A Knowledge Model for Dynamic Systems Monitoring

José A. Maestro, César Llamas and Carlos J. Alonso

Grupo de Sistemas Inteligentes
Dpto. de Informática
Universidad de Valladolid
{jose,cllamas,calonso}@infor.uva.es

Abstract A most common requirement in the development of Knowledge Based Systems in dynamic environments is the capability of expressing time. This paper presents how it is possible to express time related requirements on KBS tasks and to include time explicitly in rules. Such kind of facilities is attained using UML diagrams embedded in the usual COMMONKADS notation preserving the methodology. A significant set of tasks concerning monitoring is analyzed. Some specific lacks, in COMMONKADS, for the accurate analysis of these tasks in the time domain are identified; and the corresponding adaptations are presented. Finally, in order to express temporal elicited knowledge, a notation to include time in rules, focused on instants and intervals, is added.

Keywords: knowledge based system, real-time system, COMMONKADS, monitoring.

1 Introduction

The ability of expressing time in Knowledge Based Systems (KBS) is an essential feature when treating with applications devised to monitoring and diagnosing of continuous processes in manufacturing plants. Actually, several applications with these requirements have been developed in our workgroup to be applied in a beet sugar factory (AEROLID, TURBOLID, TEKNOLID) [3,2,4,1], some of them operating at this time.

In order to implement such kind of systems, a real time expert systems platform like G2 (Gensym) has been employed. This toolkit offers facilities to express knowledge in the form of frames, rules, procedures and so on. Also, as it is expected, this tool permits dealing explicitly with time, timed parameters and time triggered rules. In contrast, usual knowledge based methodologies do not include specifications for dynamic behaviours. Consequently, these methodologies only allow obtaining a static description of the system. As an example, COMMONKADS being a popular methodology, does not offers the adequate time notation facilities required in order to the problem solving methods be expressed in a temporal situation.

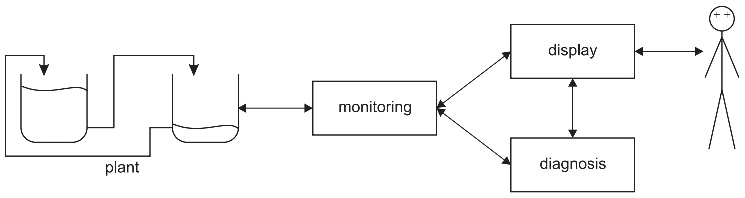


Figure 1. Location of the monitoring module in the application system.

It is not our opinion, that CommonKADS must change substantially to be a usable methodology in Soft Real-Time Knowledge Based Systems (Soft RT-KBS), although some attempts has been made to adapt the process to be able to deal with Hard RT-KBS [11,10].

Nowadays, it seems to be that UML should be an integral part of every modern specification. Therefore, special attention to software engineering methodologies that try to integrate UML and RT Systems [15,16,7] must be taken. Moreover, the integration with UML seems to be the course of action taken in the KADS methodology [13].

In this work, a knowledge based model for a monitoring task is presented. This task leans on a set temporal relations, and includes tasks that must be scheduled periodically. In its analysis, CommonKADS, some extensions for real-time [10] and UML are employed. The task is a model driven monitoring and the problem is expressed in terms not related with temporal reasoning, although some notation to include time in the rules is added. Moreover, the problems concerning the real time platform requirements and schedulability are not taken into account.

In the next section, a first approximation to the monitoring task is considered, in the framework described in [14,5,13]. In Section 3 the specification of the task in terms of CommonKADS augmented with the additional notation required is introduced. In the Section 4, a simple notation adequate for expressing time into rule is presented and employed into a rule type generalized to be used in the monitoring task.

2 The monitoring task

The monitoring task plays an important role in diagnosis and supervision systems [8]. Even more, it has a special status in the OLID generation of supervisory systems [1]. The main responsibility of a monitoring task is: observes the system evolution, in order determine whether exists an abnormal behaviour; Figure 1 shows the monitoring task location, as an isolated module previous to visualization and diagnosis modules.

