



# Approche originale utilisant le Raisonnement à Partir de Cas pour le Diagnostic en Ligne des Systèmes Automatisés de Production

N. Ben Rabah, Ramla Saddem, Faten Ben Hmida, Véronique Carré-Ménétrier, Moncef Tagina

## ► To cite this version:

N. Ben Rabah, Ramla Saddem, Faten Ben Hmida, Véronique Carré-Ménétrier, Moncef Tagina. Approche originale utilisant le Raisonnement à Partir de Cas pour le Diagnostic en Ligne des Systèmes Automatisés de Production. Colloque sur la Modélisation des Systèmes Réactifs (MSR), 2017, Marseille, France. hal-01899930

**HAL Id: hal-01899930**

**<https://hal.science/hal-01899930>**

Submitted on 20 Oct 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Approche originale utilisant le Raisonnement à Partir de Cas pour le Diagnostic en Ligne des Systèmes Automatisés de Production

Nourhène Ben Rabah<sup>1,2</sup>, Ramla Saddem<sup>1</sup>, Faten Ben Hmida<sup>2</sup>,  
Véronique Carre-Menetrier<sup>1</sup> et Moncef Tagina<sup>2</sup>

<sup>1</sup> CReSTIC UFR Sciences Exactes et Naturelles Reims, France

<sup>2</sup> COSMOS, Ecole nationale des Sciences de l'informatique, Tunisie  
nourhene.ben-rabah@etudiant.univ-reims.fr

## Résumé

Ce papier s'inscrit dans le cadre du diagnostic en ligne des Systèmes Automatisés de Production (SAP) classés ici dans les Systèmes à Événement Discret (SED). Il présente une approche originale utilisant le raisonnement à partir de cas (RàPC). L'originalité de cette proposition réside dans les quatre items suivants : (1) elle utilise ce type de raisonnement pour le diagnostic de cette catégorie de systèmes, (2) elle propose un format de représentation de cas inspiré du formalisme Signatures Temporelles Causales (STC) qui s'adapte à l'aspect dynamique des systèmes à surveiller, (3) elle expose une phase qui couple simulation et mise en forme de données pour la transformation des données issues du système simulé après son émulation en mode normal et défaillant, et (4) elle présente une phase de raisonnement et d'apprentissage qui permet non seulement le diagnostic en ligne du système surveillé mais aussi la mise à jour de la base de cas suite à l'apparition de nouveaux comportements inconnus.

**Mots clés :** Diagnostic en ligne, Raisonnement à partir de cas, Signatures Temporelles Causales, Représentation de cas, Similarité.

## 1 Introduction

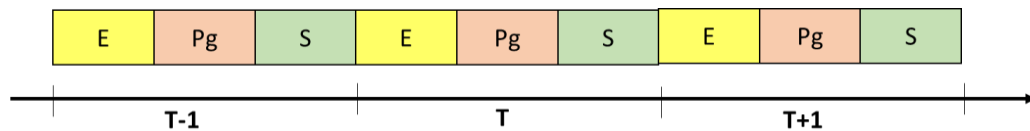
L'usine du futur est un nouveau modèle d'usine pensé pour être au cœur de son éco-système et répondre aux nouveaux besoins sociétaux. Il s'agit entre autre d'une usine aux lignes de production et logistiques innovantes, performantes, sûres, mises au point en les simulant et en les optimisant dans le mode virtuel (twin digital, jumeau numérique, ...)(Boschert et al., 2016). Par conséquent, la mise en œuvre d'approches systématiques de diagnostic constitue une phase primordiale dès la conception des Systèmes Automatisés de Production (SAP).

Le rôle de diagnostic ainsi que les approches de diagnostic dépendent des dynamiques des SAP qui peuvent être vus comme des systèmes continus, des systèmes à événements discrets (SED) (Cassandras et Lafortune, 2009) ou des systèmes hybrides. Dans notre travail, nous nous intéressons au diagnostic des SAP qui possèdent des capteurs et des actionneurs délivrant des signaux binaires (Tout ou Rien) et qui se classent dans les SED. La plupart de ces systèmes sont commandés par des automates programmables industriels (API) qui exécutent un cycle (T) comprenant trois opérations successives (E, Pg, S) :

- E : l'acquisition des entrées qui consiste à la réalisation d'une image du monde extérieur à travers l'enregistrement des états des entrées qui sont des variables non contrôlables (capteurs).

- Pg : l'exécution du programme de la commande
- S : la mise à jour des états des sorties qui sont des variables contrôlables (actionneurs).

Ces API fonctionnent soit de manière cyclique c'est à dire enchainant les cycles les uns après les autres ou de manière périodique. Dans ce travail, nous utilisons des API avec un fonctionnement cyclique (figure 1). Le cycle se répète indéfiniment tant qu'il n'y a pas d'interruption qui engendre l'arrêt temporaire ou permanent de l'automate.



**Figure 1:** Fonctionnement cyclique de l'API.

Le diagnostic consiste par conséquent à : i) récupérer les informations issues des capteurs et des ordres de commande. (ii) déduire de ces informations les événements observables et (iii) analyser ces événements afin de détecter et d'isoler les fautes possibles. La littérature distingue deux types d'approches pour le diagnostic de cette catégorie de systèmes : la première approche est à base de modèles diagnostiqueurs (Sampath et al., 1995) et la seconde est à base de formalismes de description et de reconnaissance de comportement comme les chroniques avec ses différentes formalisations (Dousson et al., 1993), (Boufaied, 2002), (Bertrand et al., 2007), (Carle et al., 2011) et les Signatures Temporelles Causales (STC) (Toguyeni et al., 1991), (Saddem, 2011).

L'avantage de la première approche est la validation de la cohérence et la complétude des défauts à diagnostiquer. Par contre, elle souffre du problème d'explosion combinatoire liée au formalisme des automates lors de l'implémentation sur des systèmes réels (Saddem et Philppot, 2014). La deuxième approche offre une efficacité élevée à cause de la connaissance des symptômes des fautes qu'elle modélise (Cordier et Dousson, 2000). Cependant, elle souffre du problème de l'acquisition des connaissances expertes et de leur mise à jour.

Dans ce papier, nous proposons une solution originale pour le diagnostic en ligne des SAP qui ont une dynamique discrète. Notre solution se base sur le Raisonnement à Partir de Cas (RàPC) qui se présente comme une méthodologie de raisonnement par analogie et aussi une méthodologie d'apprentissage issue du domaine de l'intelligence artificielle (IA). La mise en œuvre de notre solution nécessite une démarche scientifique impliquant deux phases :

- La première phase dite phase de construction de la base d'apprentissage décrit les comportements normaux et défaillants d'un jumeau numérique d'un SAP réel sous forme des connaissances empiriques représentées sous forme de cas. Cet ensemble de cas forme la base d'apprentissage.
- La seconde phase dite phase de raisonnement et d'apprentissage fonctionne en ligne et permet d'une part le diagnostic du système surveillé et d'autre part la mise à jour de la base de cas suite à l'apparition de nouveaux comportements inconnus.

L'article est structuré de la manière suivante. Dans la section 2, nous présentons brièvement la méthodologie de RàPC et nous donnons des exemples d'application pour le diagnostic des systèmes industriels. La section 3 expose la contribution de ce papier. Nous illustrons notre approche par une application sur un exemple académique dans la section 4. Enfin, cet article se termine par des conclusions et perspectives autour de ce travail.

