
Dongjin Yu, Suhang Zheng

Abstract

Nowadays, managers in organizations usually have to confront increasingly complicated decision environments and dynamically changed decision demands. Traditional Decision Support Systems (DSSs) sometimes neglect that both requirements and objectives of decisions are indeterminate, and thus do not support flexible decision making. In order to meet dynamically changed contexts, DSS generators are introduced to build specific DSSs. However, DSS generators cost greatly but only bring about limited extensions. This paper reconstructs traditional DSS architecture using Service Oriented approaches. It describes the hierarchical model, conceptual model and implementation model for the framework of Service-based DSS, or SBDSS. In addition, it introduces the layered DSS adaptability model and presents the approach to high adaptabilities with context sharing and the assembly model of SCA and Spring Framework. Finally a prototype based on SBDSS is also given to show its feasibility.

Keywords: Decision Support Systems, Adaptive Systems, Service Oriented Computing, Framework

1. Introduction

Due to the fierce market competition, the scale of organizations keeps expanding while the business complexity grows. Nowadays, the decision making faces at least three challenges. 1) The increasingly complicated decision environment. Decisions should be made across the boundaries of systems, departments and even enterprises. 2) The dynamically changed decision demands. Decisions should be made more quickly and adapt to the dynamic changing business requirements with no higher cost. 3) The more and more separate decision resources. The distributed and heterogeneous data sets and legacy systems make integration of data more difficult.

Decision Support Systems (DSSs) are interactive software-based systems intended to support business and organizational decision-making activities. To cope with above challenges, DSS should be more flexible and adaptive. The architectural pattern or style for DSS plays an important role for these essential features.

As information technology continues to evolve, one can identify various architectural patterns that seem to reoccur in the development of DSSs. Sprague proposed so-called D/IDM (Dialogue/Interface, Data, and Models) architectural pattern, which has been widely adopted up to now [1]. Besides, Robert Bonczek, Clyde Holsapple, and Andrew Whinston presented a theoretical framework, named BHW model, for understanding the issues associated with designing knowledge-oriented Decision Support Systems [2]. BHW model consists of four essential parts: language subsystem, presentation subsystem, knowledge subsystem, and problem-processing system. Holsapple et al., however, showed later that the BHW model can accommodate, as a particular case, the largely utilized D/IDM pattern of Sprague [3].

One other architectural view is the distinction of Specific DSS, DSS Generator, and DSS Tool. Sprague and Carlson argue that there are three levels of hardware and software that are characterized as DSS [4]. A Specific DSS is the actual system that a manager works with during the decision process. A DSS Generator is a software package that provides capabilities for building specific DSSs rapidly and easily. DSS Tools are hardware and software elements which facilitate the development of a specific DSS or a DSS generator.

However, no matter how it is composed, traditional decision support systems, with tight components coupling and non-standard interfaces, often neglect that both requirements and targets of decision are sometimes indeterminate. They are not capable of flexible decision due to the lack of adaptability. Historically, the DSS generator needs to be introduced while extending the specific DSS,
which costs greatly but only brings about limited extensions. Fortunately, the appearance and development of Service Oriented Computing (SOC) technology in 1990s and the following decade could settle the problems of rigidity and closeness of traditional DSS to some extent [5][6]. This paper reconstructs traditional DSS architecture using SOC, and moreover proposes the framework of Service-based Decision Support System, i.e. SBDSS. In SBDSS, distributed and heterogeneous business components are encapsulated and provided in the form of Services. With the loosely components coupling, SBDSS could dynamically select, adapt, compose and invoke the inner Service components, and therefore be easily integrated into the distributed enterprise computing environment through the way of information integration and application coupling with reduced cost.

The rest of the paper is organized in the following manner. Section 2 introduces the framework of SBDSS in detail, including its hierarchical model, conceptual model and implementation model. The layered adaptability model and the approach to high flexibilities by context sharing are illustrated in section 3. After the implementation of SBDSS discussed in Section 4, Section 5 offers related works. Finally, the last section provides concluding remarks and future research directions.

2. Framework of Service Based DSS

Remodeling the functional entities of DSS by means of Services is an alternative way to promote flexibilities of traditional DSS. Here, Services comprise unassociated, loosely coupled units of functionality that have no calls to each other embedded in them. Each Service implements one action, such as extracting data from data warehouse, access to and manipulation of a statistical, financial, optimization, or simulation model.

