Chapter 7
Validating the INTERPRETOR Software Architecture for the Interpretation of Large and Noisy Data Sets

Apkar Salatian
American University of Nigeria, Nigeria

ABSTRACT

In this chapter, the authors validate INTERPRETOR software architecture as a dataflow model of computation for filtering, abstracting, and interpreting large and noisy datasets with two detailed empirical studies from the authors' former research endeavours. Also discussed are five further recent and distinct systems that can be tailored or adapted to use the software architecture. The detailed case studies presented are from two disparate domains that include intensive care unit data and building sensor data. By performing pattern mining on five further systems in the way the authors have suggested herein, they argue that INTERPRETOR software architecture has been validated.

INTRODUCTION

In many domains there is a need to interpret high frequency noisy data. Interpretation of such data may typically involve pre-processing of the data to remove noise. Rather than reasoning on a point-to-point basis which is computationally expensive, this filtered data would be processed to derive abstractions which would be interpreted and the results reported. Such a common approach lends itself to the development of a software architecture.

Software architectures involve the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns. In general, a particular system is defined in terms of a collection of components and interactions among these components. Such a system

DOI: 10.4018/978-1-4666-2208-1.ch007

may in turn be used as a (composite) element in a larger system design. Software architectures can act as a model of computation for data flows in a system. Indeed, a good software architecture will involve reuse of established engineering knowledge (Shaw & Garlan, 1996).

In this paper we will describe and validate the *INTERPRETOR* software architecture for interpreting large and noisy data sets. INTEREPRETOR was inspired by the software architecture of ASSOCIATE (Salatian & Oriogun, 2011) for interpreting Intensive Care Unit monitor data and ABSTRACTOR (Salatian, 2010) for interpreting building sensor data - both systems have common features which facilitates a generic architecture. INTERPRETOR consists of 3 consecutive processes: *Filter* which takes the original data and removes noise; Abstraction, which derives abstractions from the filtered data; and Interpretation, which takes the abstractions and provides an interpretation of the original data.

THE INTERPRETOR SOFTWARE ARCHITECTURE

Figure 1 shows the Context Diagram of the INTERPRETOR system. The INTERPRETOR system takes high frequency noisy data and other relevant data to assist in interpretation from various input sources and presents to various output sources an interpretation of the original data.

Figure 2 shows the data flow in the INTERPRETOR system of Figure 1. Data is initially filtered to get rid of noise; rather than reasoning on a point to point basis, the resulting data stream is then converted by a second process into abstractions – this is a form of data compression. A third process to provide an assessment of the original data interprets these abstractions.

We, therefore, derive the overall software architecture of the INTERPRETOR System in form of a Structure Chart as shown in Figure 3.

It can be seen that INTEREPRETOR is a data flow architecture and model of computation. The architecture is decomposed into three processes, which can be changed or replaced independently of the others - this makes INTERPRETOR a loosely coupled system. Indeed, each process of the INTERPRETOR performs one task or achieves a single objective - this makes the INTERPRETOR a highly cohesive system. INTERPRETOR can also be considered a pipe and filter architectural style because it provides a structure for systems that process a stream of data.

We hope to extend our INTERPRETOR design architecture, such that we have a generic design pattern for voluminous and high frequency noisy data, whereby, the data is passed through three consecutive processes: Filter *Data* which takes the original data and removes outliers, inconsistencies or noise; Abstraction which takes the filtered data and abstracts features from the filtered data; and Interpretation which uses the abstractions and generates an interpretation of the original data.

APPLICATIONS OF THE INTERPRETOR SOFTWARE ARCHITECTURE

We will demonstrate the application of the INTERPRETOR software architecture to two case studies from the author's research endeavours: interpreting Intensive Care Unit (ICU) Monitor Data and interpreting building monitor data.

Figure 1. Context diagram of the INTERPRETOR system