## 2 Les concepts de base

### 2.1 Un aperçu sur le raisonnement à partir de cas

Le raisonnement à partir de cas (RàPC) est une méthodologie de résolution de problèmes et d'apprentissage qui est fondamentalement différente des autres approches de l'intelligence artificielle (IA). Elle est capable d'utiliser les connaissances spécifiques des expériences passées, formalisées sous forme des situations problématiques concrètes appelées des cas. Un nouveau problème est résolu en cherchant un cas passé le plus similaire et en l'utilisant pour la résolution de la nouvelle situation problématique. Une caractéristique très importante du RàPC est son couplage avec l'**apprentissage**. Il permet de mettre à jour des cas, d'apprendre de nouveaux cas, de s'assurer de leur cohérence et d'apprendre des succès et des échecs. En effet, chaque nouvelle

expérience est retenue à chaque fois que le problème est résolu avec succès et elle est utilisée immédiatement pour la résolution des problèmes futurs. Dans les cas d'échecs, les raisons sont identifiées et stockées afin d'éviter de commettre à nouveau les mêmes erreurs.

Le processus de RàPC s'effectue par un cycle comprenant 4 étapes (Aamodt et al, 1994) présenté en figure 2 :

- a) La recherche : Elle permet de sélectionner les cas similaires ou les plus similaires de la base de cas.
- b) La réutilisation : Les cas sélectionnés sont ensuite réutilisés pour la résolution d'un nouveau problème.
- c) La révision : C'est la révision de la solution proposée.
- d) La mémorisation : Le nouveau cas engendré est mémorisé dans la base de cas pour la résolution des problèmes futurs.

Un cas est le composant élémentaire d'un système à base de cas. Dans la littérature, il n'existe pas une définition consensuelle du terme cas. En effet, sa définition est liée par rapport à son format de représentation (Lieber et al, 2008), (Kolodner, 1993). A partir des définitions existantes dans la littérature, nous présentons 4 types de formats de représentation de cas : structurel, textuel, conversationnel et hybride.

- i. Format structurel : les cas sont représentés selon une structure commune. Ils sont construits à partir de l'extraction des caractéristiques importantes qui modélisent le problème à résoudre. Les formalismes de représentation de connaissance peuvent être statiques ou dynamiques. Ils sont statiques s'ils ne permettent pas de prendre en compte l'aspect dynamique des éléments à modéliser dans un domaine d'application. Sinon, ils sont dynamiques. Les représentations basées sur les vecteurs de caractéristiques, les frames (Plaza, 1995), la logique de description (Donini, 1996) sont des exemples de représentation statiques et la représentation basée sur les objets (Bergmann, 2002) est un exemple de représentation dynamique.
- ii. Format textuel (Lenz, 1998): Un cas peut être représenté sous forme d'un texte libre (non structuré) ou représenté sous forme du texte découpé en plusieurs portions étiquetées par des descripteurs (semi-structuré).
- iii. Format conversationnel (Aha, 2001): Un cas est représenté à travers 3 parties (description, questions, actions). (i) Description : décrit le problème à résoudre sous forme de texte. (ii) Questions : représente une série de questions/ réponses. Chaque question a une pondération pour représenter son importance par rapport au cas. (iii) Actions : c'est une description textuelle de la solution à mettre en œuvre.
- iv. Format hybride : Un cas est représenté en combinant au minimum 2 formats.

Ainsi, nous pouvons constater que le choix du format de représentation est un problème qui se pose lors de la création d'un système de RàPC. En effet, cette étape importante peut influencer toutes les autres phases du cycle du RàPC. En conséquence, nous avons ajouté au cycle du RàPC introduit par (Aamodt et al, 1994) l'étape de représentation de cas. Dans cette étude, nous accordons donc une grande importance à cette étape, elle est ainsi ajoutée en haut à gauche dans la figure 2.

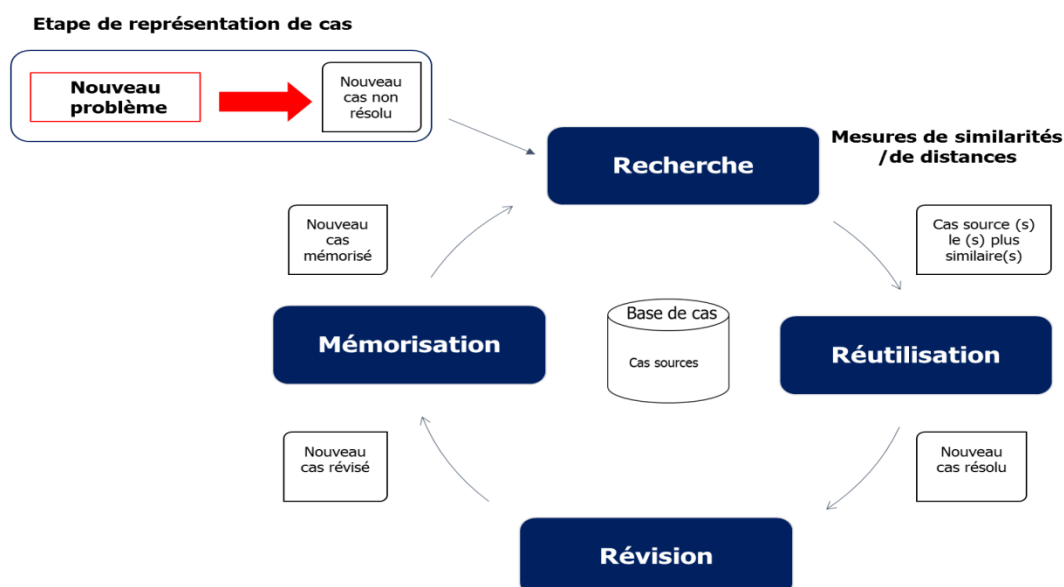


Figure 2: Cycle de RàPC.

## 2.2 Exemples d'applications de RàPC pour le diagnostic des systèmes industriels

Le RàPC a été utilisé dans plusieurs secteurs d'application du domaine industriel (Chougule et al., 2011), (Behbahani et al., 2012), (Ruiz et al., 2013), (Dendani et al., 2012), (Lejri et Tagina, 2012), (Vareilles et al., 2012), (Xiong et al., 2012). Ces travaux récents visent à résoudre un problème de diagnostic, de maintenance ou de reconfiguration tout en représentant la phase de diagnostic à base de RàPC. Dans cette étude, nous nous sommes intéressés à la manière dont les auteurs abordent les étapes de représentation et d'élaboration de cas et les phases de recherche, de réutilisation et de révision.

(Ruiz et al., 2013) et (Dendani et al., 2012) ont choisi une représentation basée sur les connaissances du domaine à travers la construction d'une ontologie de domaine. Un cas est décrit sous forme d'un ensemble de descripteurs où chaque descripteur est un doublet : un concept et sa valeur. Le problème de cette représentation réside dans la construction de l'ontologie de domaine qui reste une tâche difficile pour les systèmes complexes. (Behbahani et al., 2012) ont proposé une représentation de cas enrichie qui se base sur 15 descripteurs qui présentent les caractéristiques les plus importantes du problème. Mais le choix des descripteurs reste à réaliser manuellement par des experts du domaine. (Vareilles et al., 2012) ont présenté une représentation de cas qui prend en compte les différents types de connaissances : généraux et contextuels. Cependant le cas est représenté à travers 110 variables ce qui est très important. A chaque nouveau problème de maintenance, l'utilisateur doit remplir un formulaire de 110 entrées. (Chougule et al., 2011) ont exposé une représentation de cas typiques qui se base sur les symptômes observés, leurs dates et les informations relatives aux véhicules. (Xiong et al., 2012) ont pris en compte l'aspect temporel dans la représentation de cas à travers l'extraction des caractéristiques des signaux des capteurs en utilisant l'abstraction temporelle.

