DATA AND MODEL INTEGRATION IN FINANCIAL DECISION SUPPORT SYSTEMS

Soon-Young Huh[1], Keun-Woo Lee[1], and Hyung-Min Kim[2]

Graduate School of Management[1]
Korea Advanced Institute of Science and Technology
207-43 Chongyangni-dong, Dongdaemoon-gu, Seoul, Korea 130-012
PricewaterhouseCoopers[2]
23-6 Youido-dong, Youngdungpo-gu, Seoul, Korea 150-010

ABSTRACT

As the financial market grows, modern financial decision support systems need an efficient framework that enables us to mix and match the financial products and the various models dynamically. This paper proposes an object-oriented framework for separating the analytical models from the data objects representing the financial products and for integrating the models and data objects based on their connection needs. The proposed framework is also considered as efficient because of not being affected by the changes in data and models.

KEYWORDS

Model Management, Data and Model Integration, Financial Decision Support Systems

1. Introduction

Generally, decision support systems (DSS) separate organization's data from their models and manage them independently [5,8,10]. Thus, a data and model integration framework is required to execute the models filled with needed data. The framework cannot avoid being dependent on data schema and input/output structures of models since it queries upon data, inputs them to models, and deals with the results from model execution. Nevertheless,

most researches on data and model integration have not much considered changes on data schema or input/output structure of models since such changes hardly occur in traditional domains such as management science and operations research (MS/OR) models. However, in financial domains, financial products are being created constantly to meet the movements of financial markets and financial models are being improved to reflect the progress of financial knowledge [2,7]. If financial decision support systems (FDSS) are as inadaptable to such changes as traditional DSS, they should be continuously redeveloped resulting additional development cost and time.

In this paper, we propose a framework for overcoming such difficulties and supporting adaptable integration of financial products and models. The requirements of the framework are as follows. First, in terms of the interoperability, we introduce model interfaces to cope with the changes of model implementation. Through the model interfaces, FDSS links any pair of data and an associated model and executes it. Second, in terms of the adaptability, we develop a generic architecture of the model interfaces to cope with the changes of model interfaces. Then, FDSS can support various models including models with different model interfaces without any modification of the system.

In developing the framework, distributed object technologies [9] are adopted to effectively conceptualize the core constructs and mechanisms constituting the framework. The paper is organized as follows. In the section 2, we briefly review the related researches. Section 3 introduces our framework to facilitate the interoperability with the standard interfaces. Section 4 improves the framework for the adaptability with the port and the port dictionary. Finally, chapter 4 summarizes this paper and proposes future research plans.

2. Literatures on Data and Model Integration

Most researches related to data and model integration concentrate on the definition of model structure for effective integration with data [4,5] and the query processes of data needed by model execution [5,8]. There are two researches focusing on the relationship structure between data and models.

First, separating interface classes from concrete implementation classes, Zhang and Sternbach [11] proposed the data and model interface structure not affected by changes in the implementation classes. However, it should be modified as a whole when the interface classes are changed, such as increment of attributes. It is also recompiled whenever data or models are changed since the structure is based on the class design patterns [3] for reusability of classes in the source code level. Second, Rizzoli et al. [10] encapsulated data and models as domain classes and model classes, respectively, and proposed a structure that conveyed values and converted data format between the two kinds of classes. They intended to provide a more effective modeling environment with the structure. Their research has a meaning that they improved data reusability by structuring data classes formally and provided a more effective integration structure. However, their structure is just a conceptual and lacks definite description for implementing the processes. There is also a limit that the structure is dependent on changes of interfaces between domain classes and model classes.

In this paper, we accept the concept of separating interface from implementation and get the interoperability with the uniform interface. However, we design the integration structure independent of the interface differently from the previous researches. In doing so, we present concrete constructs constituting the structure that will be implemented using distributed object technology.

