The Research about Software Integration oriented Heterogeneous Architecture Style

Chong-chong Zhao, Li-yong Zhao
School of Information Engineering
University of Science and Technology Beijing
Beijing, China
chongchongzhao@gmail.com

Abstract—**Software architecture style is important for software reuse with the buildup of software complexity. Existing popular software architecture styles are numerous, but they can not satisfy the requirements in software integration. To solve the issues in traditional integration schemes, this paper presents a software integration oriented heterogeneous architecture style. This heterogeneous architecture style adopts three-layer style on the whole, pipeline/filter style in the data layer, blackboard style in the business layer and MVC style in the presentation layer. Additionally, a soft-bus is designed to solve the communication mismatching between subsystems with heterogeneous styles.**

Keywords- software integration; software architecture style; heterogeneous architecture style; soft-bus

I. Introduction

Software Architecture is one of the important research fields of software engineering now. The complexity of systemic and software reuse technology are the inner driving force of the development of software architecture. What the research on software architecture focuses on is not the algorithm design and the data structure design of computation. Commonly, software architecture is a design that is more abstract and bigger in granularity than code segment, algorithm or data structure. But it does not involve detailed algorithm and data structure. Software architecture style is a common used model to describe the system organization in a specific application domain. It reflects the common features of the structure and the semantic of the system in the domain, and instructs how to effectively organize each model and subsystem as an integrated system.

Existing popular software architecture styles are numerous, but they can not satisfy the practical requirements well in software integration. Especially in specific application domains, integrating a large amount of legacy systems is necessary in order to reuse existing resources and achievements. If the method is inappropriate, the integration may affect the development speed as well as the stability and reliability. Traditional integration schemes include programming-interface-based and distributed-object-based technologies. Both of them are point to point. There are problems though it can keep the autonomy and independence of integrated subsystems. First, when the number of subsystems is big, the work will become heavy. Second, it is tough to integrate a new subsystem into the existing system. Third, it is hard to modify the existing subsystems according to the new business requirements in the domain.

To solve these issues, this paper presents an integration oriented heterogeneous architecture style. This heterogeneous architecture style adopts three-layer style on the whole, pipeline/filter style in the data layer, blackboard style in the business layer and MVC style in the presentation layer. In addition, a soft-bus is designed to solve the communication mismatch between subsystems with heterogeneous styles.

The rest of the paper is organized as follows. The related research is reviewed in section II. The integration oriented heterogeneous architecture style is presented in section III. The design of the soft-bus communication platform is introduced in section IV. Section V verifies the architecture style by a system in the petroleum domain. And finally the conclusion is included.

II. Related Research

The conception of software architecture has been proposed years ago. Software Crisis makes people to pay more attention to the research of software engineering. With the growth of the scale and complexity, the interrelations of various software elements in the system affect the overall quality of the software. For the large-scale complex software system, the top-level structure design is more important than the calculation algorithm design. People also realize the importance of software architecture and agree that the further studies on software architecture will be the most hopeful way to improve the productivity and maintainability of the software which is described in [1],[2],[3],[4] and [5].

Till now, there is no unanimous agreement on the definition of the software architecture. Dewayne Perry[3], Mary Shaw[4], David Garlan[5], Barry Boehm[6], Kurt Wallnau[7] and others give the definition from different perspective. Analyzing these definitions, the main features of the software architecture are as follows.

Firstly, what the research on software architecture focuses on is larger and more abstract than algorithms and data structure design. Commonly software architecture does not involve specific algorithm and data structure. Secondly, the content of software architecture is the structure and organizational relationships between the abstract functional elements. Once the software architecture is determined, the profile of the software is definite. That is to say the catalog, quantity and relationships of the elements are definite. But its content does not involve concrete implementation details, and even the definition of each functional element is abstract, they need for specific design during the realization. Finally, the results of software architecture design include the functional elements of the software and the relationships between these elements. Before the software has been coded, the software element is only on the abstract conception level, and function is the unique character of each software element. These software elements with functional characteristics and the relationships can determine the abstract architecture of the software.