Pour la phase de recherche des cas sources les plus similaires au cas cible, le choix de la mesure de similarité ou de distance dépend de la structure de représentation de données, de type de données à comparer et de l'importance donnée par l'utilisateur à certains attributs par rapport aux autres. (Dendani et al., 2012), (Ruiz et al., 2013), (Behbahani et al., 2012) et (Lejri et Tagina, 2012) ont utilisé différentes mesures de similarité qui prennent en compte le degré d'importance de certains attributs vis-à-vis des autres.

La phase de réutilisation de la solution sélectionnée est une phase sensible. Elle est réalisée par copie de la meilleure solution sélectionnée dans le travail de (Lejri et Tagina, 2012), (Vareilles et al., 2012) et (Behbahani et al., 2012), par fusion des solutions sélectionnées dans le travail de (Xiong et al., 2012) et par l'utilisation des relations de dépendance d'une ontologie de domaine dans (Ruiz et al., 2013) et (Dendani et al., 2012). Finalement, aucun des travaux étudiés ne mentionne la phase de révision.

A travers cette étude, nous dégageons le constat suivant : (i) les systèmes de RàPC étudiés sont construits globalement autour d'une représentation de connaissances sous forme d'un vecteur constitué d'une paire attribut-valeur (vecteur de caractéristiques). (ii) le choix des caractéristiques importantes (e.g l'indexation) est fait manuellement par un opérateur humain. (iii) aucun travail n'a présenté une solution pour le diagnostic des SAP ayant une dynamique à événements discrets et (iv) les solutions existantes sont trop spécifiques aux systèmes industriels à diagnostiquer et ne peuvent pas être généralisées.

Au vu de ce constat, nous détaillons notre approche pour résoudre ces problèmes.

## 3 Proposition

La procédure proposée se base sur l'utilisation d'une méthodologie de raisonnement et d'apprentissage (RàPC) qui apparaît intéressante pour aborder le problème de diagnostic en ligne des fautes possibles dans les SAP lorsque les connaissances du domaine sont difficiles et couteuses à exprimer. Cependant, pour ces systèmes complexes, il n'est pas possible ou pratique de spécifier complètement toutes les connaissances. Notre solution consiste d'une part à l'acquisition des connaissances expertes à partir des informations issues d'un jumeau numérique et d'autre part, au diagnostic en ligne des fautes possibles et la mise à jour de la base de connaissances suite à l'apparition de nouveaux comportements inconnus. La mise en œuvre de notre approche nécessite une démarche impliquant deux phases telles que la phase de construction de la base d'apprentissage et la phase de raisonnement et d'apprentissage en ligne. La figure 3 illustre ces propos dont les détails sont donnés dans les paragraphes suivants.

### 3.1 Construction de la base d'apprentissage

Dans cette section nous décrivons la phase de construction de la base d'apprentissage de l'acquisition des données jusqu'à la forme finale utilisée pour le raisonnement et l'apprentissage. Cette phase est basée sur un module de simulation et un processus de mise en forme de données (figure 3).

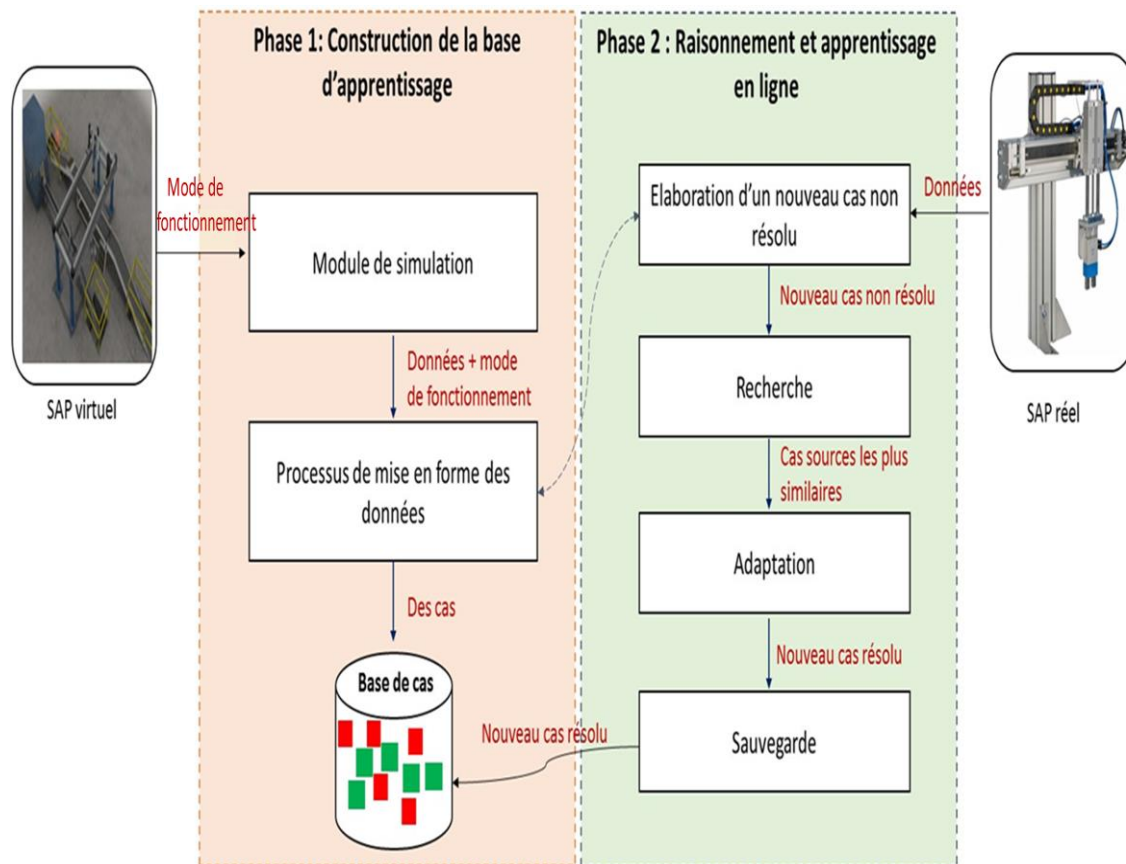


Figure 3: Approche proposée.

#### 3.1.1 Le module de simulation

La simulation permet de décrire l'évolution d'un jumeau numérique d'un SAP réel dans le temps afin de fournir des données utiles sur son comportement dynamique dans différentes situations (y compris les situations de dysfonctionnement). Ces données proviennent de différents capteurs et actionneurs du jumeau numérique, délivrant des signaux binaires (Tout ou Rien).

Le principe de ce module est comme suit :

- Etape 1 : Faire fonctionner le jumeau numérique dans plusieurs scénarios de fonctionnement normal (sans l'injection de fautes)
- Etape 2 : Pour chaque scénario possible, collecter les données des capteurs et des actionneurs et les mettre en forme finale (le passage des données vers les connaissances empiriques). Ce processus de mise en forme des données est détaillé dans le paragraphe suivant.
- Refaire l'étape 1 et 2 mais avec injection de fautes pour avoir plusieurs scénarios de fonctionnements défectueux.