The monitoring task, as is presented in Alonso *et al.* [4,12] could be described as in Figure 2. This task comprises two subtasks: *normal monitoring* and *intensive monitoring*, being, the former, a simple model driven monitoring, as mentioned by Breuker *et al.* [5, ch. 4]. The latter is a more specific model driven

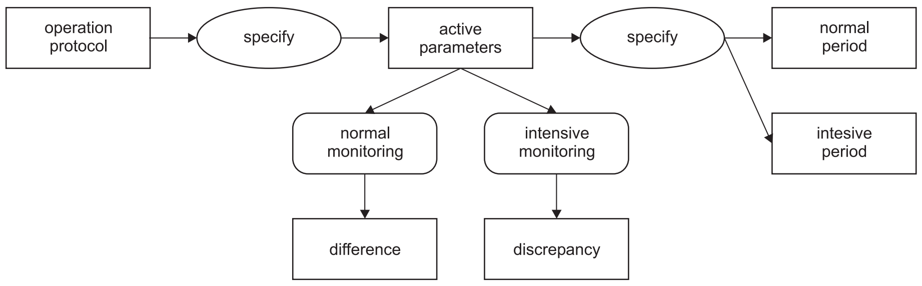


Figure 2. High level inference structure for the *monitoring* task.

monitoring task, more time consuming, and subtle to be triggered to confirm the existence of a discrepancy (*see* Alonso *et al.* [4]). Furthermore, each subtask has a specific execution period. The overall monitoring process integrates both modes; Figure 2 illustrates the monitoring task as can be seen by the rest of the system.

The notion of trajectory deviation considered is described in terms of knowledge representation involving the role *difference* and temporal constraints. Rules in both tasks, *normal* and *intensive* monitoring, contain references to thresholds and time deadlines on each parameter.

Either of these tasks is scheduled in a regular basis and its behaviour is described appropriately in terms of time intervals. Several situations could change the steady execution state of them: (*i*) a change of operation protocol, that convey a transient situation that require the monitoring be suspended, and possibly a different schedule rate when the normal operation is resumed, and (*ii*) a change of task scheduling from *normal* to *intensive* monitoring, and *vice versa.*

The monitoring, described in [4],relies on the concept of *monitored variable* that will be employed in the domain layer. A monitored variable could present three states: *normal*, *vigilance* and *critical*; its current state depends on roles *difference* and *discrepancy* and some temporal considerations.

The *intensive monitoring* begins to be scheduled in case the monitoring variable changes from the *normal* state. The effective distinction between *normal* and *intensive* monitoring makes possible to operate at different rates under each operation protocol. Intensive monitoring needs a higher scheduling rate; hence, in this way it is possible to invoke this task only when it is strictly necessary [12]. In addition, this task behaves in a way rather different from normal monitoring. Intensive monitoring has a different set of relations from the normal monitoring and interleaves, in some sense, with the diagnosis subsystem [3].

In Alonso *et al.* [4] a knowledge level description for the supervision and diagnosis, using CommonKADS [13], is presented. The monitoring model appears embedded, partially, within the *supervision and diagnosis* task.

In this description, the independence of the tasks from the specific elements of the application environment has been preserved (*e. g.* from time constraint). However, the representation obtained is far from being satisfactory, and presents some lacks: (*i*) the flow control is very complex, (*ii*) some constructors disrupt the sequential execution (fork, wait_until and break_if), and (*iii*) the inference structure becomes complex, possibly as a consequence of the previous reasons.

