구조물 모니터링 및 진단을 위한 지식모델의 개발

A Hybrid Knowledge Model for Structural Monitoring and Diagnosis

김 성 곤*
Kim, Sung-Kon

요 약

구조물 모니터링 시스템의 전산환경을 구성하기 위해 필요한 지식 및 정보를 파악하고 이를 지식기반화하는 방법을 제시하였다. 전산환경의 구축을 위한 정보로는 센서 및 하드웨어, 신호처리, 그리고 손상발견/평가를 위한 지식등이 필요한데, 이들은 모두 다른 형태의 지식이므로,-즉 수학 연산, 서술적 지식, 수치모델 등-어느 특정의 모델링 기법 단독으로는 이들을 효과적으로 수용하기가 매우 어렵다. 이를 해결하기 위하여 객체지향적 모델링기법과 논리언어를 혼합 사용하는 방법(Hybrid Modeling Paradigm)이 제시되었고, 이의 타당성 및 효율성 검증을 위해 모델구조물을 이용한 예제를 수행하였다.

Abstract

A hybrid knowledge model which amalgamates an object-oriented modeling approach and logic programming implementation is presented for structural health monitoring and diagnosis of instrumented structures. Domain knowledge in structural monitoring and diagnosis is formalized and represented in a logic-based object-oriented modeling environment. The model and environment have been implemented and illustrated in the context of a laboratory case study of damage detection in a successively damaged steel structure.

Keywords : structural monitoring, knowledge model, object-oriented model , logic language

1. Introduction

There is a need to devise appropriate computational abstractions and support environments for health monitoring and condition assessment of instrumented structures[1,2].

Such structures contain a network of sensors which sense important measurands related to current condition of the structure and relay signals to a computer for processing, interpretation and recommendations. In order to

* 정회원 · 서울대학교 토목공학과 연수연구원, 공학박사

• 이 논문에 대한 토론을 1996년 12월 31일까지 본 학회에 보내주시면 1997년 6월호에 그 결과를 게재하겠습니다.

develop comprehensive computational environments for these purposes, a *model* of the information describing the system is required. This model must support a meaningful and computable representation of the components and their complex inter-relationships that are characteristic of engineering systems, such as physical configurations, sensors, signal, and diagnostic knowledge.

Since the modeling directly affects the analytical capability, flexibility and efficiency of the reasoning mechanism, the development of a suitable model is the key issue in the development of a computational environment for structural monitoring and diagnosis[3]. The object-oriented approach is an active focus of information modeling efforts for engineering domains. This modeling paradigm has been demonstrated to be well suited to represent structural systems based on hierarchical description and abstraction. The inheritance and encapsulation mechanism also enhance program structure and maintainability. However, object-oriented environments lack a deductive retrieval capability and pattern matching that are basic to most knowledge-based applications. On the other hand, logic programming languages like Prolog possess deductive retrieval capability through backtracking and pattern matching via unification. A diagnostic operation often involves deductive reasoning to infer the best or plausible explanations for given symptoms. Therefore, the need for a kind of hybrid paradigm, e.g., combining logic and object-orientation, is indicated to develop a model for a structural monitoring and diagnosis system.

A hybrid model combining object-oriented and logic approaches for the structural monitoring and diagnosis domain is presented in this paper. Domain knowledge in structural monitoring and diagnosis is formalized and represented in a logic-based object-oriented modeling environment. The model and environment have been implemented and illustrated in the context of a laboratory case study of damage detection in a successively damaged steel truss structure.

2. Domain Analysis: Structural Health Monitoring

Health monitoring operation for a structural system is considered to consist of three distinct tasks: identifying damage, locating the damage, and assessing of condition (severity and seriousness of damage). The variety of external excitations, including load effects and environmental influences such as temperature changes and corrosive environments, results in damage of the structure over time. This damage, which reflects the current condition of

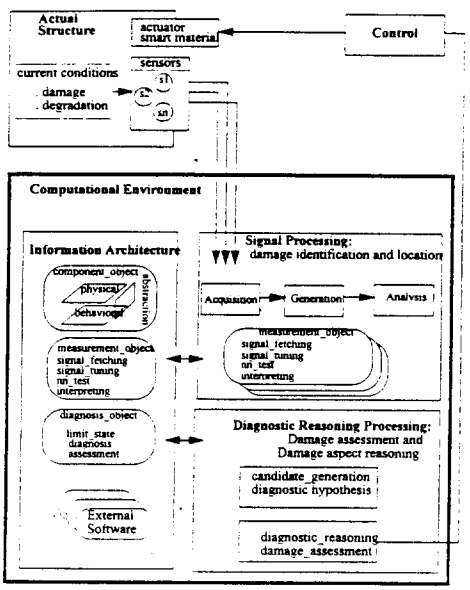

Fig. 1 Structural health monitoring

the structure, is sensed and considered herein to be transmitted reliably to a data acquisition system, where sensed voltage fluctuation data are converted to specific performance parameters such as acceleration, strain, acoustic emission, displacement, or temperature. In order to identify and locate the damage such data must be processed and further interpreted for use in reasoning about the current condition of the structure.