The software architecture style is the conventional model to describe the system organization way of one specific domain.[1][2][8] An architecture style is defined by the component sets and the interactive rules between them. Every style has its own way of the component division, the design rule of communication protocol, thus define the topology structure between components. The architecture styles differ in three points. Firstly they have different functional definition about components. Even designing a same application, the styles more or less differ in the layout and granularity of the overall function decomposition, because of the different perspectives and technologies being used. Secondly the interactive rules between components (i.e. the communication protocols) are different. Differences in the data exchange method, synchronization type or the format of exchanged data, will lead to communication protocol mismatch between components in different architecture styles. At last, the topology structures between components are different. They will be affected by the differences of the software function definitions. Because of certain function relationships between components, there are many interactions and combinations between components. So when the component functions are defined, the integration order and relationship between components have also been set implicitly.

Every architecture style is abstracted from the cases that successfully solve the same problems. Different architecture styles are applicable to different applications. Till now, there have summarized several architecture styles, such as batch processing sequence, pipeline/filter, main program/subprogram, object oriented, hierarchical structure, process communication, event system, interpreter, rule-based system, database system, hypertext system, blackboard system and so on.[1][2][9]

But these software architecture styles have limitations in software integration applications. During the domain oriented software development, it is a common requirement to take full advantage of existing resources and achievements, which means integrating lots of legacy systems. If the integration method is inappropriate, it may affect the development speed as well as the stability and reliability. Traditional integration schemes include programming-interface-based and distributed-object-based (e.g. DCOM, CORBA, EJB, etc.) technologies. Both of them are based on point to point. The point to point integration can keep the autonomy and independent of every integrated subsystem, but there are problems as follows. 1). When the number of the integrated subsystems is big, the relationships between subsystems form a net topology [10]. That means a subsystem needs to get the universal interface set of other components in order to communicate. This increases the difficulty of the subsystem's development and re-development. And the potential defects in the communication between subsystems are also increased. 2). When integrating a new business subsystem, it may need to adjust or change the interfaces of lots of existing subsystems. It is tough and may result in unreliability. 3).In practice, with the business evolution in a domain, we need to do adaptable revision to the original subsystem. This requires that the combination of the subsystems can be adjusted easily according to business needs. However it is difficult depending on point to point integration.

III. Integration Oriented Heterogeneous Architecture Sytle

Every architecture style has advantages as well as disadvantages. One architecture style can guarantee an optimal solution to a simple question. When dealing with problems as complicated as software integration, one style can be optimal in the local but not sure in the whole. Aiming at above issues, this paper presents an integration oriented heterogeneous architecture style. As shown in Fig.1, this style integrates multiple heterogeneous styles. It adopts three-layer style as a whole which adopts pipeline/filter style, blackboard style and model-view-control style locally. The object is to realize "global flexible, local optimum".

A. *Global Style Design*

Although three-layer style has shortages such as inflexibility and weak pertinence, choosing it as the overall style has advantages as follows.

- Three-layer style is a special hierarchy style. Hierarchy style decomposes a complex problem into a series of simple problems. It is beneficial to decrease the risk of failure due to losing control of the system complexity.

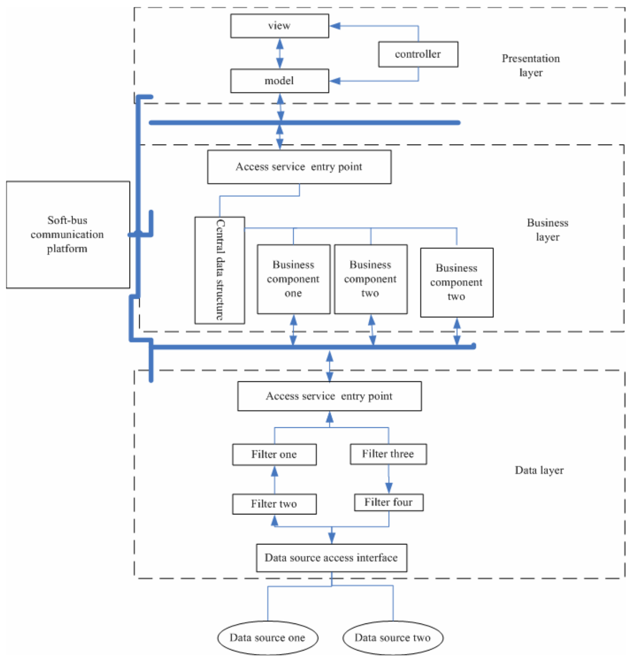

Figure 1. Integration oriented heterogeneous architecture style

- As long as keeping the interfaces unchangeable, different methods can be used to implement each layer and adaptable revisions can be done inside the layer. This can improve the flexibility and scalability of systems.
- Each layer is relatively independent. So it is easy to do specific design according to the characteristics of the problems to be solved in different layers.

