the number of known variables. Therefore, not all the tests are performed in each case: this is where the diagnostic strategy comes into play.

Note that the diagnostic strategy must be defined separately from the diagnosis reasoning because different ways of reasoning may be implemented in the same diagnosis system. Therefore, the tools used for designing strategy have to be independent of the symptom analysis.

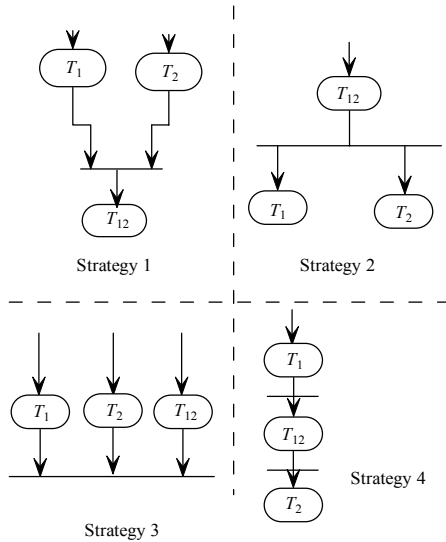


Figure 4 – Some diagnostic strategy

For the simple example of figure 3, there are many possibility shown in (see figure 4).

The problem is to select a most relevant strategy. Estimators are necessary to solve this problem.

The first proposed estimator will be called Normal Cycle Time (*NCT*). This is the number of consistency tests performed when the diagnosed system is free of faults. This is an important estimator because it points out the typical time consumption for normal behavior.

The results for the 4 strategies presented in figure 4 are shown in the following table.

Table 1 – Normal Cycle Time

Strategy	1	2	3	4
NCT	2	1	3	1

In the absence of faults, the strategies 2 and 4 consume less time than the others. The 3rd one is the slowest.

The detection ability must now be estimated. A Detection-per-Level Rate can then be defined. At level 0, the DLR_0 corresponds to the ratio between the number of detectable abnormal states and the number of possible abnormal states. At level $l > 0$, DLR_l corresponds to the ratio between the number of abnormal states detectable by the tests of level l and the number of abnormal states that would have been detected if these tests were at level 0.

The following table summarizes the links between the tests and the states of the system to be diagnosed:

Table 2 – Composition of NBO tests

	H_1	H_2	H_3	H_4	H_5
T_1	1	1	1		
T_2			1	1	1
T_{12}	1	1		1	1

Let's consider in details the 2nd strategy. Test T_{12} detects faults satisfying: $\neg H_1 \vee \neg H_2 \vee \neg H_4 \vee \neg H_5$ (H_1 is not satisfied or...). Only one abnormal state is not detected: $H_1 \wedge H_2 \wedge \neg H_3 \wedge H_4 \wedge H_5$. The DLR_0 is equal to 30/31 (they are 2^5-1 abnormal states). The DLR_1 is unchanged because at level 1, the detected faults satisfied:

$$\begin{aligned} & (\neg H_1 \vee \neg H_2 \vee \neg H_4 \vee \neg H_5) \wedge \\ & ((\neg H_1 \vee \neg H_2 \vee \neg H_3) \vee (\neg H_3 \vee \neg H_4 \vee \neg H_5)) = \\ & = (\neg H_1 \vee \neg H_2 \vee \neg H_3 \vee \neg H_4) \end{aligned}$$

and because tests T_1 and T_2 would detect 31 abnormal states if they were at level 0.

The following table summarizes the results for the different strategies:

Table 3 – Detection per Level Rate

Strategy	DLR_0	DLR_1	DLR_2
1	31/31	30/30	-
2	30/31	30/31	-
3	31/31	-	-
4	28/31	27/30	24/28

Level 0 provides the most important results. In strategies 1 and 3, all the abnormal states will be detected. In strategy number 2, one fault on H_3 will not be detected. In strategy 4, 3 abnormal states will never be detected.

DLR for other levels provide information on the detection ability of other level. If the scores are high, it means that the strategy does not decrease the test detection abilities.