Les scénarios de fonctionnement normal sont déduits à partir du cahier de charge du SAP alors que les scénarios de fonctionnement défectueux sont définis à partir de fautes que l'on souhaite diagnostiquer dans le système. Il existe deux types de fautes : observables comme le passage inattendu de 0 à 1 ou de 1 à 0 d'un capteur ou d'un actionneur et non observables tels que le blocage à 1 ou à 0 d'un capteur ou d'un actionneur.

### 3.1.2 Le processus de mise en forme de données

Dans ce processus les données issues du jumeau numérique sont traitées pour extraire des informations pertinentes qui sont modélisées ensuite sous forme de cas. Ces cas caractérisent et modélisent les comportements normaux et défaillants du SAP. Et comme nous avons souligné dans la section 2, nous accordons une grande importance à l'étape de représentation de cas. De ce fait, nous proposons dans la suite un format de représentation de cas qui s'adapte à l'aspect dynamique des systèmes à surveiller.

#### a) Formalisation des cas

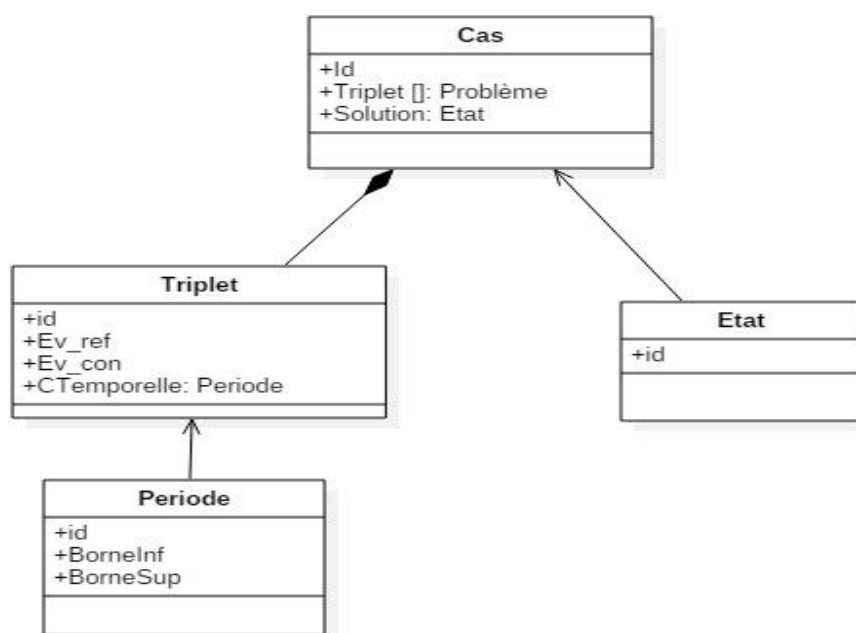
Pour la formalisation de nos cas, nous nous sommes inspirés des Signatures temporelles causales (Toguyeni et al., 1991 ; Toguyéni et al., 1997) qui constituent un formalisme de description et de reconnaissance des comportements très proche des chroniques qui permettent le diagnostic des SED.

Une STC est une règle qui peut être formellement définie comme suit :  $X \rightarrow Y$  avec  $X$  est une séquence de triplets et  $Y$  est l'état du système suite à cette séquence (normal ou défaillant).

Soit  $(A, B, ct)$  un triplet. 'A' et 'B' sont deux événements observables avec 'A' est l'évènement référent et 'B' est l'évènement contraint. 'ct' est une contrainte temporelle de type période qui permet de modéliser avec un degré d'incertitude le temps entre l'occurrence de deux événements. 'ct' devient 'nct' dans le cas d'absence de contrainte temporelle. '\*' est un opérateur de séparation entre les triplets. 'In' est un évènement observable qui est toujours occurrent et qui n'a pas de contrainte temporelle.

Afin d'assurer une formalisation adéquate de nos cas, nous proposons d'adopter un format de représentation basé sur les objets. Un cas est représenté par un objet qui a un identifiant unique et décrit par deux attributs (*Problème*, *Solution*). L'attribut *Problème* est représenté par un vecteur d'objets de type Triplet et l'attribut *Solution* est représenté par un objet Etat qui peut être de type normal ou défaillant.

La figure 4 décrit la formalisation adoptée sous la forme d'un diagramme de classe UML.



**Figure 4 :** Le format de cas proposé sous la forme d'un diagramme de classe UML.

Pour la génération de nos cas, nous proposons un algorithme qui permet de générer les parties Problèmes des cas en se basant sur les signaux binaires des capteurs et des actionneurs. Le résumé de l'algorithme est présenté dans le paragraphe suivant.

## b) Génération des cas

A chaque cycle automate (noté T), le jumeau numérique du SAP envoie des signaux binaires des capteurs et des actionneurs. Notre algorithme consiste à stocker ces signaux dans une variable nommée « signature binaire » qui est associée au cycle T. Si cette signature binaire est différente de la signature binaire du cycle précédent (T-1), un objet triplet (Ev\_ref, Ev\_con, CTemporelle) est construit. Ev\_ref est l'évènement référent, Ev\_con est l'évènement contraint et CTemporelle est une contrainte temporelle de type période. La partie *Problème* du cas est représentée par un vecteur d'objets de type Triplet et d'une taille minimum supérieur ou égale à 2. En effet, un seul triplet n'est pas suffisant pour décrire les comportements du système surveillé.

On note *TempsCyclecourant* la mesure du temps associée à la signature binaire du cycle d'automate T et *TempsCycleprécédent* la mesure du temps associée à la signature binaire du cycle d'automate (T-1) sachant que les deux signatures binaires sont différentes.

L'algorithme proposé est illustré par quatre étapes :

- ✓ **Étape 1:** Regrouper les signaux des capteurs et actionneurs du système à diagnostiquer pendant le cycle de l'automate (T) afin de construire une signature binaire et lui associer un temps de cycle.
- ✓ **Étape 2:** Formuler l'évènement de référence (Ev\_ref): Si c'est le premier cycle de l'automate exécuté alors Ev\_ref est un évènement 'In' sinon l'évènement contraint du cycle API précédent (T-1) devient l'évènement de référence du cycle PLC (T).
- ✓ **Étape 3:** Formuler l'évènement contraint (Ev\_con): Chaque élément de la signature binaire est transformé en un évènement qui peut être soit le front montant (désigné par le préfixe R) ou le front descendant (désigné par le préfixe F) d'un capteur ou actionneur. Cet évènement est défini comme un évènement contraint.
- ✓ **Étape 4:** Formuler la contrainte temporelle (CTemporelle) qui est décrite à travers deux attributs BorneInf (borne inférieure de la période) et BorneSup (borne supérieure de la période) : si l'évènement référent (Ev\_ref) est égal à 'In' alors absence de contrainte temporelle : BorneInf=0 et BorneSup=0 sinon la contrainte temporelle est calculée après une estimation des erreurs aléatoires de mesure dues au matériel, aux paramètres physiques et à l'opérateur. Cela conduit à exprimer premièrement un temps estimé :  
$$\text{Temps}_{\text{estime}} = \text{TempsCyclecourant} - \text{TempsCycleprécédent}$$
  
par la suite :  
$$\text{BorneInf} = \text{Temps}_{\text{estime}} - \Delta t \text{ et } \text{BorneSup} = \text{Temps}_{\text{estime}} + \Delta t$$
  
avec  $\Delta t$  représentant l'incertitude sur la valeur du temps,  $\Delta t$  est  $>0$  et elle est fixé d'avance par un expert du domaine.
- ✓ **Étape 5:** regrouper le résultat des 3 étapes précédentes pour construire un objet Triplet.