In consideration of generality, the integration oriented heterogeneous architecture style has three layers including data layer, business layer and presentation layer from bottom to top.

B. *Style Design of Data Layer*

The data layer provides data service for the business function, and satisfies the requirements not only the integrality and consistency but also the flexibility and modifiability. Taking above requirements into account, the data layer adopts pipeline/filter style and its advantages are as follows.

- The component can be reused easily. Two filters can be connected only if the interface is match between the in-pipe and out-pipe. So it is easy to accomplish a certain data processing that just the filters combination.
- It is easy to modify. The filters are independent and new filters can be added according to the needs of update, upgrade and extension.
- The system has native concurrent capability. In a pipeline, each filter can run independently and serially. Therefore the filters can concurrently work with other tasks.

C. *Style Design of Business Layer*

The business layer includes a mass of domain business components. One the one hand, these components are based on the service of the data layer. On the other hand, a business process flow call business components by certain sequence and the business components should meet the requirements of flexibility. Therefore, the business layer faces the challenge that the loose coupling components share data. Take this into account, the business layer adopts blackboard style and the advantages are as follows:

- The business logic is easy to extend. Blackboard style can add new business components in the system conveniently.
- It can satisfy the needs of adaptability revision. The black-box reference between components in this style is based on interfaces. So if the interface does not change, the new component can replace the old one conveniently.
- It supports reuse. The relationship between components depends on the central data structure. The components can be combined flexibly for complex business processes.

D. *Style Design of Presentation Layer*

The presentation layer can receive the user input and show the processing result of the business layer. The popularization of GUI promotes visualization being the basic demand of current software interaction. The separation of presentation and business has become a trend of architecture design. The MVC style is adopted in presentation layer and the advantages are as follows.

- The interactive interface has good revisability. In MVC style, the interface and process are separated, and thus avoid the complete recode because of a small interface modification.
- This style can create and use multi-views for a data model at runtime. The change-propagation mechanism can insure that the related views receive the data changes. So, all the related views and controllers can be synchronized for the data consistency.
- Views and controllers can be plug-in that they can be easily changed, dynamically opened/closed and substituted at runtime.

IV. DESIGN OF SOFT-BUS COMMUNICATION PLATFORM

The styles integrated in this heterogeneous architecture style differ in interactive rules (i.e. the communication protocols). The main differences are reflected in data format and message synchronization [11]. It can be called communication mismatch. For example, there is communication mismatch between pipeline/filter style and blackboard style.

In the pipeline/filter style, the components receive input messages and produce output data stream by inner processing. The upstream filter needs to insure sending data to the downstream filter correctly. And the downstream filter needs to insure receiving data from the upstream filter completely. So the synchronous invoking mode is reasonably adopted.

In the blackboard style, there are two kinds of components. The central data component shows the current state and the independent business component executes processes on the central data component. The business components communicate with each other indirectly by the central data component. So the asynchronous message exchange mechanism is reasonably adopted.

As mentioned above, a proper design is needed to address the mismatch in the heterogeneous architecture style. And this paper chooses soft-bus. Soft-bus learns from hardware and is a set of virtual data transfer lines [12]. Applying soft-bus for heterogeneous style to solve the communication mismatch has the following advantages.

- It can simplify the communication topology between components and reduce the complexity of the communication interface. In soft-bus, the topology is star type. That means all the integrated subsystems communicate with the soft-bus directly. The soft-bus will transmit the messages sending by a subsystem to the corresponding destination subsystem. Compared with peer to peer communication, it can greatly reduce the complexity.