For example, in the 4th strategy, there are only 27 abnormal states that may be detected at level 1 whereas 30 abnormal states could be detected if test T_{12} was performed at level 0.

Let's consider now the robustness of the diagnostic strategy. 3rd Strategy is for instance more robust than the 1st one. If there is a large uncertainty on the behavior of the component related to H_3 , each test containing H_3 , namely tests T_1 and T_2 , will be imprecise and will not be able to detected anything.

In order to estimate the robustness, the DLR_l are recalculated in successively taking into account large uncertainties on the components related to the H_i .

Let U_i denote a large uncertainty on the component related to H_i . Recalculating the DLR_0 for the 2nd strategy when a large uncertainty is present on H_1 leads to:

$$DLR_0 = (-H_1 \vee -H_2 \vee -H_4 \vee -H_5) \dots \\ \dots \wedge -I_1 \wedge -I_2 \wedge -I_4 \wedge -I_5 = 0$$

In the same way, if the component related to H_2 is uncertain i.e. I_2 is true, $DLR_0=0$; if I_3 is true, $DLR_0=30/31$; if I_4 is true, $DLR_0=0$ and if I_5 is true, $DLR_0=0$. The average value of all these DLR_0 is the Detection per Level 0 Rate with Uncertainties $DLRU_0$. For the second strategy, the following result is obtained:

$$DLRU_0 = \frac{0+0+30+0+0}{5} = 6$$

In the same way, the average values on all the different uncertain H_i are then evaluated. The Detection per Level Rate with Uncertainties ($DLRU_i$) are thus obtained:

Table 4 – Detection per Level Rate with Uncertainties

Strategy	DLRU ₀	DLRU ₁	DLRU ₂
1	22.4/31	0/30	-
2	6/31	0/31	-
3	28.4/31	-	-
4	11.2/31	0/30	0/28

In a general case, each test related to the assumptions H_i where $i \in \Omega$ will detect the abnormal states satisfying the following expression:

$$\bigvee_{i \in \Omega} (-H_i) \wedge \bigwedge_{i \in \Omega} (-I_i)$$

DLR_i and $DLRU_i$ result from the number of abnormal states satisfying conjunctions and/or disjunctions of these kinds of expressions, depending on the test triggering scheme.

The 2nd and 4th strategies are the quickest ones however both do not detect all the abnormal states. The 2nd one has better detection ability however, the 4th strategy, which has been designed for being a dummy strategy, has a better average detection rate in presence of large uncertainties.

The 1st strategy is slower but it detects any abnormal states. The 4th strategy is the slowest; it also detects any abnormal states but it has better results than the 4th one when large uncertainties are present.

Notice that the distribution of uncertainties is known, the average leading to the $DLRU_i$ can be reduced to a given set of large uncertainties.

Computing these performance estimators is easy to generalized to any strategy in formalizing the logical expression based on tests assumptions and then in counting the abnormal states satisfying these expressions.

Thanks to these estimators, if an user describes a diagnostic request in terms of time speed, expected diagnosis accuracy and expected robustness, the 3 proposed estimators are tools to select the best strategy.

3.2 Abnormal-behavior oriented strategy

In a diagnostic system, tests relying on models of faulty behaviors are generally available. All these tests should be triggered because some of them may be incompatible the results of NBO tests. A strategic point is to define triggering rules.

The composition of Abnormal Behavior Oriented (ABO) tests may be described by a table likely as table 2. Abnormal state assumptions appear of course. To point out the relationships between the normal and abnormal state assumptions about a same component, references to the related components have been added. For instance, normal assumption H_1 related to component C_1 will be written $H_1(C_1)$. Table 5 gathers assumptions on same components.