This operation, as depicted in Figure 1, requires a comprehensive computational environment which utilizes diverse types of data and knowledge. In order to develop such a comprehensive computational environment for these purposes, a *model* of the information describing the system is required. This model must support a meaningful and computable representation of the components and their complex inter-relationships that would be characteristic of smart structural systems, such as physical information, sensors, signals, and diagnostic knowledge. Particularly, regarding the required characteristics for health monitoring, that model should :

- contain both physical and behavioral information about the structural system and its principal components,
- contain sensor and signal information,
- support on-line signal processing to handle the avalanche of signals produced by a multi-channel instrumented civil structure,
- support health monitoring operation at various abstraction levels.

3. Model Development Considerations

Model development activity starts with investigation of the domain problem and the actual goals to be achieved by specific engineering tasks, and ends in a validated model. Some considerations involved in developing the domain model are as follows:

- Context of domain : health monitoring operation requires various types of knowledge.
- Type of model : for diagnosis purposes the model should be either a correct or fault model. A correct model represents normal behavior, while a fault model reflects abnormal behavior.
- Abstraction : health monitoring requires navigation at various abstraction levels of the model.
- Representation scheme : selecting appropriate knowledge representation scheme is important because the problem solving strategy heavily depends on the representation chosen.

4. Representation Schemes

In order to satisfy the various requirements for knowledge representation, different techniques should be used for different types of knowledge[4]. Selecting an appropriate representation can have the impact of either rendering the system successful or a failure. The knowledge representation scheme can effectively limit what the system can perceive, know, or understand[5].

The typical knowledge representation schemes are classified into : procedural[6], network-based[7], frame-based[8], and object-oriented[9,10]. Details for these representation schemes are introduced in articles by[11]. These representation schemes are evaluated on the basis of considerations such as the following[5,12] :

- Expressiveness : the representation scheme needs to make all of the important

distinctions among the relevant concepts.

- Structural representation : the representation scheme needs to be compact and clear but still sufficiently robust to support all the required functions.
- Computational efficiency : the representation scheme needs to facilitate efficient computation of various inferences required for the task.
- Representation uniformity : Different types of knowledge can be uniformly represented, and therefore can be logically consistent, in the same representation scheme.
- Modifiability : Represented knowledge needs to be easily modifiable.

The object-oriented scheme is a preferred scheme for modeling necessary knowledge. However, the object-oriented scheme has some shortcomings when it is applied alone to a complex engineering problem such as health monitoring. In order to address this shortcoming, using a hybrid knowledge representation which is combining the object-oriented paradigm with the logic paradigm is proposed and applied.

5. Hybrid Knowledge Representation

It is recognized that a health monitoring system requires various kinds of knowledge and reasoning, e.g., hierarchical object description, inheritance, rule-based diagnostic knowledge, data-driven signal monitoring and goal-driven assessment[2]. Since one uniform approach is too rigid or inflexible to support sufficiently comprehensive representational and reasoning needs, some kind of hybrid paradigm is naturally required.

Object-oriented representation/programming is particularly suitable for problems in which underlying knowledge can be categorized hierarchically. However, this approach suffers from the following limitations when it is applied alone to a complex engineering problem such as health monitoring.

- no deductive retrieval and pattern matching that are basic to diagnostic reasoning.
- limited ability to refine, add or delete, the attributes or methods during run-time.
- lack of nondeterminism.

On the other hand, a logic programming language like Prolog[13] has built-in facilities for compensating the drawbacks of object-oriented paradigm, while also having its own deficiencies : Its declarative style performs poorly for tasks that are implicitly procedural in nature, such as data-driven inference which is appropriate for monitoring tasks. In this work, the identified knowledge is formalized and represented in the context of the object-oriented modeling scheme[10], and implemented in Prolog[14].