3 Analysis of the monitoring subtasks, with time

In a non-static domain, most of the knowledge involved has a dynamic flavour. There will exists some timing restrictions and dynamic and temporal relations, which are elicited knowledge and, therefore, must be included in a suitable manner in the knowledge model. The task description for the monitoring could be

```
TASK monitoring ;
  GOAL : "Analyze an ongoing process to find an abnormal behaviour." ;
  ROLES :
    INPUT :
      protocol: "Current system operation point." ;
    OUTPUT :
      discrepancy : "Indication of deviant system behavior." ;
  SPECIFICATION : "Watch the system evolution to discover whether any parameter
    behaves not according to system expectations." ;
END TASK

TASK-METHOD two-step-monitoring ;
  REALIZES : monitoring ;
  DECOMPOSITION :
    INFERENCES : specify ;
    TASKS :
      normal-monitoring, intensive-monitoring ;
  ROLES :
    INTERMEDIATE :
      active-parameters : "Set of parameters to observe the system evolution." ;
      parameter : "A parameter to be monitored." ;
      normal-period : "Period to monitor a parameter showing normal behavior." ;
      intensive-period : "Period to monitor a parameter showing abnormal behavior." ;
  CONTROL-STRUCTURE :
    specify (protocol → active-parameters) ;
    FOR-EACH parameter IN active-parameters DO
      specify (parameter → normal-period + intensive-period) ;
      ACTIVITIES one-parameter-monitoring DO
        normal-monitoring ( normal-period , parameter → difference ) ;
        intensive-monitoring ( intensive-period , parameter → discrepancy ) ;
      END ACTIVITIES
    END FOR-EACH
END TASK-METHOD
```

Figure 3. Task and method description for the *monitoring* task.

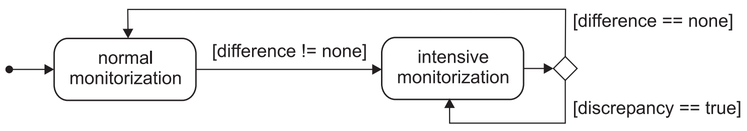


Figure 4. Activity diagram for *one-parameter-monitoring.*

enhanced noticeably if the schedule, for the normal and intensive monitoring of the set of parameters, could be expressed appropriately. Also, a more adequate representation for the control structure is required – in this case, an activity diagram in UML is enough.

In practice, good methodology for use in real-time systems design would contemplate the use of diagrams and other textual ways of expressing the tempo-

```
TASK normal-monitoring ;
  GOAL : "Analyze an ongoing process to find an abnormal behaviour." ;
  TYPE : periodic ;
  RELATIVE-TIME : Yes ;
  PERIOD : period ;
  ROLES :
    INPUT :
      parameter : "A signal which behavior is been analyzed." ;
      period : "Period which the task is realized." ;
    OUTPUT :
      difference : "Indication of deviant parameter behavior." ;
  SPECIFICATION : "Watch the parameter evolution to discover whether it behaves
      not according to system expectations." ;
END TASK

TASK-METHOD system-driven-monitoring ;
  REALIZES : normal-monitoring ;
  DECOMPOSITION :
    INFERENCES : compare, specify ;
    TRANSFER-FUNCTIONS : obtain ;
  ROLES :
    INTERMEDIATE :
      norm : "Expected normal value for the parameter." ;
      parameter-value : "Current value for the parameter." ;
  CONTROL-STRUCTURE :
    specify (parameter → norm) ;
    obtain (parameter → parameter-value) ;
    compare (parameter-value + norm → difference) ;
END TASK-METHOD
```

Figure 5. Task description for the *normal monitoring.*

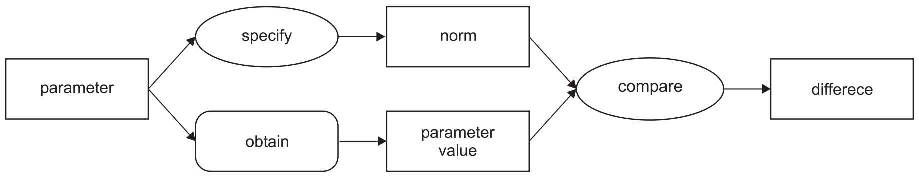


Figure 6. Inference structure for the *normal monitoring* task.

rization of the tasks, events and state changes in the system [6]. In the Figure 3, a task method specification for the monitoring task is presented. It employes a new primitive, ACTIVITIES, bounded to the activity diagram, *one-parameter-monitoring* that appears in Figure 4. This diagram permits us represent the activity states of the monitoring task, without tangling the control structure of the monitoring task.

