8-16-1996

ENTERPRISE SOFTWARE ARCHITECTURE: A CASE FOR DEVELOPMENT OF ENTERPRISE-WIDE INFORMATION SYSTEMS

Vassilka Kirova
Department of Computer and Information Science, New Jersey Institutie of Technology, kirova@cis.njit.edu

Wilhelm Rossak
Department of Computer and Information Science, New Jersey Institute of Technology

Thomas Marlowe
Department of Computer and Information Science, New Jersey Institute of technology

ENTERPRISE SOFTWARE ARCHITECTURE:
A CASE FOR DEVELOPMENT OF ENTERPRISE-WIDE INFORMATION SYSTEMS

Vassilka Kirova, Wilhelm Rossak and Thomas Marlowe
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, NJ, USA
e-mail: kirova@cis.njit.edu

Abstract

Under the pressure of constantly increasing organizational demands for cost saving, sophisticated, and finally integrated business applications, Information Systems are evolving into enterprise-wide integrated information processing facilities organized as a large-scale distributed systems of systems. The complexity of these systems and the process of their development and integration raise a whole set of new issues. To address them, new approaches are needed that: (1) cope with complexity and change, (2) support development of open systems, and (3) preserve consistency.

In this paper we present an architecture-based approach for development and evolution support of Enterprise-Wide Information Systems (EWIS). We introduce the concept of an Enterprise Software Architecture (ESA) and discuss its role as a means in the development process in respect to the above three issues. Finally, we identify an object-based ESA as an attractive alternative for development of EWIS.

1. Introduction

There is no doubt that today's enterprise information systems are only an approximation of the vision of an enterprise-wide integrated information facility. However, under the constant business pressure and the drive to meet the standards of modern information technology, characterized by an extensive sharing of services and information across an entire enterprise, these systems are steadily evolving into large-scale distributed systems of systems, composed of many long-lived, integrated business applications.

The long life-cycles and integration multiply the complexity of the overall system and complicate the process of its development and evolution support. They raise an array of issues and set new requirements for the way in which information systems are developed. Currently, each business application serving an enterprise is built without any concerns for the enterprise's overall needs and constitutes a stand-alone solution to a specific business problem. Applications are designed for simpler information needs and are based on closed software architectures. The underlying technologies and engineering traditions differ. The development efforts take place in isolation and are neither based on a common conceptual framework, nor do they envision the requirement for future interoperation with the rest of the enterprise's information-processing facilities. All these make so-called "post-facto" integration of an enterprise's collection of business applications a tedious, expensive and "per-case solution" task, and substantiate the need for new engineering strategies and new technologies.

The recognition of these issues and needs have led to proliferation of research and development efforts. Among others, these efforts focus on identification of effective software architectures - system organization and cooperation models - and on development of sound engineering methods and tools that can successfully cope with increased complexity. These efforts have been fueled by the rapid advancement of networking technology, by the wide application of decentralized distributed processing, and especially by the increased acceptance of object-oriented technology and strategies for distributed object management. They are also closely related to a set of standardization initiatives, such as CORBA, [8], and ISO/ODP, [1].

In the rest of this paper, we discuss the above issues and present an architecture-based solution. We do not address a specific architecture and its properties, but rather examine the role of an architectural description in the context of the development process.

We delineate how a rigorously specified software architecture can provide a common conceptual framework for design and implementation of coherent Enterprise-Wide Information Systems (EWIS) constructed from a consistent set of interoperable business applications.

2. What is an Enterprise Software Architecture?

We introduce the concept of an Enterprise Software Architecture (ESA) as a high level design framework for development and integration of business applications in the context of an enterprise, [4]. An ESA prescribes the organization and behavior of software systems as well as describes the available information processing services that support the operation of a particular enterprise. It constitutes a common design model for an enterprise's collection of software systems.

2.1. The Scope of an ESA

We will use three major, commonly-recognized in development of EWIS issues - coping with complexity and change, support for development of open systems, and preserving consistency - as a metric for evaluating our approach.

Complexity and Change:

As with any software architecture, an ESA provides a system description at a high level of abstraction where a software system is seen as a collection of high level interacting components constrained by a set of configuration and behavior rules. It encapsulates the irrelevant detail of individual systems, precisely defines their interfaces, and prescribes the way they will interoperate across an enterprise's full ensemble of software systems.

Thus, an ESA is characterized by a high level of abstraction, encapsulation, and modularity, which are key mechanisms for reducing complexity and managing change.

Openness:

Meeting the requirement for openness is a basic goal in development of enterprise-wide software solutions. The key point in this process is the concept of conformance.

Generally speaking, by conforming to a standard, a system becomes open to all other systems obeying the standard, where openness typically implies interoperability, portability and scalability. We apply this general idea of open systems to the development of an enterprise-specific set of business applications. Evidently, in order to develop such applications as open systems which can interoperate across an enterprise, meeting its information needs, an "enterprise software standard" is required that will provide a point of reference and conformance in the process of development and evolution (also selection and adaptation) of business applications for the particular enterprise. Being a common design model of an enterprise's collection of software applications, an ESA can successfully serve as such a standard, given that it is commonly accepted and rigorously specified (so conformance can be verified), [3].

Consistency:

The common design basis provided by an ESA and the systematic engineering approach established by using it as a reference model and a guiding plan in development of the constituents of an EWIS, imply consistency of the overall system.