6. Representation of Domain Knowledge

Based on the notion of class in the object-oriented concept, three types of object groups which will be represented as classes are selected. In health monitoring operation, structural components(for both physical and behavioral information), signals, and knowledge for assessment play key roles. Spatial and other physical information about a structural system is represented in the *ComponentClass*. Any physical components are represented and aggregated into an assembly by virtue of this class information. *MeasurementClass* is a super class of any specific measurand class(e.g., acceleration or strain) which monitors and processes the

sensor signal obtained from sensory devices. Assessment and diagnostic knowledge is represented in the DiagnosisClass. An instance of this class represents a domain-specific diagnostic algorithm. Behavioral information are collected in the name of specific structural components. In order to represent the specific function and behavior of physical components, *BehaviorClass is introduced.* These are shown in Figure 2.

Fundamental building blocks in the formal language to represent the various knowledge are the definition of the class, instance, relationship, attributes, and methods. A class is described in terms of five properties : ClassName, SuperClassName, relationship, {Attributes}, and [Methods] as shown in Figure 3 . It is necessary to provide explanation for conventions and guidelines /restrictions for naming classes, objects, attributes and methods in this representation[15]. There are also certain restrictions and guidelines for selecting and using names of the fundamental building blocks as follows.

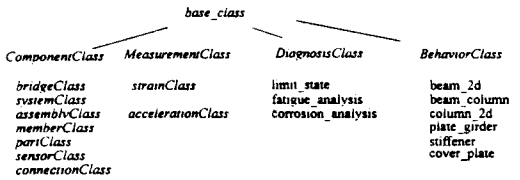

Fig. 2 Types of classes

```
class ComponentClass is_a base_class    % class, superclass name and relationship
    instance_variables                  % attributes
        [publics]:
        length(Length):
        perspective_view(Levels):
        material(MaterialName):
        material_grade(Value):
    methods                             % methods
        [publics]:
        get_property:
        draw_component:
        damage_id:
        damage_location:
        damage_assessment:
        report
end_class.
```

Fig. 3 Class representation -*Cinoibebt Class*

- class name : class names selected in this representation imply names of structural components, or functions of specific knowledge. Since these names are unique and imply a certain knowledge in this representation, naming conventions must be crafted carefully.
- object name : an object is an instance of specific class. Since each class imply unique state and behavior (i.e., attributes and methods), it is important to map proper class to an object during object creation.
- attribute name : name for attribute value is based on general names of structural components widely used and specific names of parameters which are necessary for monitoring operations. Relationship names are strictly used for inheritance or other associations between classes. Therefore, attribute values for relationships are limited to names of classes or objects.
- method name : a method is an operation of a class in which one class invokes actions in another. Therefore, methods are limited to the scope and purpose of the domain specific knowledge representation. In this work, since the main operation is focused on health monitoring, method names are mainly selected based on this category.

Based on this formal representation syntax, a class ComponentClass is introduced here and other classes are described elsewhere[3] in detail. An information architecture for a structural system is considered to consist of a number of named component objects which contain information about themselves such as geometry, topology, and some general information such as material type and maintenance history. These components are collected into various physical categories in

order to support reasoning at different levels of representation.

ComponentClass is represented with its particular set of attributes and methods as follows.

```
class ComponentClass is_a base_class
        {length :: Value} :
        {perspective_view :: Value} :
        {material :: Value} :
        {material_grade :: Value} :
        {has_connection :: Value}
        [get_property] :
        [draw_component] :
        [damage_id] :
        [damage_location] :
        [damage_assessment] :
        [report]
end_class
```

In addition to the explicit attributes defined within the class and the inherited attributes from *base_class*, a number of the asserted attributes are used while performing methods. For example, while executing [draw_component], incidences and nodes information, linked component names, and coordinates of nodes are used and asserted as attributes of the underlying component.

As key methods of ComponentClass, [damage_id], [damage_location], and [damage_assessment] are for performing health monitoring operations.

The contents of a method [damage_id] are as follows.

```
[damage_id] ::-
  〈obtain_corresponding_sensors〉,
  〈obtain_corresponding_measurement_object〉,
  〈send_message_to_measurement_object〉,
  〈store_returned_results〉.
```

When in a component object which is under investigation [damage_id] is invoked to identify any damage, it sends a message to a corresponding measurement object to monitor incoming signals. Results of signal monitoring returned from the measurement object are then asserted as an attribute as follows :

```
{identified_damage :: (Component_name,
DamagedOrNot)}
```

Where Component_name is the name of the object under investigation, and DamagedOrNot is a symbolic value indicating either "damaged" or "no_damage". This asserted value is used subsequently in operations such as [damage_location] and [damage_assessment].

7. Hybrid Implementation

A logic-based object-oriented computational environment to support the system depicted in Figure 1 is designed and implemented using Quintus Prolog. This environment includes class library, class interpreter, message operator, and object modeler[15].