The part of the control structure that can not be described appropriately using pseudocode has been replaced with an activity diagram. A more rigorous approach could be taken considering an additional task for one parameter.

Task timing constraints can be expressed with little changes in the task description presented in Figures 5 and 10 (*see* Appendix A). Tree additional fields –type, period, relative-time– are added to the standard notation of the task description, following the notation proposed by Henao *et al.* in [10,11].

4 Rule time notation

The special nature of the time knowledge relations makes difficult to grasp them into the usual CommonKADS rule notation. In our domain, there are temporal relations that must be expressed in the knowledge base – for example, in Figure 7 a complex state diagram is presented, which includes conditional and temporal transitions. Maybe, the usual state diagram is the most powerful tool

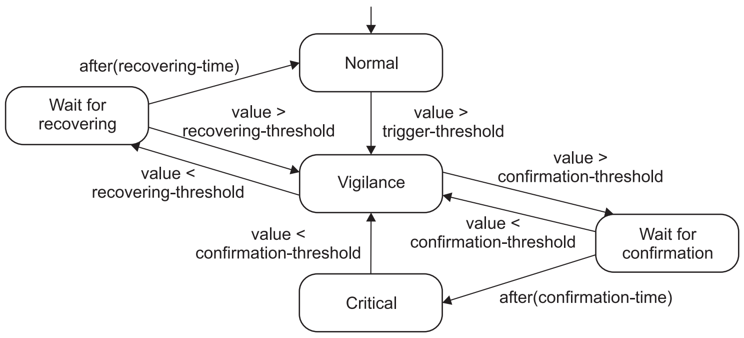

Normal
after(recovering-time)
Wait for recovering
value > recovering-threshold
value > trigger-threshold
Vigilance
value < recovering-threshold
value > confirmation-threshold
value < confirmation-threshold
value < confirmation-threshold
Wait for confirmation
Critical
after(confirmation-time)

Figure 7. State diagram for a monitored variable.

```
RULE-TYPE monitoring-rule ;
  ANTECEDENT: monitored-variable ;
    CARDINALITY: 1 ;
  CONSEQUENT: monitored-variable ;
    CARDINALITY: 1 ;
  CONNECTION SYMBOL: detect ;
END RULE-TYPE
```

Figure 8. The rule type declaration for monitoring rules.

for describing this kind of systems [15], being intuitive and permitting being formalist.

This state diagram represents all the necessary knowledge to identify a discrepancy and to recover from it. As can be seen, a monitored variable has three main states and two auxiliary wait states.

As CommonKADS stands a textual description for rules rather than a graphical one, it would be necessary to translate the temporal transitions into the standard rule notation.

For describing the rule content a kind of first order logic alike notation is proposed by Schreiber *et al.* in [13]. Whilst CommonKADS includes an almost entire Bnf notation definition of Cml2 (Concept Modelling Language), it does not include a specific definition for rule content (*see* [13, ch. 14]). It could be used as a way to extend the methodology to those kinds of problems that are not taken into account, such as dynamic domains.

In Figures 8 and 9 a rule-type and abstract rule instances are proposed. These rules reflect the state diagram shows in the Figure 7 into a CommonKADS rule

```
MV.state = normal AND MV.value > MV.trigger-threshold
  DETECT
MV.state = vigilance

MV.state = vigilance AND ALWAYS t IN [NOW - MV.t-confirmation, NOW]
      MV.historical(t) > MV.confirmation-threshold
  DETECT
MV.state = critical

MV.state = critical AND MV.value < MV.confirmation-threshold
  DETECT
MV.state = vigilance

MV.state = vigilance AND ALWAYS t IN [NOW - MV.t-recuperation, NOW]
      MV.historical (t) < MV.recovery-threshold
  DETECT
MV.state = normal
```

Figure 9. The rule set for the monitoring tasks. MV stands for "monitored variable".

description alike. A kind of temporal first order logic has been employed. This logic is inspired in the work of *Allen and others* compiled by Galton in [9].

