Expert Systems with Applications 39 (2012) 9031–9040

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An integrated architecture for fault diagnosis and failure prognosis of complex engineering systems

Chaochao Chen [a,*], Douglas Brown [a], Chris Sconyers [a], Bin Zhang [b], George Vachtsevanos [a], Marcos E. Orchard [c]

[a] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b] Impact Technologies LLC, Rochester, NY 14623, USA
[c] Electrical Engineering Department, University of Chile, Av. Tupper 2007, Santiago, Chile

ARTICLE INFO

Keywords:
Fault diagnosis
Failure prognosis
.NET framework
Particle filtering
Bayesian theory
Software architecture

ABSTRACT

Complex engineering systems, such as aircraft, industrial processes, and transportation systems, are experiencing a paradigm shift in the way they are operated and maintained. Instead of traditional scheduled or breakdown maintenance practices, they are maintained on the basis of their current state/condition. Condition-Based Maintenance (CBM) is becoming the preferred practice since it improves significantly the reliability, safety and availability of these critical systems. CBM enabling technologies include sensing and monitoring, information processing, fault diagnosis and failure prognosis algorithms that are capable of detecting accurately and in a timely manner incipient failures and predicting the remaining useful life of failing components. If such technologies are to be implemented on-line and in real-time, it is essential that an integrating system architecture be developed that possesses features of modularity, flexibility and interoperability while exhibiting attributes of computational efficiency for both on-line and off-line applications. This paper presents a .NET framework as the integrating software platform linking all constituent modules of the fault diagnosis and failure prognosis architecture. The inherent characteristics of the .NET framework provide the proposed system with a generic architecture for fault diagnosis and failure prognosis for a variety of applications. Functioning as *data processing*, *feature extraction*, *fault diagnosis* and *failure prognosis*, the corresponding modules in the system are built as .NET components that are developed separately and independently in any of the .NET languages. With the use of Bayesian estimation theory, a generic particle-filtering-based framework is integrated in the system for fault diagnosis and failure prognosis. The system is tested in two different applications—bearing spalling fault diagnosis and failure prognosis and brushless DC motor turn-to-turn winding fault diagnosis. The results suggest that the system is capable of meeting performance requirements specified by both the developer and the user for a variety of engineering systems.

1. Introduction

A reliable and real-time fault diagnosis and failure prognosis system constitutes the major condition monitoring and assessment block of Condition-Based Maintenance (CBM) and Prognostics and Health Management (PHM). Its four principal components are: *data processing*, *feature extraction*, *fault diagnosis* and *failure prognosis* (Vachtsevanos, Lewis, Roemer, Hess, & Wu, 2006). Research over the past years has focused particularly on the development and application of algorithms and tools for these modules in a variety of engineering systems (Chen, Vachtsevanos, & Orchard, 2010; Chen, Zhang, & Vachtsevanos, 2012; Chen, Zhang, Vachtsevanos & Orchard, 2011; Heng et al., 2009; Hillerstrom, 1996; Orchard & Vachtsevanos, 2009; Wang, 2008).

As CBM and PHM technologies continue to mature and algorithms are developed and tested to meet specified requirements, the military and industrial sectors are eager to witness a transitioning phase that will bring these technologies on-board their critical assets. In order to accomplish this crucial step, additional technologies must be developed to assist the transitioning phase. For example, verification and validation (V&V) tools are essential if diagnostic and prognostic modules are to be qualified for air worthiness or other important application domains. Amongst them is the need for integrating frameworks that will aggregate efficiently and effectively the diverse components of the PHM architecture.

Few attempts have been reported on the development of a generic, modular and flexible software architecture that integrates effectively and efficiently diagnostic and prognostic routines

* Corresponding author. Tel./fax: +1 404 894 4130.
E-mail address: chaochao.chen@gatech.edu (C. Chen).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.02.050

(Propes et al., 2002; Zhou et al., 2005). However, there is a growing need for such an architecture since the system developer is continually expected to produce new and improved algorithms for system components more efficiently and modularly and to integrate these algorithms with existing ones more easily and seamlessly; the system user, on the other hand, would prefer the ‘‘pushbutton’’ approach, that minimizes the effort for both the software installation and usage without the need for recoding, recompilation and redebugging.

From the standpoint of the system developer, the ideal architecture must possess the following features:

(1) Modularity—Each system component is established as an individual module and utilized independently.
(2) Flexibility, including
    a. Flexibility in system components update and integration.
    b. Flexibility in programming languages.
(3) Interoperability—The system is able to access the functionalities that are implemented outside the current software environment.