3. Uniform Interface Structure of Models

In defining and describing models, there are three approaches, defining as standard statements, as data, and as subroutines [1,4,5]. Especially, the subroutine approach can be applied to more various domains than the others since it uses general programming languages for defining models. To get such a wide applicability, we use the subroutine approach. Moreover, recent distributed object technologies [9] help this approach be more flexible to the changes of models, i.e., the whole system needs not be recompiled when any model within the system is either created or modified. However, for achieving such interoperability, generic model interfaces between data and models are required. We consider these interfaces as input/output attributes of models and separate them from actual implementation of the models. Then, FDSS does not concern the internal implementation of the models and users can link any combinations of data and a model if the models have the same interfaces.

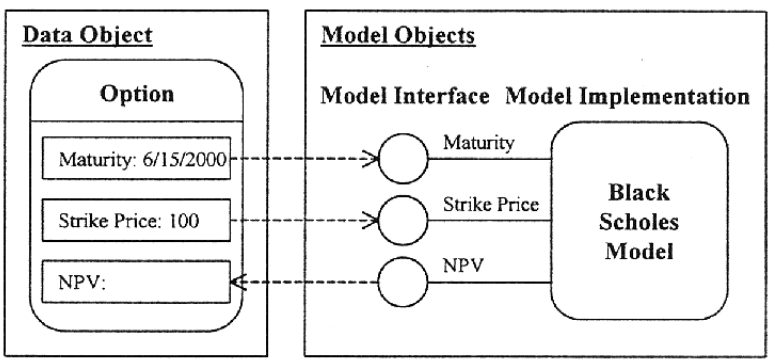

Figure 1. Model interface and model implementation

Figure 1 shows the model interfaces of a Black-Scholes option-pricing model [2,7] that is one of the most representative financial analysis models. In figure 1, option can only see the model interface that has maturity, strike price, and NPV. If another modified model is added into the FDSS and has an identical model interface with the model such as figure 1, it can be linked with the option data without any modification of the FDSS.

Although we can get the interoperability of models through uniform interfaces, we cannot guarantee the interoperability when model interfaces are changed. In the next section, we address this issue.

4. Adaptable Data and Model Integration Framework

In this section, we introduce a generic architecture of the model interfaces to cope with the changes of model interfaces.

Note

- **Port**: a data structure flexibly representing various attributes of data and model interfaces, for example, strike price, maturity, and so on.
- **Inport**: a port representing input attributes of model interfaces.
- **Outport**: a port representing output attributes of model interfaces.

A port has properties including **port_name**, **port_type**, **port_unit**, and **port_value**. Using the ports, data can manage their static information as a set of ports and model interfaces can be represented with a set of inports and outports. Our idea for integration is matching data ports and model ports, setting values from data ports into inports of models or getting the results from the outports. The system can match ports that have the same names. However, some ports can have different names even though their meanings are identical. To store the information of such ports, we introduce the **port dictionary**.