Monitoring has only references to past and present time so that a linear time model is proposed. Instants and intervals are necessary to express our monitoring model. The usual FORALL quantifier has been replaced with a more expressive ALWAYS. This is an interval quantifier so that an interval must be provided. In [12] another quantifier, SOMETIME, and a set of predicates to deal with intervals and instants are provided . Additionally, it is supposed each monitored variable has an historical that maintains past data.

5 Conclusions

A model for a real-time task of a knowledge-based system has been presented. In this model CommonKADS and Uml have been employed in order to represent suitably the knowledge involved and the dynamic behaviour of the task. This integration has been made following Schreiber (*et al.*) latest CommonKADS [13] methodology book recommendations, and a *monitoring* task, that is a generalization of those used in the OLID generation, has been described.

It has been made clear that, if the system to be analyzed has a temporal description, the usual notation to describe the task method could not be the best choice. Furthermore, Real-Time systems demands its own concepts and notation, usually a graphical one, which is difficult to describe in pseudocode without yielding a procedural program.

Hence, an alternative to the method specification of the monitoring task has been used, in this case, the activity diagram. This alternative, founded in the Uml set of diagrams, has just been justified in the framework of the CommonKADS methodology. This kind of diagram allow us to describe sequences of tasks and the possible paralelism among them without having to write constructions as the usual fork-join couple.

State diagrams have been employed in wherever situations the expressivity could be enhanced. In this paper, apart from the activity diagram, a state diagram has been presented for the monitored variable. This diagram, also, solves partially the problem of expressing time relations in contrast to the pseudocode form.

The rule notation has been augmented to express time dependencies. This has permitted to simplify the task description. Although the time model is simple, covers a broad range of applications.

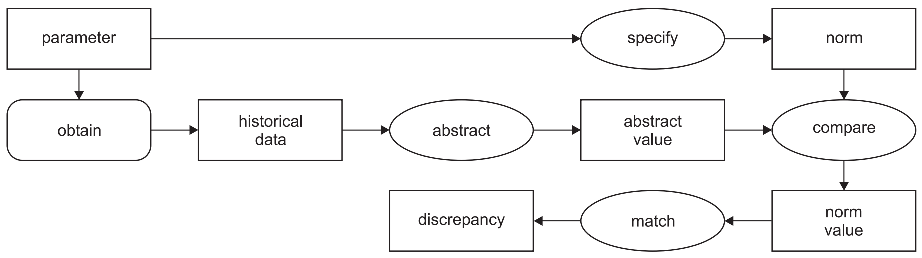
References

1. G. Acosta, C. Alonso, and B. Pulido. Basic Tasks for Knowledge Based Supervision in Process Control. *Engineering Applications of Artificial Intelligence*, 14:441–455, 2002.
2. C. Alonso, G. Acosta, J. Mira, and C. de Prada. Knoledge based process control supervision and diagnosis: the AEROLID approach. *Expert Systems with Applications*, 14:371–383, 1998.