La complexité de l'algorithme est une complexité linéaire par rapport à la taille de la signature binaire :  $O(K(n + m))$  où K est le nombre de cycles d'automates réalisés par le système de production automatisé, n est le nombre de capteurs et M est le nombre d'actionneurs.

## 3.2 La phase de raisonnement et d'apprentissage en ligne

Pour le diagnostic en ligne d'un SAP réel, nous nous sommes basés sur les connaissances empiriques formalisées sous forme de cas et stockées dans la base de cas. A chaque cycle d'API, les signaux des capteurs et des actionneurs sont représentés sous la forme d'une nouvelle signature binaire. Ensuite, cette signature est transformée en une partie *Problème* (e.g un vecteur d'objets triplet) d'un nouveau cas appelé nouveau cas non résolu, via l'algorithme décrit dans la section précédente. Ce nouveau cas représente le nouveau problème à résoudre.

Cette phase permet d'une part la reconnaissance en ligne de ce nouveau problème et d'autre part la mise à jour de la base de cas. Afin d'atteindre ces objectifs, elle consiste à rechercher les cas sources similaires ou les plus similaires, à réutiliser la solution du cas source sélectionné et à sauvegarder le nouveau cas résolu dans la base (figure 3).

### ➤ Recherche

L'étape de recherche permet d'extraire pour chaque nouveau cas, les cas similaires ou les plus similaires. Dans cette étude, cette étape est basée premièrement sur l'extraction des différents cas qui ont le même nombre de triplets que le nouveau cas et deuxièmement sur le raffinement du choix par le calcul de similarité entre le nouveau cas et le premier groupe sélectionné. Le calcul de similarité est basé sur l'utilisation d'une métrique de



similarité qui permet de calculer le degré de ressemblance entre les cas. Elle est généralement dérivée de produits scalaires ou de mesures de dissimilarités (Distance) (Lesot, 2005).

Nous utilisons un « Calculateur de Distance » pour calculer la distance entre deux cas. Il a comme entrée le cas cible et comme sortie la distance calculée entre ce cas et chaque cas source appartenant au premier groupe sélectionné.

Soit D une relation de distance définie par :

$$D = \text{Cas} \times \text{Cas} \rightarrow [0, 1]$$

- Si  $\text{Problème}_i$  et  $\text{Problème}_j$  sont identiques et ont la même taille alors  $D(\text{Cas}_i, \text{Cas}_j) = 0$
- Si  $\text{Problème}_i$  et  $\text{Problème}_j$  n'ont pas la même taille ou aucun triplet identique alors  $D(\text{Cas}_i, \text{Cas}_j) = 1$

Afin de calculer la distance séparant deux cas, nous devons calculer la distance séparant ses deux problèmes ( $\text{Problème}_i, \text{Problème}_j$ ). Un *Problème* est un vecteur contenant des objets de type Triplet. Un objet Triplet est défini par des caractéristiques de différent type (événements de type chaîne de caractères et contrainte temporelle de type période) ce qui rend le calcul de distance une étape difficile.

Pour faire face à ce problème, nous discrétisons les valeurs des attributs de l'objet Triplet comme suit. Nous associons la valeur  $DV_e$  définie dans (1) pour la discrétisation des attributs de type chaîne de caractères (événement contraint, événement référent) et nous associons la valeur  $DV_{ct}$  spécifiée dans (2) pour la discrétisation des attributs de type contrainte temporelle.

$$DV_e = \begin{cases} 1 & \text{si } \text{Val}(e_{t_k,i}) = \text{Val}(e_{t_k,j}) \\ 0 & \text{sinon} \end{cases} \quad (1)$$

Avec Val est la valeur de l'évènement, ( $e_{t_k,i}$ ) est un évènement du triplet  $t_k$  appartenant au  $\text{Cas}_i$  et ( $e_{t_k,j}$ ) est un évènement du triplet  $t_k$  appartenant au  $\text{Cas}_j$ .

$$DV_{ct} = \begin{cases} 1 & \text{si } \text{Val}(\text{BorneInf}(C_{t_k,i})) \geq \text{Val}(\text{BorneInf}(C_{t_k,j})) \\ 0 & \text{sinon} \end{cases} \quad (2)$$

Avec Val est la valeur de l'évènement, ( $C_{t_k,i}$ ) est une contrainte temporelle du même triplet  $t_k$  appartenant au  $\text{Cas}_i$  et ( $C_{t_k,j}$ ) est une contrainte temporelle du même triplet  $t_k$  appartenant au  $\text{Cas}_j$ .

Le choix de la métrique de distance dépend de la structure de représentation de données, de type de données à comparer (nominal, ordinal, continu ou binaire) et de l'importance donnée par l'utilisateur à certains attributs par rapport aux autres. Dans notre étude, nous choisissons une distance inspirée de la distance de Manhattan (Bisson, 2000) pour calculer la distance entre deux problèmes pour deux raisons : (i) Un *Problème* est un vecteur contenant des objets de type Triplet et pour chaque objet Triplet, nous associons une valeur du type numérique et (ii) les objets Triplet ont la même importance. Le calcul de la distance est effectué comme suit :

$$D(\text{Problème}_i, \text{Problème}_j) = \frac{1}{3m} * \sum_{k=1}^m |V_{t_k,i} - V_{t_k,j}| \quad (3)$$

Avec  $m$  est le nombre de triplets de deux problèmes à comparer,  $V_{t_k,i}, V_{t_k,j}$  sont les valeurs des triplets.

#### ➤ Adaptation

L'étape d'adaptation permet de déterminer la Solution du nouveau cas (cas cible) par la réutilisation de la solution du cas source sélectionné. Elle peut être manuelle, par copie ou automatique à l'aide d'algorithme, de

formules et de règles (Wilke et Bergmann, 1998). Dans notre étude, nous avons utilisé une adaptation par copie où le nouveau cas hérite la solution du cas source qui a la distance minimale.

#### ➤ Sauvegarde

Le nouveau cas résolu est retenu dans la base de cas et il est utilisé immédiatement pour la résolution des problèmes de reconnaissance futurs.

## 4 Exemple Illustratif

L'ensemble de l'approche proposée est illustré à travers un système virtuel de la collection ITS PLC proposée par la société portugaise Real Games ([www.realgames.pt](http://www.realgames.pt)). ITS PLC est un outil d'éducation, de formation à l'automatisation et de validation de l'algorithme de commande grâce à une expérience interactive en temps réel (Riera et al., 2010). Il offre des simulations 3D des parties opératives de 5 systèmes industriels (tri de caisses, système de mélange, palettiseur, robot « pick and place », magasin automatisé). ITS PLC existe en deux versions : une version bêta et une version professionnelle.