Migrating to Service Oriented Architecture or constructing Service-based DSS (SBDSS) in other words, is a natural evolution of traditional DSS. To illuminate the framework of SBDSS, three models are necessary: the hierarchical model, the conceptual model and the implementation model (Fig. 1). Both conceptual and implementation models take the form of hierarchical structure. The former focuses on structural and functional parts of each layer, while the latter gives means of their realizations. The existent relationships, i.e. generalization and refinement in the view of UML, provide convenience to trace among above three models.

2.1. Conceptual Model of SBDSS

The conceptual model of SBDSS views SBDSS from 5 separated layers. From the side of decision makers toward the side of physical facilities, they are named as the interactive layer, the Service combination layer, the Service layer, the business component layer and the resource layer, respectively. Each layer communicates only with its neighboring layers. Therefore, the modification of one layer, which only affects its upper layer, is localized.

The interactive layer presents Web interfaces between decision-makers and DSS, through which decision-makers prompt the DSS and the DSS returns its response. The Service combination layer assembles different atomic Services from Service layer in sequential, iterative, parallel or selective order, and provides required manipulations again as Services again. The layer of Service is then followed by the components layer and the resource layer. The former deals with the physical implementation of Services, while the latter provides consistent interface to different resources, such as database, data warehouse, model base, knowledge base and case base.

![Figure 1. Models of SBDSS Framework](image)
2.2. Implementation Model of SBDSS

As for the implementation model, SBDSS adopts SCA (Service Component Architecture) [7] for service modeling, the SDO (Service Data Object) for data and message modeling and the BPEL (Business Process Execution Language) for service assembly respectively (Fig. 2). SBDSS finally integrates the different units with the help of JBI (Java Business Integration) based Enterprise Service Bus.

SBDSS is configured with the SCA-compliant process service engine above the JBI-based Enterprise Service Bus to fulfill Service coupling. All shareable function units should be reconstructed as the standard Service components. The interaction therefore could be realized between the coherent interfaces of Services and references. Besides, the component implemented in BPEL is configured as the portal, which orchestrates the individual Service components to achieve required values.

3. Adaptabilities of DSSs

Adaptive capacity is the capacity of a system to adapt if the environment where the system exists is changing. A system with high adaptive capacity exerts complex adaptive behavior in a changing environment. As applied to Decision Support Systems, the adaptive capacity is determined by creative flexibility in decision making and problem solving. Since the decision contexts, such as decision objectives, decision constraints and decision-makers’ preferences, may be rapidly changing and are not easily specified in advance, adaptive capacity is usually regarded as one of the most important features of the effective and efficient DSSs.

3.1. Layered DSS Adaptability Model

In SBDSS, the adaptability of DSSs is modeled as four consecutive levels: Adaptability to self-Adjustment (AoA), Adaptability to Refactoring (AoR), Adaptability to Service Reconstruction (AoS), and Adaptability to Integration (AoI) as Fig. 3 shows. Adaptability to self-Adjustment represents the flexibility of dynamically changing of DSS itself, while the latter three together represent the capacities of DSS of being changed or evolved by its designers.

**Definition 1 - Adaptability of self-Adjustment (AoA):** the capacity to adjust DSS itself by decision-makers along with contextual changing. DSSs with AoA provide customizable human-computer interaction, such as individual oriented portals and configurable decision support processes.

![Figure 2. Implementation Model of SBDSS](image-url)
Definition 2 – Adaptability of Refactoring (AoR): the capacity to rebuild DSSs while keeping frameworks unchanged. DSSs with AoR provide skeletons and common behaviors of general DSSs, from which domain specific DSSs could be easily extended and evolved.

Definition 3 – Adaptability of Service reconstruction (AoS): the capacity to reconstruct serviceable components by assembling related classes. DSSs with AoS provide flexible means to change implementations of serviceable components.

Definition 4 – Adaptability of Integration (AoI): the capacity to couple components with those outside DSSs. DSSs with AoI provide the interoperability with external systems. AoI could be achieved by migration of legacy assets of DSSs to Service-Oriented Architecture.

Table 1 compares different features of AoA, AoR, AoS and AoI.