3. C. Alonso, B. Pulido, and G. Acosta. On Line Industrial Diagnosis: an attempt to apply Artificial Intelligence techniques to process control. In *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98. LNAI*, volume 1415, pages 804–813. Springer-Verlag, 1998.
4. C. Alonso, B. Pulido, G. Acosta, and C. Llamas. On-line Industrial supervision and diagnosis, knowledge level description and experimental results. *Expert Systems with Applications*, 20(2):117–132, February 2001.
5. J. Breuker and W. Van de Velde, editors. *CommonKADS Library for Expertise Modelling. Reusable problem solving components*, volume 21 of *Frontiers in Artificial Intelligence and Applications.* IOS Press, Amsterdam, 1994.
6. Alan Burns and Andy Wellings. *Real-Time Systems and Their Programming Languajes.* Addison-Wesley, 3 edition, 2001.
7. Bruce Powel Douglass. *Real-Time Uml: developing efficient objects for embedded systems.* Addison Wesley Logman, 3 edition, May 1998.
8. O. Dressler and P. Struss. *Principles of knowledge representation*, chapter The consistency based approach to automated diagnosis of devices., pages 269–314. CSLI publications, 1996.
9. Anthony Galton. *Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter Time and Change for AI, pages 175–240. Oxford Science Publications, 1995.
10. M. Henao, J. Soler, and V. Botti. Developing a Mobile Robot Control Application with CommonKADS-RT. In *Engineering of Intelligent Systems. LNAI*, volume 2070, pages 651–660. Springer, June 2001.
11. Mónica Henao. *CommonKADS-RT: Una Metodología para el Desarrollo de Sistemas Basados en el Conocimiento de Tiempo Real.* PhD thesis, Universidad Politécnica de Valencia, June 2001.
12. J. A. Maestro, C. Llamas, and C. Alonso. Anotación de aspectos temporales en la especificación de la tarea en CommonKADS: Aplicación a la monitorización. In *IX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA-TTIA'01)*, pages 449–458, Gijón, España, 2001.
13. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. *Knowledge Engineering and Management, The CommonKADS Methodology.* The MIT Press, 1999.
14. G. Schreiber, B. Wielinga, and J. Breuker. *KADS. A Principled Approach to Knowledge-Based System Development.* Academic Press, 1993.
15. Bran Selic, Garth Bullekson, and Paul T. Ward. *Real Time Object-Oriented Modeling.* Software Engineering Practice Series. John Wiley & Sons, New York, 1994.
16. Bran Selic and James Rumbaugh. Using UML for Modeling Complex Real-Time Systems. Whitepaper, ObjecTime Limited, March 1998. http://www.objectime.com/otl/technical/umlrt.html.

A Analysis of the intensive monitoring

In this appendix, the corresponding analysis for the *intensive monitoring* task is described. It presents a task with similar time constraints specification as the *normal monitoring.*



```
TASK intensive-monitoring ;
  GOAL : "Analyze an ongoing process to find an abnormal behaviour." ;
  TYPE : periodic ;
  RELATIVE-TIME : Yes ;
  PERIOD : period ;
  ROLES :
    INPUT :
      parameter : "A signal which behavior is been analyzed." ;
      period : "Period which the task is realized." ;
    OUTPUT :
      discrepancy : "Any classification of abnormal behavior of the system being
        monitored." ;
  SPECIFICATION : "Watch the parameter evolution to confirm whether it behaves
      not according to system expectations." ;
END TASK

TASK-METHOD temporal-abstraction-monitoring ;
  REALIZES : intensive-monitoring ;
  DECOMPOSITION :
    INFERENCES : abstract, compare, match, specify ;
    TRANSFER-FUNCTIONS : obtain ;
  ROLES :
    INTERMEDIATE :
      norm : "Expected normal value for the parameter." ;
      abstract-value : "Some kind of abstracted result from historical data." ;
      historical-data : "Collection of past values for the parameter." ;
      norm-value : "Truth value that indicate whether a norm is fulfilled." ;
      norm-values : "Set of norm values." ;
  CONTROL-STRUCTURE :
    obtain (parameter → historical-data) ;
    WHILE HAS-SOLUTION abstract (historical-data → abstract-value) DO
      specify (parameter + abstract-value → norm) ;
      compare (abstract-value + norm → norm-value) ;
      norm-values = norm-values ADD norm-value ;
    END WHILE
    match (norm-values → discrepancy ) ;
END TASK-METHOD
```

Figure 10. Inference structure and task description for the *intensive monitoring*.