Table 5 – Assumptions and abnormal behavior oriented tests

	$H_1(C_1)$	$F_1(C_1)$	$H_2(C_2)$	$H_3(C_3)$	$F_1(C_3)$	$F_2(C_3)$	$H_4(C_4)$	$H_5(C_5)$
T_4		1	1	1			1	1
T_5	1		1		1		1	1
T_6	1		1			1	1	1
T_7		1	1		1		1	1

As an illustration, if T_{12} is false, the following expression is true: $\neg H_1(C_1) \vee \neg H_2(C_2) \vee \neg H_4(C_4) \vee \neg H_5(C_5)$. Then all the ABO tests, which are compatible with this expression, should be triggered. Test T_4 contains the assumption $F_1(C_1) \leftarrow \neg H_1(C_1)$. Tests T_7 contains also $F_1(C_1) \leftarrow \neg H_1(C_1)$. Consequently, in this case, only tests T_4 and T_7 has to be triggered because they may complete the results of the NBO tests.

In a general way, a ANO test is compatible with a NBO test described by the following expression:

$$\bigvee_{i \in \Omega} \neg H_i(C_i)$$

if it contains at least one fault assumption $F_j(C_j)$ satisfying $j \in \Omega$.

Any test compatible with at least one false NBO test should be triggered because it may bring more information on the actual state of the system to be diagnosed.

4 Application of diagnostic reasoning

An application concerning 2 water tanks is presented here as an illustration. Thanks to a digital-to-analog converter, a Kammer gate controls an input water flow which pours into an upper tank. At the same time the upper tank is filling, the water it contains can flow through an underneath restriction into a lower tank. The water contained in the tank can also flow by an underneath restriction to be drained away. The water height in upper tank (denoted by x_1) and in lower tank (denoted by x_2) is measured thanks to precise condenser sensors. A digital controller controls this physical system. The different components of the system are :

C_1 : Kammer gate, C_2 : Uper tank, C_3 : Upper tank restriction, C_4 : Lower tank, C_5 : Lower tank restriction, C_6 : DA converter, C_7 : x1 sensor, C_8 : x2 sensor.

Components C_6 , C_7 , and C_8 are information components whose models are generally very simple but which must be made apparent.

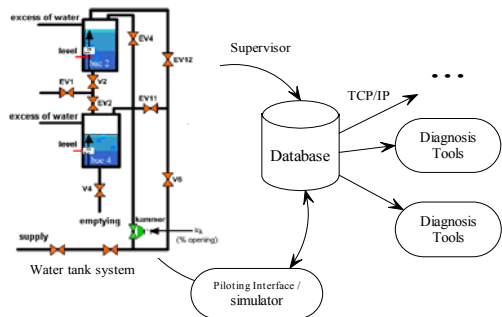


Figure 5 – The diagnostic system

Figure 5 illustrate that several users can start up one tools of diagnosis on the same water tank system, as being distributed on different machines thanks to environments of Corba type or RMI. A distributed data base and one simulator interfacing that substitutes the water tank system are present since the system of analysis diagnosis is not supposed to function in line but from data recorded in the data base.

A collaboration diagram of figure 6 is an interaction diagram which shows the sequence of messages that implement an operation or a transaction. These diagrams show objects, their links, and their messages. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

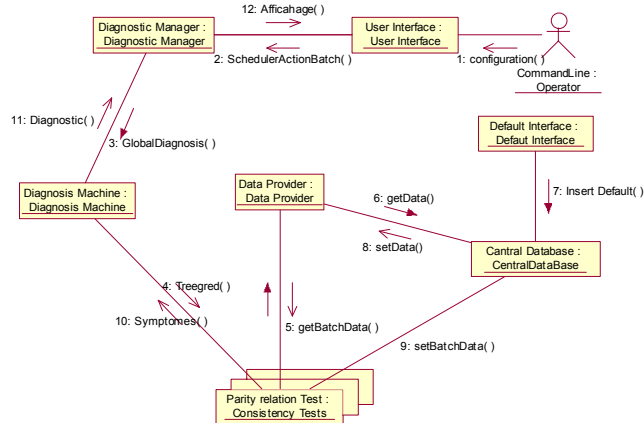


Figure 6 : Collaboration diagram

Diagnostic analyses can be made in two different working modes : Batch mode or Real Time mode.

The Batch mode allows the operator to trigger a diagnosis during a prescribed time period. The specification of parameters of the beginning and the end of the temporal window this fact at the time of system configuration.