- The asynchronous and synchronous communication mechanisms can coexist. The message packets with unified format are transferred on the soft-bus. After the packet has been received, the message type will be checked in order to determine whether to send feedback to the sender.
- It can improve the openness and scalability of the system. Based on soft-bus, a developing component needn't know about the details of other components. The rule that a component should obey is only to provide communication interface and realize receiving/dispatching mechanism according with the regulations. Any component according with this rule can be integrated into the system readily.
- The centralized control strategy can control the communication topology, quality flexibly and conveniently. For the soft-bus topology, all messages should go through the middle node. As long as adding an independent quality control module and routing module in the bus, the centralized control is working.

A. *Soft-bus Communication Platform Structure*

The soft-bus communication platform is divided into the conversion layer and control layer as shown in Fig. 2.

The conversion layer consists of converters which are designed for each subsystem. The converter is a translator between the soft-bus communication protocols and subsystem inner communication protocols. The converter is responsible for the transformations between the bus messages and the subsystem inside information including data and control commands. The control layer is responsible for routing management and other massage management strategies.

The typical message transfer is that the converter transforms the subsystem inner message into a new one understood by the controller. Then the controller determines the routing path and dispatches it

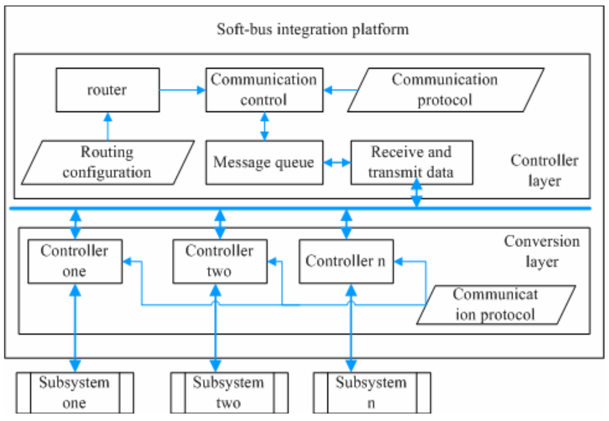

Figure 2. The structure of soft-bus communication platform

B. *Communication Protocol*

The communication protocol of the soft-bus communication platform is a contract that should be complied by all the components connected to the bus. The contract includes the range and semantic specifications of routing information, control information and packet data. The routing information includes the source subsystem and the destination subsystem. The control information includes the synchronization mode, message type and the subsequent data type. The packet data includes the data length and the data content.

V. EXPERIMENTAL RESULTS

Drilling is the key step in oil exploration and production. The drilling simulation integrated system is important and we can use it to make better decisions, to improve drilling speed and to decrease cost. The research of drilling simulation integrated system faced the challenges as follows.

First, there is a large number of legacy software to be integrated. Drilling is a huge systematic project and lots of simulation software has been partially used. The drilling simulation integrated system should not be re-development completely. And integration as far as possible is preferred. Second, the data processing is complex. Existing simulation software to be integrated usually uses different units for data processing. To integrate them in one system should consider unit conversation. Third, the data presentation requires flexibility. Drilling simulation consumes and products large amounts of data. Users usually like to view not the origin data directly but the display as form, curve, cylinder and three-dimension. And the representation modules should be reusable.

To solve these issues, the drilling simulation integrated system applied software integration heterogeneous system style.

The data layer structure is shown in Fig. 3. Considering the conversion between the metric, British and mixed units, the pipelines include the forward ones from data source to business layer and the backward ones from business layer to data source.

The business layer structure involves ten subsystems integrated from the legacy software, including track design, cost evaluation, track control, casting design, bottom hole assembly design, casing string design, cementing simulation, well control and killing well, drill and drilling parameter design , drill and hydraulic simulation.

The presentation layer structure involves some common views such as two-dimension, three-dimension, grid, curve, pie and column views.