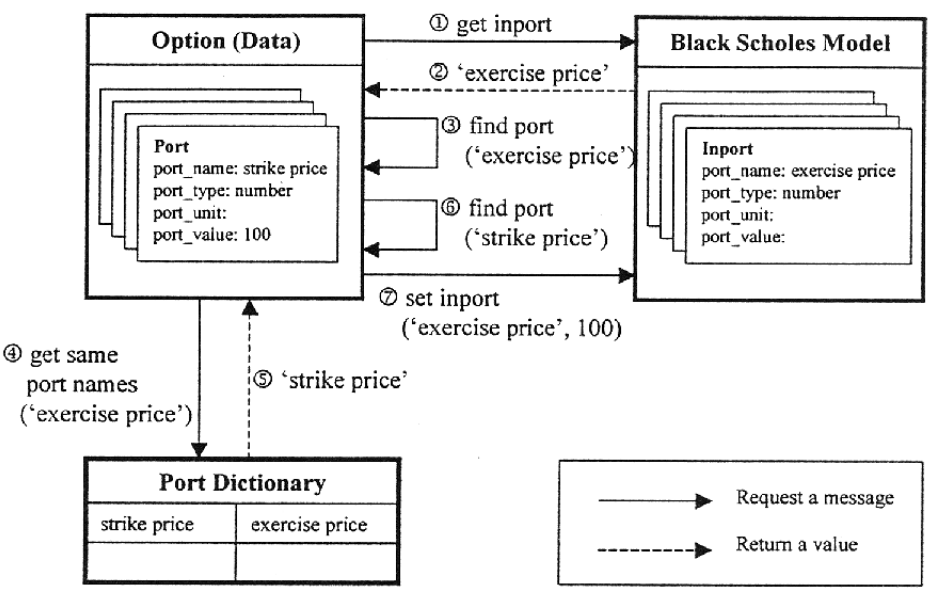

Figure 2. Port matching process for ports with different names

Figure 2 shows a procedure for the integration of an option and a model. The procedure matches the 'strike price' port of the option to the 'exercise price' port of the Black Scholes [2,7] model to calculate the theoretical price of the option. First, the option requests the inport name (①) of the model to feed the needed information and gets the 'exercise price' (②) as the result. It then tries to find (③) the same port in its port set. Because there is no port of the same name, the option queries to the port dictionary (④) about ports that have the same meaning with 'exercise price' and gets 'strike price' (⑤). It then matches the 'exercise price' port of the model with the 'strike price' port of

the option (⑥) and can transfer the value (⑦) to the model. By following the port matching process described above, we can match any pairs of ports that have the same meaning in any cases whether their names are identical or not.

5. Conclusions

For DSS's whose domain is changing frequently such as FDSS, the data and model integration framework requires the abilities connecting them freely and reflecting the changes seamlessly. In this paper, we propose a framework to facilitate the interoperability and the adaptability. We define the generic model interface for the interoperability. As all models have the model interfaces, they can be used on any data sets with no concern of the internal implementation. We also introduce the port and the port dictionary for the adaptability. With the port and the port dictionary, changes of model interfaces become just a matter of the port set and do not affect the interface structure itself.

With regard to this study, we are working for the method to construct composite models by linking some models and the mechanism to integrate the composite models with the data under the integration framework proposed in this paper.

REFERENCES

1. Chang,A.M., Holsapple,H.C., and Whinston,A.B. (1993). Model management issues and directions. Decision Support Systems, 9, 19-37
2. Copeland,T.E. and Weston,J.F. (1992). Financial theory and corporate policy, 3rd ed.. Addison-Wesley, Reading, MA
3. Gamma,E., Helm,R., Johnson,R., and Vlissides,J. (1995). Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA
4. Geoffrion,A.M. (1989). The formal aspects of structured modeling. Operations Research, 37(1), 30-51
5. Huh,S.Y. (1993). Modelbase construction with object-oriented constructs. Decision Science, 24(2), 409-434
6. Huh,S.Y. and Chung,Q.B. (1995). A model management framework for heterogeneous algebraic models: object-oriented database management systems approach. Omega, Int. J. Mgmt Sci., 23(3), 235-256
7. Hull,J.C. (1997). Options, futures, and other derivatives, 3rd ed.. Prentice Hall, NJ
8. Liang,T.P. (1985). Integrating model management with data management in decision support systems. Decision Support Systems, 1, 221-232
9. Pritchard,J. (1999). COM and CORBA® side by side: architectures, strategies, and implementations. Addison-Wesley, Reading, MA

10. Rizzoli,A.E., Davis,J.R., and Abel,D.J. (1998). Model and data integration and re-use in environmental decision support systems. Decision Support Systems, 24(2), 127-144

11. Zhang,J.Q. and Stembach,E.J. (1996). Financial software design patterns. Journal of Object-Oriented Programming, Feb., 6-12