Table 1. Features of Four Levels of Adaptability

<table>
<thead>
<tr>
<th></th>
<th>Scope of Adaptability</th>
<th>Extent of Adaptability</th>
<th>Bearer of Adaptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>AoA</td>
<td>[N/A]</td>
<td>Low</td>
<td>End Users</td>
</tr>
<tr>
<td>AoR</td>
<td>Component Level</td>
<td>Medium</td>
<td>Developers</td>
</tr>
<tr>
<td>AoS</td>
<td>Service Level</td>
<td>Medium</td>
<td>Configurer</td>
</tr>
<tr>
<td>AoI</td>
<td>Inter-service Level</td>
<td>High</td>
<td>Deployer</td>
</tr>
</tbody>
</table>

3.2. Context Sharing

The granularity of Services is always one of key concerns when designing SBDSS. Small-sized Services increase the management overhead, while large-sized Services reduce the reusability. As the concept of Service usually corresponds to the business value, it is better not to keep the granularity of Service too small. However, Services of bigger size are usually less flexible. Although the implementation of composite Services could be changed by reassembling related Services, atomic Services have no way. Adaptability of Service Reconstruction (AoS) aims to flexibly change the implementation inside the Service itself, where Adaptability of Refactoring (AoR) focuses on capacities of component refactoring. Both AoS and AoR do not need to take the heterogeneity into consideration. However, to increase the adaptability of DSS, it’s better for AoS and AoR to be able to reuse outside Services from inside Service. In other words, the context inside Services, i.e., the context
of AoS and AoR, should be accessible from within the context outside Services, i.e., the context of AoI in controllable manners and vice versa.

As Fig. 4 illustrates, the contexts of SBDSS are separated as 3 levels, i.e. Domain Context, Composite Context and Component Context. Along with 3 context levels, 4 accessing rules are giving as following.

1) Rule of Access from Direct Outer to Inner. The services, references and attributes defined in one context could be accessible only by the neighboring outer context.

2) Rule of Access from Inner to All Outers. The inner context could access services, references and attributes defined in all outer contexts which contain this inner context.

3) Rule of Same-Name-Overwrite. The Services, references and attributes defined in the inner context could overwrite those with same names in all outer contexts which contain this inner context.

4) Rule of Direct Promotion. The services, references and attributes defined in the inner context could be promoted as those in the neighboring outer context.

4. Implementations and its Evaluation

A prototype of Service-based Labor Market Decision Support System (SBLMDSS) has been developed to verify the feasibility of SBDSS. As a domain specific DSS, SBLMDSS provides general data mining algorithms of classification, clustering and association. It helps conduct feature analysis, structural analysis, transferring analysis and trend analysis of demand and supply of regional labour resources.

SBLMDSS employs Apache Tuscany SCA (tuscany.apache.org/) to achieve Adaptability of Service reconstruction (AoS) in composite context. In SBLMDSS, computations of mining algorithms and manipulations of data are encapsulated in SCA-compliant components. In addition, SBLMDSS employs Spring IoC to decouple the dependency among beans inside Service implementation to achieve Adaptability of Service Construction (AoS) and Adaptability of Refactoring (AoR). The SCA Spring Component Implementation specification describes two ways in which a component could use Spring for an implementation [8]. Instead of using a complete Spring application context to implement a composite component within an SCA assembly, SBLMDSS uses explicit declaration of SCA related beans inside the Spring configuration to achieve controllable access.

The following code segment presents how to configure the classification service in Spring application context in SBLMDSS based on J48, the open source Java implementation of the C4.5 algorithm in the Weka data mining tool (www.cs.waikato.ac.nz/ml/weka/).

```xml
<beans
   xmlns=http://www.springframework.org/Schema/beans
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xmlns:sca="http://www.springframework.org/schema/sca"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/sca"
   ...
</beans>
```

Figure 4. Contexts of SBDSS
Fig. 5 depicts the assembly model which above code segments suggests. As it shows, the implementation of Bean of J48 depends on Bean of Filter, Bean of Model Selector and Bean of Classifier, which in turn depends on Bean of Splitter and Bean of Pruner. Bean of filter has a reference to the service of data preprocess which is implemented outside the component. The actual beans of Model Selector, splitter and pruner could be injected at runtime using Spring IoC. For the case showed in Fig. 5, they are C45ModelSelector, C45Split and C45PruneableClassifierTree. In addition, the confidence factor could be set via the SCA property named as confidenceFactor.
SBLMDSS has collected massive records of employment from regional labor markets for more than 10 years. Based on organization scales, lines of industry, and posts wanted, SBLMDSS could suggest reasonable salary offers for a given organization to recruit employees of different ages, genders, and education backgrounds. SBLMDSS reuses some graphical components (Fig. 6) and underlying mining logics from Weka, and reassembles components based on SBDSS. Tests were conducted to evaluate the performance of SBLMDSS on servers with Intel Core i3-540 (3.06GHz) and 4GB memory. To classify 46682 records with 5 attributes in a 10-time test, SBLMDSS took min. 26 seconds and max. 40 seconds to give the final classification model. Meanwhile, SBLMDSS gave the model in min. 30 and max. 42 seconds. Compared with Weka, SBLMDSS showed a trifle of lower performance due to its loose-coupling of components. However, it was still acceptable.