2.2. The substance of an ESA

An enterprise software architecture must fit the form - the software system organization and behavior - to the function - the specifics of the business process and its information needs. Thus, it has to reflect two areas of knowledge - knowledge of the domain of software and knowledge of the business area.

The clear distinction between these two domains substantiates the separation of two aspects of an architectural specification: extra-functional (the system organization, behavior and properties) and functional (business-aware services). Based on this, fundamental to our approach, separation we identify two parts of an ESA specification: a conceptual architecture and an application architecture.

A conceptual architecture is concerned with the extra-functional aspect and: (1) prescribes the system organization (e.g., types of software components and types of configurations); (2) defines the model of interaction between the components (e.g., invocations and data types); and (3) provides design solutions for the set of required properties (e.g., transparency, transactions, reliability, security, etc.) in the form of so-called architectural services, [1], or in object-oriented terms - Enterprise System Objects.

All three elements of a conceptual architecture specification are independent from the business-specific logic and therefore guide acquisition and customization of an infrastructure for the enterprise (e.g., CORBA or DCE, [5]). The conceptual architecture also determines the organization of the overall integrated system (e.g., a 3-tier, [7] or OSCA, [2], based organization) and resolves the mapping of this organization model to the model of the selected infrastructure (if there is no "perfect" match). For example, reportedly ([6]) OMG's CORBA - ORB products are better suited for development of centralized applications, while choosing a decentralized model will require customization of the infrastructure - adding a layer of services over the basic CORBA-based software bus (e.g., services that support sending the state of objects from one process to another).

According to our architecture-based approach, such a customization process has to be documented into the conceptual architecture of the particular ESA. The new services become a part of architectural description which is then verified and validated to become an enterprise standard.

If software applications in an enterprise are designed in conformance with the conceptual portion of the ESA specification (and implemented correctly), they will be able to interact and exchange data successfully. This, however, is not sufficient for a meaningful exchange and reuse of information throughout an enterprise. In addition, a common application semantics has to be established.

To meet this need, we introduce an additional level of architectural descriptions, referred to as application architecture. An application architecture is an instantiation of the conceptual portion of an ESA, where business-aware system aspects are made explicit. It defines precisely the available business-specific services (e.g., Enterprise Business Objects) and identifies the configurations to be established in order to meet the functional requirements of the enterprise. Thus, the major responsibility for the application architecture is to support a consistent service and information view across an enterprise's collection of business applications.

This clear separation of concerns between a conceptual and application architecture creates opportunities for large scale (design) reuse. An established and verified conceptual architecture, such as a client-server based 3-tier architecture, for example, can serve a set of different enterprises and, hence, can be (re)used in the context of many ESA as the design solutions it captures are independent from the specifics of the business domain. According to our approach, such a reusable conceptual architecture has to be further instantiated into an application architecture, which will constitute a separate level of architectural specifications and will provide the developers and customers with a map of services specific for the particular business domain, e.g., Banking, Insurance, University, and Telecommunications.

Details on how to apply the approach presented here and some results from a small case-study for an Insurance enterprise can be found in [4].

3. ESA and Object Oriented Development

Increasingly, object-oriented methods and tools are being used to construct information systems. And what is more, the introduction of distributed objects, the strong industrial support, and the standardization, make the technology ever more promising as a basis for development of EWIS.

From an ESA perspective, applying an object-oriented architecture as a conceptual framework for development and integration of EWIS is only one of many possible architectural solutions. There are, however, several fundamental aspects of an object-oriented approach that make such a choice an attractive alternative:

(1) the object oriented paradigm provides mechanisms for building abstractions that, as we have seen above, is fundamental in architectural thinking;

(2) object orientation provides a natural abstract mechanism for communication and a framework for encapsulation and reuse needed to facilitate the integration of business applications;

(3) an object oriented approach emphasizes the objects in the problem domain and thus creates a framework for mapping the problem domain into an application architecture.

4. Conclusions

In summary, there are four important features of the presented architecture-based approach to development of EWIS:

(1) it leads to a system description at a high level of abstraction, which is key in coping with complexity;

(2) it creates a premise for development of open business applications, that can cooperate in achieving common goals;

(3) it encourages large scale design reuse with the requirement for a clear separation between the domain of software and the specific business domain; and

(4) it implies design discipline which is extremely important in development of EWIS.

References

[1] Basic Reference Model for ODP, ISO/IEC JTC 1/SC 21, Interim revised CD text, 1992.

[2] The Bellcore OSCA Architecture, Bellcore - Bell Communications Research, Technical Advisory, TA-STS-000915, Issue 3, March 1992.

[3] Kirova V., Rossak W., "Representing Architectural Designs: A Central Issue in

the Development of Complex Systems", Proc. of the IEEE ICECCS, 1995, pp. 80-87.

[4] Kirova V., Rossak W., Lawson H., "Software Architecture: An Analysis of Characteristics, Structure, and Application," In Proceedings of ICSE-17: IW on Architectures for Software Systems, 1995, pp. 166-175.

[5] Lockhart H., "OSF DCE Guide to Developing Distributed Applications", McGraw-Hill, Inc., 1994.

[6] Roy M., Ewald A., "Two Distributed Object Application Models", Object Magazine, September 1995, pp. 79-80.

[7] San Soucie M., Almarode J., "Supporting Enterprise-Wide Business Objects," Object Magazine, Vol. 5(5) pp. 44-59, September 1995.

[8] Vinoski S., "Distributed Object Computing With CORBA", C++ Report Magazine, July/August 1993, pp. 74-77.