In real time mode, The tool of diagnosis must be in the capacity to achieve the diagnosis of the water tank system while taking the necessary data to flown it.

For our application of the water tank system, tree strategies are possible (see figure 7). The choice of the most relevant strategy depends on measures defined in the §3. Thanks to these estimators, the user describes a diagnostic request in terms of time speed, robustness and expected diagnosis accuracy.

Table 6 gathers different NBO and ABO tests for the water tank system and their assumptions on the same components. A component C_i behaving normally will be described by $h_i = \neg AB(C_j)$. If this component is altered by a fault D , it will be written as follows: $h_i = D(C_j)$.

Table 6 - Composition of NBO and ABO tests

	$\neg AB(C_1)$	$\neg AB(C_2)$	$F(C_2)$	$\neg AB(C_3)$	$\neg AB(C_4)$	$\neg AB(C_5)$	$\neg AB(C_6)$	$\neg AB(C_7)$	$\neg AB(C_8)$
T_1	1	1		1			1	1	
T_3				1	1	1		1	1
T_4	1	1		1	1	1	1		1
T_2	1		1	1			1	1	
T_5	1		1	1	1	1	1		1

Tests T_1 , T_3 and T_4 are tests of normal behavior. There are also abnormal behavior test called T_2 and T_5 . These tests refer to fault assumption $F(C_2)$ over the component C_2 .

Table 7 – Performance estimators

	NCT	DLR ₀	DLR ₁	DLRU ₀	DLRU ₁
Strategy 1	2	255/255	254/254	186/255	0/255
Strategy 2	1	254/255	254/255	31.75/255	0/255
Strategy 3	3	255/255	-	217,75/255	-

The DLR_0 determines the strategies providing the maximum detection ability. Strategies 1 and 3 allow the detection of all abnormal states of the tank systems. Otherwise, the fault over the component C_7 is not detected by strategy 2.

The DLR_1 confirms that strategy 1 is better than strategies 2 and 3 in the absence of faults, because this approach enables the detection of other faulty components of the system.

The 2nd strategy is the quickest one. However, it does not detect all the abnormal states, the 3rd one has better detection ability. The 3rd strategy has a better average detection rate in presence of large uncertainties than the 2nd one. The 3rd strategy is slower than both 1st and 2nd strategies but it detects any abnormal states and it has better results than both 2nd and 1st when large uncertainties are present.