5. Related Works

According to Keen, the concept of decision support has evolved from two main areas of research: the theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early 1960s, and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s [9]. Inadequate attention has been given to defining these patterns or styles for DSS.

Two most traditional architectures or patterns of DSS were I/IDM [1] and BHW [2]. Many new systems developed later seemed to imitate or adopt these two patterns by incorporating updated hardware and networking technologies. This eventually leads to the generation of many other patterns later, such as Sprague-Carlson’s four patterns of DSS Network, DSS Bridge, DSS Sandwich, and DSS Tower [4], and Power-Kaparthi’s six patterns of distributed dialogue, remote dialogue, distributed model, distributed data, remote data and stand-alone [10].

Current study about DSS architecture mainly focuses on the application of new IT technology. Zhang and Goddard presented the layered software architecture and component-based framework to assist in the design of a Web-based DSS, which provides a formal and hierarchical view of the Web-based DSS at the design stage [11].

Yang and Calmet proposed a Service oriented decision support system based on an ontology-driven uncertainty model to deal with uncertainty and structure complexity [12]. The general framework for the system consists of four layers as the repository layer, the management layer, the service layer and the application layer. As another case, Yuen Kevin Kam Fung presents in [13] an Enterprise Decision Platform (EDP) for the development of the adaptive decision applications. EDP applies the architecture concept of SOA which is implemented by web-services. However, these adoptions of Service of Computing technology are limited. In other words, they are not fully service oriented.

Figure 6. Screenshot of SBLMDSS
Dongjin Yu, Suhang Zheng
Advances in Information Sciences and Service Sciences, Volume3, Number7, August 2011

Cheung et al. adopted the Business Process Execution Language for Web Services in a service oriented distributed data mining (DDM) platform to choreograph DDM component Services and fulfill global data mining requirements [14]. Different with SBDSS, it was confined in the application of data mining, which is just one part of general DSS.

As a distributed computing infrastructure, the grid is now becoming one of supporting technology of DSS. Research projects such as the TeraGrid project and the Grid-DataMining project aim at developing data mining services on grids, whereas systems like the Knowledge Grid, Discovery Net, and GridMiner developed KDD systems for designing complete distributed knowledge discovery processes on grids. Congiusta et al. discussed how grid computing could be used to support distributed data mining [15]. However, grid only aimed to promote the utilization of decentralized resources, but not the adaptability as SBDSS would bring about.

6. Conclusions

Service Oriented Computing contributes to developing information systems that are compatible with the needs of agile organizations [16]. The study of Service based Decision Support System has significant practical values. Its achievements could be adopted to construct the next generation of Decision Support System, which helps decision makers adapt to the dynamic changing decision environments and demands in the flexible, agile and fast manner, compared to the many existing alternatives.

Software architecture must reflect the business goals of the enterprise that wishes to use the software system. Adaptability is one quality which represents the extent of a given DSS to satisfy the dynamic changing decision environments and individual decision styles [17]. However, there are many other business needs that would emphasize other quality attributes besides adaptability. Our future research mainly focuses on the other qualities of SBDSS, such as security, reliability and maintainability.

On the other hand, an evaluation framework still needs to be elaborated to validate the merits of the SBDSS approach. Current work just presented a specific sample system, which was not adequate to prove its structural adaptability or the benefit to human decision makers.

7. Acknowledgment

The work is supported by the Foundation of Key Science and Technology Projects (No. 2008C11099-1) of Zhejiang province of China, the National Innovation Fund for Technology-based Firms (No. 08C26213300677), and the Science Foundation of the Hangzhou Dianzi University (No. KYS055608069).

8. References

[7] Michael Beisiegel, Henning Blohm, Dave Booz, Mike Edwards, Oisin Hurley, Sabin Ileceanu, Alex Miller, Anish Karmarkar, Ashok Malhotra, Jim Marino, Martin Nally, Eric Newcomer,
Dongjin Yu, Suhang Zheng
Advances in Information Sciences and Service Sciences, Volume3, Number7, August 2011