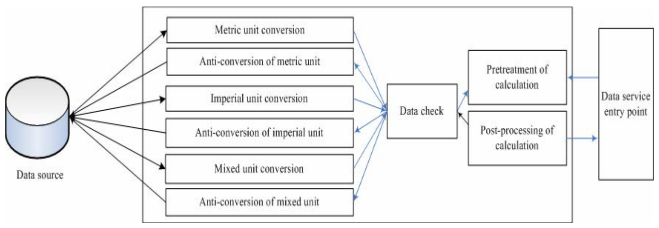
Metric unit conversion
Anti-conversion of metric unit
Imperial unit conversion
Anti-conversion of imperial unit
Mixed unit conversion
Anti-conversion of mixed unit
Data source
Data check
Pretreatment of calculation
Post-processing of calculation
Data service entry point

Figure 3. Data layer structure of drilling simulation integrated system

The main content of soft-bus communication protocol using in drilling simulation integration system is shown in TABLE I.

The drilling simulation integrated systems has been developed. The practical results show that the system can be able to integrate legacy software quickly and efficiently. And the desired scalability is also achieved.

VI. CONCLUSION

The software architecture style enhances the software reuse, which is closely associated with the domain application. To address the issues in domain software integration, this paper presents a heterogeneous system style. This style uses multiple styles and takes advantage of different styles, such as the flexibility and versatility. In addition, to improve the communication mismatch between heterogeneous styles, a soft-bus communication platform is adopted. A system has been developed based on the style and was running in an enterprise of petroleum domain. The result shows the advantages as expected.

TABLE I. THE INTRODUCATION OF COMMUNICATION PROTOCOL

Name	Type	Length	Description
Target adapter encoding	int	4B	Each adapter has a positive integer ID. Special coding includes: 0: The control information from subsystems to soft-bus, not transmit; -1:Broadcast message to all subsystems.
Source adapter encoding	int	4B	Each adapter for use of a positive integer ID
Synchronous /Asynchronous	char	1B	0- Synchronous; 1- Asynchronous.
Message type	char	1B	0-Management message; 1-Control Command; 2- Packet.
Data type	int	4B	Depending on the specific content of the communication subsystem
Date length	int	4B	
Packet	char	4B	

REFERENCES

[1] Perry D.E., Wolf A.L., “Foundations for the study of software architecture”, ACM SIGSOFT Software Engineering Notes, Vol.17, Issue: 4, Page 44-52, 1992.

[2] Metayer D.L., “Describing software architecture styles using graph grammars”, IEEE Transactions on Software Engineering, Vol.24, Issue: 7, Page 521-533, 1998.

[3] Perry D.E., Porter Adam, Wade Michae l W, Votta Lawrence G, Perpich James, “Reducing Inspection Interval in Large-Scale Software Development”, IEEE Transactions on Software Engineering, Vol.28, Issue: 7, Page 695-705, 2002.

[4] Shaw M, “Procedure calls are the assembly language of software interconnection: Connectors deserve first-class tatus”, Studies of Software Design. ICSE '93 Workshop, Page 17-32, 1996.

[5] Garlan D, Perry D, “Introduction to the Special Issue on Software Architecture”, IEEE Transactions on Software Engineering, Vol.21, Issue: 4, Page 269-274, 1995.

[6] Bhuta J., Boehm B., Meyers S., “Process Elements: Components of Software Process Architectures”, Unifying the Software Process Spectrum, International Software Process Workshop, SPW 2005, Page 332-346, 2006.

[7] Wallnau, K, “Methods of Component-Based Software Engineering: Essential Concepts and Classroom Experience”, In: Proc. of the 23rd International Conference on Software Engineering, Page 709-710, 2001.

[8] Chen Hai-Shan, “Survey on the style and description of software architecture”, In: Proc. of 8th International Conference on Computer Supported Cooperative Work in Design, Vol.1 Page 698-700, 2004.

[9] Shaw M., “Making choices: A comparison of styles for software architecture”, IEEE Software, Vol.12, Issue: 6, Page 27-41, 1995.

[10] Keshav R, Gamble R, “Towards a taxonomy of architecture integration strategies”, In: Proc. of International Software Architecture Workshop, Page 89-92, 1998.

[11] Shaw M, “Architectural issues in software reuse: It's not just the functionality; it's the packaging”, In: Proc. of the ACM SIGSOFT Symp. on Software Reusability. New York: ACM Press, Page 3-6, 1995.

[12] ZHANG Shi-Kun, WANG Li-Fu, Yang Fu-Qing, “Software Architecture Style based-on Hierarchy Message Bus”, Science in China(E), Page 393-400, 2002.