Fundamental building blocks in the formal language to represent the various knowledge are the definition of the class, instance, relationship, attributes, and methods. As shown in Figure 4, any classes are defined using the class name, superclass name, relationship, attributes, and methods. Between "class" and "end_class.", class name and relationship with superclass are defined first, and then

variables and methods on this class level are defined. All class representations are interpreted into internal Prolog facts and rules when the class library is loaded into the interpreter[15].

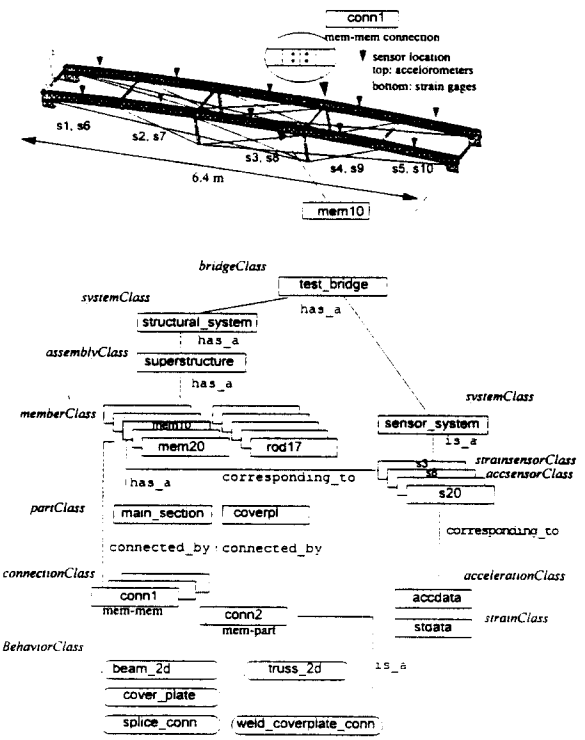

Fig. 4 Bridge model and object data model test_bridge

A method is a clause in the logic programming sense : it consists of a head and an optional body, expressed in normal Prolog syntax as : head : - ⟨body⟩. Here head is the name of the method and body is a series of goals to be satisfied. For example, the hypothesis generation rule for damage cause in the damage_assessment method of a *ComponentClass* is as follows.

```
hypotheses_generate(ObjName) :-
validate_rule(Hypotheses,ObjName). %loop
...
validate_rule(fatigue-analysis, Obj) : -
    %rule for fatigue
  component_type_check(Obj),
  damage_check(Obj),
  damage_aspect(Obj),
  has_connection(Obj).

validate_rule(corrosion, Obj) : -
    %rule for corrosion
...
```

An object performs when a method is invoked via "message passing". A message passing has two arguments, the receiver and message, and can be written(16) as : send(Receiver, Message), where Message is used either for carrying an instance variable or invoking a certain method.

8. Case Study

A health monitoring scenario, particularly involving damage identification, location and assessment with diagnosis, is investigated using the vibration signals from a series of real and simulated experiments performed on an instrumented scale model bridge. Two kinds of vibration signals, acceleration and strain, are obtained under a number of damage scenarios created by imposing saw cuts on the model bridge. Using the obtained series of signals within the information architecture, monitoring operations to locate and assess damage are performed in the developed system. Three discrete tasks involved in this case study are: i) a series of real and simulated experimental tests to obtain signals, ii) training and testing of the neural networks employed for signal processing to identify and locate damage and quantify its severity[17], and iii) monitoring

operations in the developed computational environment.

A health monitoring system, implemented in Prolog and linking with external software, can be defined by combining hardware components with the developed computational environment. The elements of this system include sensors, data acquisition and conditioning system, signal processors(client processors), and main processor. A number of software modules, e. g. , class library, class interpreter, and operator are compiled based on Quintus Prolog Library and linked with the external software into an executable program comprising the computational environment.

The health monitoring operation is specified into three stages, i. e. , object data modeling (creating), signal processing(damage identification and location), and damage assessment. Each event can be performed in either different processes or in a single process. Every task in the computational environment is performed in a graphic-based interface window which is provided by HOOPS software. Task descriptions in each stage are described in detail elsewhere[15].

A message is sent from a structural component object which was created in the system based on the developed computational environment to a measurement object in client processor 1 to do monitoring of acceleration signals from the corresponding sensors to identify damage. The measurement object(acceleration) retrieves applicable network information such as number of input nodes, signal type(time-domain or frequency domain), signal format(row or column vector), and sampling rate, so that the incoming signal is tuned to fit the trained networks. Then the incoming signals are tested by invoking the applicable neural networks. Test results for the incoming signal set are interpreted into symbolic format, e. g. , damaged, normal, unidentified, and sent back to the component object. The component object invokes the measurement object(strain) in client processor 2, if the message from the client processor 1 indicates damage, to pinpoint possible damage location. Based on the results the component object then invokes diagnostic reasoning to infer likely damage cause and assess the severity and seriousness of the damage. The real bridge model and bridge data model created in the system are shown in Figure 4, and an example message passing is illustrated in Figure 5.