Dans le cadre de cette étude, nous utilisons la première version qui permet de :

- Utiliser des scripts IronPython (<http://ironpython.net>) pour écrire nos propres contrôleurs dans un langage proche du langage ST (Texte Structuré).
- Accéder à un interpréteur IronPython interactif permettant à l'utilisateur d'interagir avec chaque système simulé en accédant par exemple à ses entrées / sorties via l'objet IO. IO.Actuators et IO.Sensors renvoient respectivement les signaux des actionneurs et des capteurs.
- Injecter des pannes dans les capteurs et les actionneurs.

Notre proposition a été mise en œuvre par 2 scripts. Le premier script permet de commander le système industriel virtuel sans avoir besoin d'un API réel et le second permet d'accéder aux entrées/sorties du système pour chaque cycle d'automate et d'implémenter l'algorithme proposé et le calculateur de distance.

### 4.1 Description du système

Dans le cadre de ce travail, nous avons utilisé le système de tri de caisses de l'outil ITS PLC. L'objectif est de transporter des caisses du convoyeur d'alimentation au monte-charge en les triant selon leur hauteur (petite ou grande caisse). Dans ce papier, nous présentons uniquement nos résultats pour un composant du système de tri de caisses qui est le plateau tournant. La Figure 5 présente un automate avec 6 états et 10 transitions afin de représenter toutes les évolutions possibles de la partie opérative du plateau.

Le comportement normal du composant est décrit à travers deux scénarios possibles :

#### Scénario 1 :

À partir de l'état initial "0", le plateau tournant est dans la position de chargement c4. Si l'actionneur S4 est activé, le plateau tournant est en mouvement et le capteur c4 est désactivé (passage de l'état "1" à "2"). De là, si la commande est toujours active, le plateau tournant revient à la position de déchargement (transition de l'état "2" à "3"). La désactivation des actionneurs S4 permet de revenir à la position d'origine (indique "4" après "5" puis "0").

#### Scénario 2 :

De l'état "2", pendant le mouvement, si le contrôleur envoie la désactivation de S4, le plateau revient directement à l'état "0".

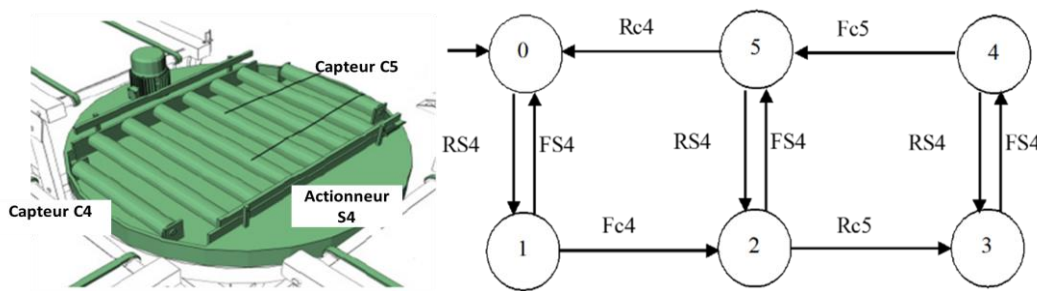


Figure 5: Le plateau tournant et son modèle.

Les différentes fautes qui peuvent se produire dans le plateau tournant sont : **F1** : Capteur c4 bloqué à 0, **F2** : Capteur c4 bloqué à 1, **F3** : Capteur c5 bloqué à 0, **F4** : Capteur c5 bloqué à 1, **F5** : Plateau bloqué en c4 (S4 bloqué à 0), **F6** : Plateau bloqué en c5 (S4 bloqué à 1), **F7** : Capteur c4 de 0 à 1 inattendu, **F8** : Capteur c4 de 1 à 0 inattendu, **F9** : Capteur c5 de 0 à 1 inattendu, **F10** : Capteur c5 de 1 à 0 inattendu.

## 4.2 Résultats

### • Phase de construction de la base d'apprentissage

Durant la phase de construction de la base d'apprentissage, nous activons le système tri de caisse avec plusieurs scénarios ; fonctionnement normal (sans injection des fautes) et fonctionnement défaillant (avec injection des fautes). A chaque cycle d'API, les signaux des différents capteurs et actionneurs du plateau tournant (C4, C5, S4) sont modélisés sous la forme d'une signature binaire qui est transformée ensuite en un cas labélisé normal ou défaillant.

Et comme nous avons mentionné ci-dessus, un cas est représenté par un attribut *Problème* et un attribut *Solution* (Figure 3). Le type de l'attribut *Problème* est un vecteur d'objet Triplet avec un objet Triplet est représenté par 4 attributs (id, Ev\_ref, Ev\_con, CTemporelle). Le type de l'attribut CTemporelle est un objet Periode qui est décrit par (id, BorneInf, BorneSup). En respectant cette représentation, le tableau 1 montre différents exemples d'instances de la base d'apprentissage.

L'instance 5 présente un cas normal qui décrit le scénario de fonctionnement 1 (introduit ci-dessus). Le sens de l'instance est le suivant. Si on a : (**t1**) l'occurrence de l'événement « **front montant** de l'actionneur S4 noté **RS4** », ensuite (**t2**) si on a l'occurrence de l'événement « **front descendant** du capteur C4 (**FC4**) » satisfaisant la contrainte de temps [80, 85] par rapport à RS4, après (**t3**) l'occurrence de l'événement « **front montant** du capteur C5 noté **RC5** » satisfaisant la contrainte [2816, 2821] par rapport à FC4, ensuite (**t4**) l'occurrence de l'événement « **front descendant** de l'actionneur S4 (**FC4**) » satisfaisant la contrainte de temps [6912, 6917] par rapport à RC5, puis (**t5**) l'occurrence de l'événement « **front descendant** de l'actionneur C5(**FC5**) » satisfaisant la contrainte de temps [80, 85] par rapport à FS4 et finalement (**t6**) l'occurrence de l'événement « **front montant** du capteur C5 (**RC5**) » satisfaisant la contrainte de temps [2816, 2821] par rapport à FC5 alors la solution de cas est le SAP est normal.

L'instance 11 présente un cas anormal qui décrit le scénario de la faute F2 qui correspond à un blocage à 1 du capteur C4. Le sens de l'instance est le suivant. Si on a : (**t1**) l'occurrence de l'événement « **front montant** de l'actionneur S4 noté **RS4** », ensuite on a (**t2**) l'occurrence de l'événement « **front montant** du capteur C5 (**RC5**) » satisfaisant la contrainte de temps [2896, 2901] alors on peut déduire la faute F2 au sein du SAP.

### • Phase de raisonnement et d'apprentissage en ligne

Après la phase de simulation et de mise en forme des données vient la phase de raisonnement et d'apprentissage en ligne en se basant sur les expériences stockées dans la base d'apprentissage. Cette phase utilise les expériences passées pour résoudre de nouveaux problèmes. Elle consiste à formaliser les signaux des capteurs et des actionneurs sous la forme d'une partie *Problème* d'un nouveau cas (appelé aussi nouveau cas non résolu). La figure 6 montre un exemple de son élaboration.