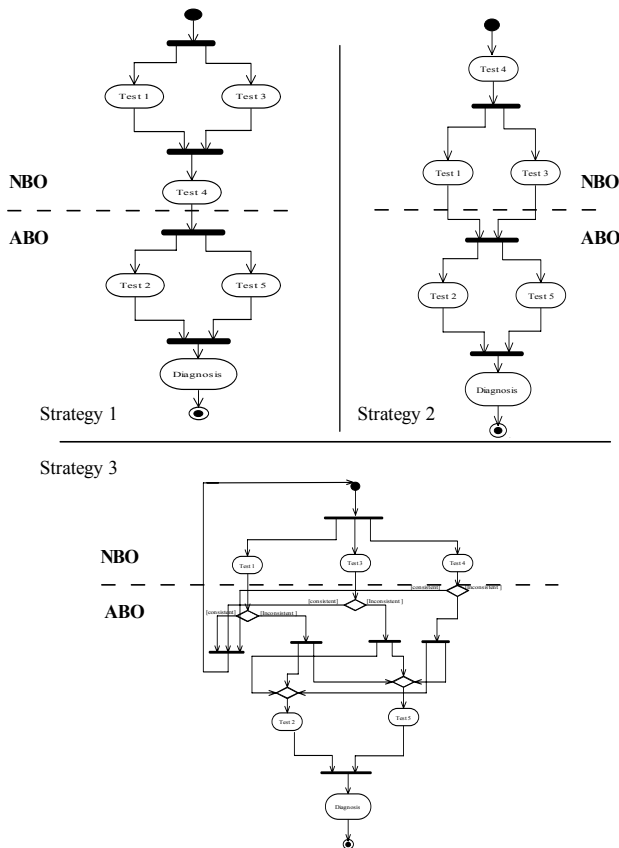


Figure 7 - Different diagnosis strategy

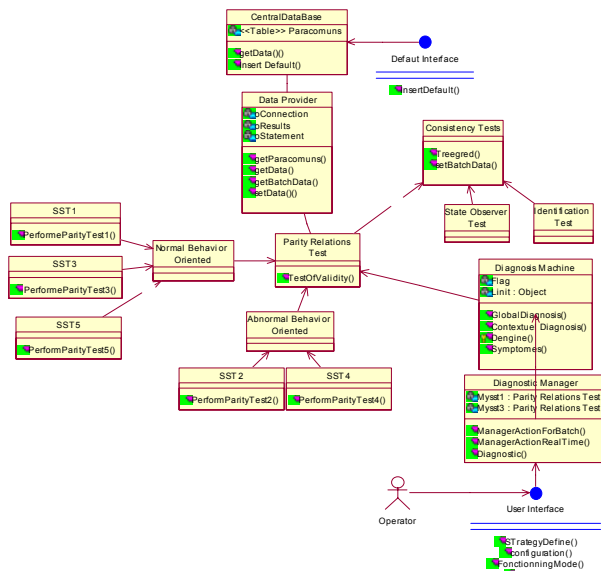


Figure 8 - Class diagram of the Diagnostic system

The question of when re-initializing a test, is a strategic problem to solve. When a test has led to an inconsistency, even if no longer symptoms are detected, a fault may be still present. Then, this test should memorize the inconsistency until a re-initialization. Re-initialization event depends on the nature of the fault that has occurred. Some faults may be permanent. When they occur, the system state can not be changed by itself. An operator intervention is

required. In this case, the operator himself will re-initialize tests. Some other faults may be temporary. In this case, tests may be automatically re-initialized after a relevant time period where no longer symptom would be detected.

The re-initialization problem has to be distinguished from the false alarm problem. When an event is detected by mistake then that is a false alarm. It has to be considered as an unacceptable event if reliability is expected. It means that the reference model is false and therefore that it has to be improved by re-identification or by reducing its validity domain.

Finally, an UML class diagram, which is suitable to the design of object-oriented software, has been used in figure 8 to summarize the overall architecture of a diagnostic system.

5 Conclusion

A general software architecture suitable for the design of diagnostic systems has been presented. This architecture is able to support different kinds of consistency tests and different kinds of diagnostic process management strategies. The proposed software architecture, which can handle all the different diagnostic procedures, can be considered as an 'open' model, a minimal class proposition is made permitting by successive sophistications to add functionalities. Another aspect deals with the management of complete diagnostic systems, particularly with the management of consistency tests. The problem of selecting the most relevant strategy has been solved thanks to performance estimators. The design of software based on this kind of architecture dedicated to a water tank system has shown the applicability of the approach.

6 References

Cordier, M.-O., P. Dague, M. Dumas, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-massuyès (2000). A comparative analysis of AI and control theory approaches to model-based diagnosis. *14th European Conference on Artificial Intelligence*, Berlin, Germany.

Dague, P. (2001). Théorie logique du diagnostic à base de modèles. in: *Diagnostic, Intelligence artificielle et reconnaissance de formes*, B. Dubuisson, ed., 17-104, Hermès, Paris.

De Kleer, J., and B. C. Williams (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32, pp. 97-130.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - A survey and some new results. *Automatica*, 26(3), pp. 459-471.

Gertler, J. (1988). Survey of Model-Based failure detection and isolation in complex plants. *IEEE Control Systems Magazine*, 11, pp. 3-11.

Leyval L., S. Gentil and S. Feray-Beaumont (1994). Model based causal reasoning for process supervision, *Automatica*, 30 (8), pp. 1295-1306.

McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28, pp. 89-116.

Montmain J., S. Gentil (2000) Dynamic causal model diagnostic reasoning for on-line technical process supervision, *Automatica* 36, 1137-1152.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorenson (1990). Object-oriented modelling and design, eds. Prentice Hall.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32, pp. 57-95.