From the point of view of the system user, the following additional features are desirable:

(1) ‘‘Pushbutton’’—Only minimum effort and expertise required in system usage. There are no advanced skills or training needed to use the system.
(2) Simplified deployment—The system is deployed easily on the target computers.
(3) Real-time—The system is implemented in real-time.
(4) Friendly Graphical User Interface (GUI).
(5) Easy update—The algorithms and parameters of each system component are updated conveniently, and additional components can be added without changing the existing ones.

The inherent features of the .NET framework (Chen, Brown, Sconyers, & Zhang, 2010), including modularity, interoperability, programming language independence, simplified deployment and a base class library with a large range of functions, meet the performance requirements stated above from both the developer's and the user's standpoint. Details are described in the sequel.

A generic Bayesian state estimation technique, called particle filtering, has been developed and will be employed, in combination with real-time measurement and fault modeling, to implement the proposed .NET fault diagnosis and failure prognosis (Orchard, 2006; Patrick, 2006). It is well known that the Bayesian estimation algorithms are appropriate to solve the problems of real-time state estimation, since they can incorporate process data into the *a prior* state estimation by considering the likelihood of sequential measurements. As a recursive Bayesian algorithm, particle filtering is the sequential Monte Carlo method that can use any state-space fault models to estimate and predict the behavior of a faulty system.

The .NET framework is utilized in this work as the foundation of the system architecture, and the four system modules, *data processing*, *feature extraction*, *fault diagnosis* and *failure prognosis*, are built as the .NET components. A general particle-filtering-based framework is integrated in the system to achieve the real-time fault diagnosis and failure prognosis. The system is tested in two different types of engineering systems and the results are discussed.

The remainder of this paper is organized as follows: In the next Section, the .NET framework is introduced. Section 3 presents the architecture of the system, and a generic particle-filtering-based framework for fault diagnosis and failure prognosis is described. Two cases, bearing spall fault diagnosis and failure prognosis and brushless DC motor turn-to-turn winding fault diagnosis, are studied in Section 4. Finally, Section 5 provides a few concluding comments.

2. .NET framework

The .NET framework is a windows-based software framework provided by Microsoft, which has been regarded widely as a paramount technology in the software world. All other Microsoft technologies in the future are believed to be depended on it. The framework possesses two main components: the common language runtime (CLR) and the .NET framework class library.

Functioning as a software execution environment, the CLR is the foundation of the .NET framework providing a large range of services, such as memory management, thread management, enhanced security, debugging and profiling services, exception handling, and code checking and compilation, to aid the system developer during the creation of .NET components. The class library is a comprehensive, object-oriented code collection that can be combined with the developer's own codes to be used in a variety of applications. Note that this class library can be accessed from any of the .NET languages. Therefore, what all programmers have to learn is only this single library even if different programming languages, such as Visual Basic .NET and Visual C#.NET, are utilized.

Fig. 1 shows the architecture of this framework. The source codes written in any of the .NET languages are compiled into Microsoft Intermediate Language (IL) that is processor independent and can be portable to a number of platforms. A variety of instructions are included in IL for different types of operations, such as loading, storing, initializing, and calling methods on objects, as well as arithmetic and logical computation, control flow, direct memory access, and exception handling. Then, the IL is delivered to the user in the form of DLL and/or EXE. Finally, Just-In-Time (JIT) compilers provided by Microsoft compile the IL into native machine code. Note that the JIT compilers only convert the IL as needed during execution and store the resulting native code for subsequent calls, which results in a fast code execution speed.

The .NET framework can be regarded theoretically as platform independent at the present time, since Microsoft only provides Windows-based CLR to convert IL into native platform codes despite the fact that IL codes are platform independent. Currently, the third-party Mono project has been designed to allow the .NET developers to easily implement .NET applications on Linux.

The .NET framework possesses a number of features that are well suited to meet the performance requirements of fault diagnosis and failure prognosis of a given system. Firstly, the object-oriented coding style in the framework allows the developer to build up a modular and flexible software system. Secondly, the framework possesses excellent interoperability. The developer can access not only the codes written in any of the .NET languages but also those executed outside the .NET environment. Any of the .NET languages, such as Visual Basic, Visual C++, Visual C# and Visual J#, can be utilized as the programming language, and many other popular languages such as JAVA and MATLAB also can be easily converted into the .NET languages. Thirdly, the framework provides a class library with a huge amount of functionality such as file reading and writing, graphic rendering and window forms, which aids the system developer to decrease massively the coding workload. Finally, simplified deployment can be achieved by either the exclusive tools in Visual Studio .NET or as simple as XCOPY operations.