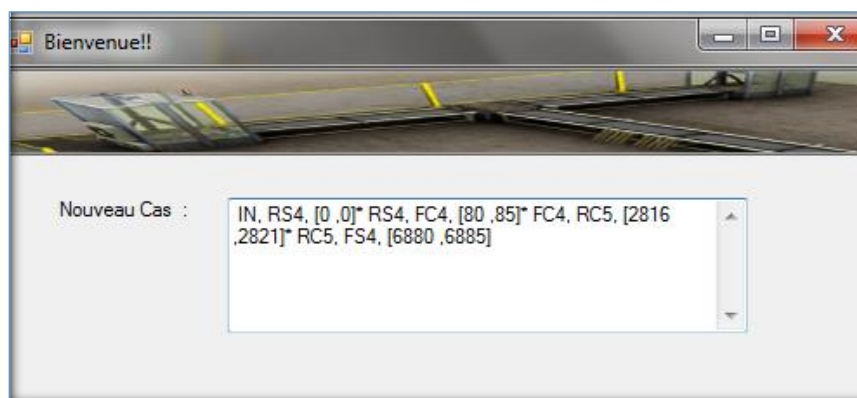


Figure 6 : Elaboration d'un nouveau cas non résolu.

Problème																																Solution					
ID	t1						t2						t3						t4						t5						t6						
	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L	U	e <sub>r</sub>	e <sub>c</sub>	L		U				
1	In	RS4	0	0	RS4	FC4	80	85																										N			
2	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821																						N			
3	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6912	6917																		N			
4	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6912	6917	FS4	FC5	80	85														N			
5	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6912	6917	FS4	FC5	80	85	FC5	RC5	2816	2821										N			
6	RC4	RS4	21760	21765	RS4	FC4	80	85																										N			
7	RC4	RS4	21760	21765	RS4	FC4	80	85	FC4	RC5	2816	2821																						N			
8	RC4	RS4	21760	21765	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6864	6869																		N			
9	RC4	RS4	21760	21765	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6864	6869	FS4	FC5	80	85														N			
10	RC4	RS4	21760	21765	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6864	6869	FS4	FC5	80	85	FC5	RC5	2816	2821										N			
11	In	RS4	0	0	RS4	RC5	2896	2901																										F2			
12	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FS4	6928	6933	FS4	FC5	80	85	FC5	RC5	2496	2501										F4			
13	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	1616	1621																							F9		
14	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FC5	720	725																			F6		
15	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	RC4	1376	1381																			F7		
16	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	RC4	656	661	RC4	FS4	6240	6245															F5		
17	In	RS4	0	0	RS4	FC4	80	85	FC4	RC5	2816	2821	RC5	FC5	192	197																			F10		

**Problème** : un vecteur d'objets Triplet, **Solution** : Etat du SAP, ti : un objet triplet d'un cas, e<sub>r</sub>: Ev\_ref, e<sub>c</sub>: Ev\_con, I: BorneInf, S: BorneSup.

**Tableau 1:** Exemples d'instances de la base d'apprentissage.

Ensuite vient l'étape de recherche qui extrait les différents cas de la base qui ont le même nombre de triplets que le nouveau cas et qui calcule la distance entre ce cas et les anciens cas du groupe sélectionné. Le calcul de distance se fait suivant la formule 1. Le tableau 2 montre un ensemble de calcul de distance entre le nouveau cas générée dans la figure 4 et les cas de la base de cas.

Id du cas	Problème	Solution	Distance proposée
3	(In, RS4, [0,0])*(RS4, FC4,[80,85])*(FC4,RC5,[2816,2821])*(RC5,FS4,[6912, 6917])	Normal	<u>0,333</u>
14	(In, RS4, [0,0])*(RS4, FC4,[80,85])*(FC4,RC5,[2816,2821])*(RC5,FC5,[720, 725])	Défaillant	0,416
15	(In, RS4, [0,0])*(RS4, FC4,[80,85])*(FC4,RC5,[2816,2821])*(RC5,RC4,[1376, 1381])	Défaillant	0,416
8	(In, RS4, [0,0])*(RS4, FC4,[80,85])*(FC4,RC5,[2816,2821])*(RC5,FS4,[6864, 6869])	Normal	0,833

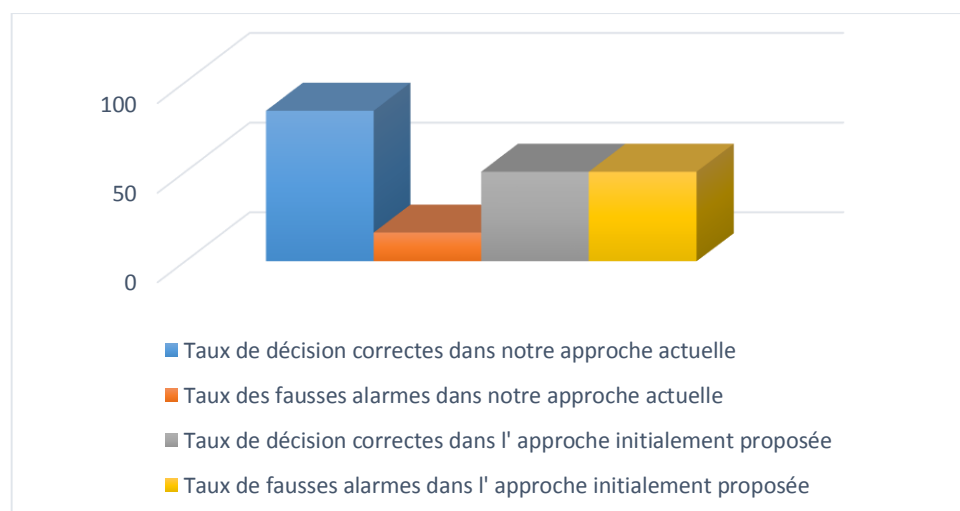
**Tableau 2 :** Recherche des cas sources les plus similaires.

Le nouveau cas héritera la solution du cas qui a la distance minimale, il est stocké dans la base de cas et il est utilisé immédiatement pour la résolution des problèmes futurs.

### 4.3 Récapitulation des résultats expérimentaux

Afin d'évaluer les performances de l'approche proposée pour le diagnostic en ligne des fautes possibles dans le plateau tournant, nous avons mené une série de cinquante tests et nous avons comparé ses résultats avec la proposition présentée dans (Ben Rabah et al., 2017). En effet, cette approche initialement proposée présente un cas en deux parties : *Problème* et *Solution*. La partie *Problème* est décrite par un ensemble de descripteurs où chacun est un triplet ( $Co_i$ ,  $V_t$ ,  $V_{t-1}$ ) avec  $Co_i$  est un capteur ou actionneur du SAP,  $V_t$  est sa valeur au cycle d'API ( $T$ ) et  $V_{t-1}$  est sa valeur au cycle d'API ( $T-1$ ).

Dans la série de tests, nous avons injecté les fautes décrites ci-dessus et nous avons noté le taux de décisions correctes et le taux de fausses alarmes pour les deux approches. Les résultats de diagnostic sont résumés dans la figure suivante (figure 7) :



**Figure 7 :** Comparaison des résultats de notre approche actuelle et l'approche initialement proposée dans (Ben Rabah et al., 2017)

Les résultats de simulations montrent bien que l'approche proposée fournit le résultat attendu dans la majorité des cas en offrant plus que 80% de taux de décisions correctes et aussi une amélioration des résultats par rapport à l'approche initialement proposée. Cette amélioration est due à la nouvelle représentation de cas qui permet de couvrir plus de fautes essentiellement les blocages des capteurs et des actionneurs à 0 et à 1.

Mais le problème posé dans cette approche est l'augmentation du nombre de cas. En effet, cette augmentation peut améliorer les performances de la réutilisation car il y a plus de chance de trouver un cas proche par contre le nombre de cas à considérer lors l'étape de recherche devient de plus en plus important. Une étape de révision s'avère donc nécessaire afin de contrôler l'augmentation de la base de cas tout en ne conservant que les cas de bonnes performances.