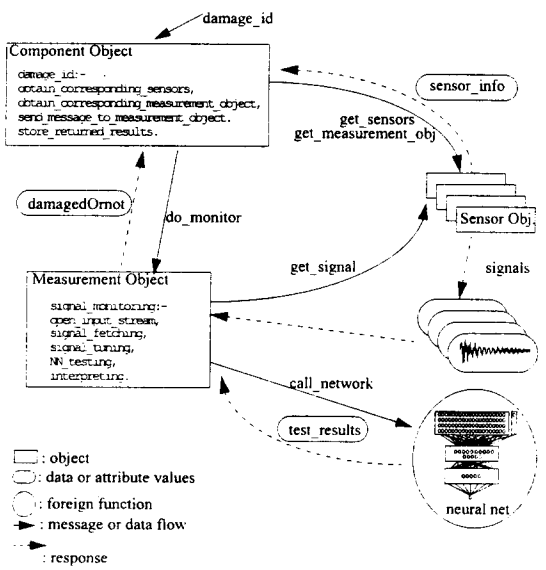

Fig. 5 Message passing during health monitoring

9. Conclusion

A hybrid data model which amalgamates an object-oriented modeling approach and logic programming implementation is presented for health monitoring and condition assessment of instrumented structures. Based on the results

the case study, the proposed hybrid model and resulting computational environment are considered to be a viable foundation of an eventual smart structural health monitoring system. The object-oriented approach is suited to represent various types of knowledge which are necessary in health monitoring operation. Development of a hybrid implementation paradigm reveals a number of advantages which would not be obtained from any single paradigm.

References

1. Chen, S. S. , 1991, "Towards a Computational Environment for Smart Structures", ASCE Structures Congress, Indianapolis, May 1991, pp 42-45.
2. Chen, S. S. and S. Kim, 1995, "Information Architecture Considerations for a Smart Structural System", Civil Engineering Systems, Vol. 12, pp. 1-19.
3. Kim, S., 1994, *Hybrid Knowledge-Based Computational Environment for Smart Structural Health Monitoring, Ph. D. Dissertation, Dept. of Civil Engineering, State University of New York at Buffalo, NY.*
4. Winston, P. H., 1984, *Artificial Intelligence, Addison-Wesley, 1984*
5. Woods, W. A., 1983, "What is important about knowledge representation", IEEE Computer, Oct. 1983
6. Buchanan, B. G. and E. H. Shortliffe, (eds.), 1984, *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, Mass., 1984.*
7. Brachman, R. J, and H. J. Levesque, (eds.), 1985, *Readings in Knowledge Representation, Morgan Kaufmann, 1985.*
8. Minsky, M., 1975, "Framework for Representing Knowledge", P. Winston, eds., *The Psychology of Computer Vision, McGraw-Hill, 1975.*
9. Cox, B., 1986, *Object Oriented Programming, Addison-Wesley, MA, 1986.*
10. Booch, G., 1991, *Object-Oriented Design: with Applications, The Benjamin/Cummings Pub., California.*
11. Brodie, M., M. Mylopoulos, and J. Schmit. (eds.), 1984, *On Conceptual Modeling, Springer-Verlag, NY, 1984.*
12. Bobrow, D. G., 1977, "Panel Discussion on AI", Proceedings of IJCAI, 1977
13. Bratko, I., 1990, *Prolog Programming for Artificial Intelligence, Second edition, Addison-Wesley.*
14. Fromherz, M. P. J., 1993, *OL(P) : Object Layer for Prolog-Reference Manual, Xerox Corp.*
15. Kim, S. and S. S. Chen, 1994, *Logic-based Object-Oriented Class Library Specifications for Structural Health Monitoring and Diagnosis, Technical Report, AAIE-94-01, Graduate Group for Applied AI in Engineering, School of Engineering and Applied Sciences, State University of New York at Buffalo, 1994.*
16. Stabler, E. P., 1986, "Object-Oriented Programming in PROLOG", AI Expert, Oct. 1986, pp. 46-57.
17. Chen, S. S. and S. Kim, 1995, "Automated Signal Monitoring using Neural Networks in a Smart Structural Systems", Journal of Intelligent Materials and Structures, Vol. 6, July 1995, pp. 509-515.

(접수일자 : 1996. 5. 17)