## 5 Conclusion

Les modes de raisonnement tels que le raisonnement à base de modèle ou à base de règles nécessitent pour être efficaces un modèle précis et approfondi du domaine. Dans le domaine du diagnostic industriel, il est difficile de construire les connaissances expertes nécessaires à ces types de raisonnement afin de couvrir toutes les situations critiques. Dans cet article, nous avons présenté un système de RàPC pour le diagnostic en ligne des SAP. Plusieurs travaux de recherche ont traités cette problématique en utilisant ce type de raisonnement. Cependant, les auteurs ne précisent pas la manière dont les bases de cas ont été créées dans leurs systèmes et généralement l'étape de formulation des cas est faite manuellement par un expert du domaine.

C'est pourquoi nous avons, dans nos travaux, accordé une importance particulière à la phase de construction de la base d'apprentissage. Cette phase génère automatiquement un ensemble de connaissances empiriques stockée dans une base de cas. A partir de cette base et d'une phase de raisonnement et d'apprentissage en ligne, il est ainsi possible de diagnostiquer en ligne le système surveillé, d'apprendre de nouvelles connaissances et de mettre à jour la base de cas.

L'utilisation de cette approche pour le diagnostic en ligne des SAP nous semble opportune et exploitable. Nous projetons donc d'étudier différentes pistes théoriques parmi lesquels nous pouvons citer : (1) la prise en compte des changements d'états des entrées/sorties qui ne possèdent pas nécessairement une relation de causalité entre elles (2) la définition d'un seuil de similarité afin de ne garder que les cas sources les plus similaires au nouveau cas et (3) la vérification de l'utilité des cas afin de limiter la mémorisation aux seuls cas ajoutant de la nouvelle connaissance au système.

## 6 Remerciements

Ce travail bénéficie du soutien du projet C-Roads France (2015-FR-TM-0378-S) financé par la commission européenne.

## 7 Références

- Boschert, S., and Rosen, R. (2016). *Digital Twin—The Simulation Aspect*. In *Mechatronic Futures* (pp. 59-74). Springer International Publishing.
- Cassandras, C. G., and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science and Business Media.
- Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9), 1555-1575.
- Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition: representation and algorithms. In *13th international joint conference on Artificial intelligence* (Vol. 1, pp. 166-172). Morgan Kaufmann Publishers Inc.
- Boufaied, A., Subias, A., and Combacau, M. (2002). Chronicle modeling by Petri nets for distributed detection of process failures. In *Systems, Man and Cybernetics, IEEE International Conference on* (Vol. 4, pp. 6-pp). IEEE.
- Bertrand, O., Carle, P., and Choppy, C. (2007). Chronicle modelling using automata and colored Petri nets. In *DX-07 18th International Workshop on Principles of Diagnosis*, (pp. 229-234), USA.

- Carle, P., Choppy, C., and Kervarc, R. (2011). Behaviour recognition using chronicles. In *Theoretical Aspects of Software Engineering (TASE), Fifth International Symposium on* (pp. 100-107). IEEE.
- Toguyeni, A.K.A., CRaye, E., and Gentina, J.C. (1991). A method of Temporal Analysis to perform on-line Diagnosis. In *Industrial Electronics Society (IECON'91), the context of Flexible Manufacturing System*, California, USA.
- Saddem, R., Toguyeni, A., and Tagina, M. (2011). A model-checking approach for checking the Consistency of a set of Causal Temporal Signatures. In *9th European Workshop on Advanced Control and Diagnosis* (pp. 1-6).
- Saddem, R., and Philippot, A. (2014). Causal Temporal Signature from diagnoser model for online diagnosis of Discrete Event Systems. In *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on* (pp. 551-556). IEEE.
- Cordier, M. O., and Dousson, C. (2000). Alarm driven monitoring based on chronicles. In *Proceedings of Safeprocess* (pp. 286-291).
- Aamodt, A., and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1), 39-59.
- Lieber, J. (2007). Application of the revision theory to adaptation in case-based reasoning: The conservative adaptation. In *International Conference on Case-Based Reasoning* (pp. 239-253). Springer Berlin Heidelberg.
- Kolodner, J L. (1993). *Case Based Reasoning*. Morgan Kaufmann Publishers Inc.
- Plaza, E. (1995). Cases as terms: A feature term approach to the structured representation of cases. In *International Conference on Case-Based Reasoning* (pp. 265-276). Springer Berlin Heidelberg.
- Donini, F. M., Lenzerini, M., Nardi, D., and Schaerf, A. (1996). Reasoning in description logics. *Principles of Knowledge representation*, 1, 191-236.
- Bergmann, R. (2002). *Experience management: foundations, development methodology, and internet-based applications*. Springer-Verlag.
- Lenz, M., Bartsch-Spörl, B., Burkhard, H. D., and Wess, S. (Eds.). (2003). *Case-based reasoning technology: from foundations to applications* (Vol. 1400). Springer.
- Aha, D. W., Breslow, L. A., and Muñoz-Avila, H. (2001). Conversational case-based reasoning. *Applied Intelligence*, 14(1), 9-32.
- Chougule, R., Rajpathak, D., and Bandyopadhyay, P. (2011). An integrated framework for effective service and repair in the automotive domain: An application of association mining and case-based-reasoning. *Computers in Industry*, 62(7), 742-754.
- Behbahani, M., Saghaee, A., and Noorossana, R. (2012). A case-based reasoning system development for statistical process control: Case representation and retrieval. *Computers & Industrial Engineering*, 63(4), 1107-1117.
- Ruiz, P. A. P., Kamsu-Foguem, B., and Noyes, D. (2013). Knowledge reuse integrating the collaboration from experts in industrial maintenance management. *Knowledge-Based Systems*, 50, 171-186.
- Dendani, N., Khadir, M., and Guessoum, S. (2012). Hybrid approach for fault diagnosis based on CBR and ontology: Using jCOLIBRI framework. In *Complex Systems (ICCS), 2012 International Conference on* (pp. 1-8). IEEE.
- Lejri, O., and Tagina, M. (2012). Representation in case-based reasoning applied to control reconfiguration. In *Industrial Conference on Data Mining* (pp. 113-120). Springer Berlin Heidelberg.
- Lesot, M. J. (2005). Similarity, typicality and fuzzy prototypes for numerical data. In *6th European Congress on Systems Science, Workshop Similarity and resemblance* (Vol. 94, p. 95).
- Bisson, G. (2000). La similarité: une notion symbolique/numérique. *Apprentissage symbolique-numérique*, 2, 169-201.
- Wilke, W., and Bergmann, R. (1998, June). Techniques and knowledge used for adaptation during case-based problem solving. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 497-506). Springer Berlin Heidelberg.
- Riera, B., Marangé, P., Gellot, F., Nocent, O., Magalhaes, A., and Vigario, B. (2010). Complementary usage of real and virtual manufacturing systems for safe PLC training. *IFAC Proceedings Volumes*, 42(24), 89-94.
- Rabah, N. B., Saddem, R., Hmida, F. B., Carre-Menetrier, V., and Tagina, M. (2017). Intelligent Case Based Decision Support System for Online Diagnosis of Automated Production System. In *Journal of Physics: Conference Series* (Vol. 783, No. 1, p. 012009). IOP Publishing.