COM and COM+ Programming: Unveiling Distributed Object-Oriented Technology

Introduction

COM and COM+, Unveiling Distributed Object-Oriented Technology, introduces the fundamental concepts and practical techniques of distributed object-oriented programming using Microsoft's Component Object Model (COM) and Component Services (COM+). This comprehensive guide provides a thorough understanding of the foundational principles, architectural components, and programming methodologies of COM and COM+, enabling you to create robust and scalable distributed applications.

Embark on a journey into the world of COM, where you'll discover the distributed object paradigm, explore

the benefits and challenges of distributed object-oriented programming, and delve into the intricacies of the COM architecture. Gain insights into the essential concepts of interfaces, COM objects, and servers, and equip yourself with the knowledge to navigate the complexities of COM security and threading.

Unravel the power of COM+ services, unlocking the potential for building transactional, queued, and event-driven components. Delve into the techniques for creating, consuming, and debugging COM+ applications, and master the best practices for effective COM+ programming. Explore advanced COM concepts, such as customizing, extending, and aggregating COM objects, and delve into advanced security mechanisms to enhance the robustness of your applications.

Discover the intricacies of Distributed COM (DCOM), understanding its architecture, configuration, and troubleshooting techniques. Implement object persistence with COM+, employing various strategies to

ensure the durability of your objects. Learn to leverage COM+ services to enhance your distributed applications, achieving scalability, reliability, load balancing, and failover. Explore the future of COM and COM+, examining the challenges and limitations of these technologies and the emerging trends that are shaping the landscape of distributed object-oriented programming.

Through a series of engaging case studies and real-world examples, COM and COM+, Unveiling Distributed Object-Oriented Technology, provides a practical and comprehensive guide for software developers, architects, and system administrators who seek to master the art of building distributed applications with COM and COM+. Join us on this enlightening journey into the realm of distributed object-oriented programming and unlock the full potential of COM and COM+.

Book Description

In a world of interconnected systems and distributed computing, COM and COM+ Programming: Unveiling Distributed Object-Oriented Technology stands as an indispensable guide for software developers seeking to harness the power of distributed object-oriented programming. This comprehensive book provides a profound understanding of the Component Object Model (COM) and Component Services (COM+), the cornerstone technologies that have revolutionized the way software components interact and communicate across networks.

Delve into the depths of COM, exploring its fundamental concepts, architectural components, and programming methodologies. Grasp the essence of distributed object-oriented programming, unlocking its benefits and overcoming its challenges. Master the intricacies of COM interfaces, objects, and servers,

gaining the knowledge to construct robust and scalable distributed applications.

Unveil the power of COM+ services, unlocking the potential to create transactional, queued, and event-driven components. Delve into the techniques for developing, consuming, and debugging COM+ applications, and master the best practices for effective COM+ programming. Explore advanced COM concepts, such as customizing, extending, and aggregating COM objects, and delve into advanced security mechanisms to enhance the resilience of your applications.

Discover the intricacies of Distributed COM (DCOM), understanding its architecture, configuration, and troubleshooting techniques. Implement object persistence with COM+, employing various strategies to ensure the durability of your objects. Learn to leverage COM+ services to enhance your distributed applications, achieving scalability, reliability, load balancing, and failover. Explore the future of COM and

COM+, examining the challenges and limitations of these technologies and the emerging trends that are shaping the landscape of distributed object-oriented programming.

Through a series of engaging case studies and real-world examples, COM and COM+ Programming: Unveiling Distributed Object-Oriented Technology provides a practical and comprehensive guide for software developers, architects, and system administrators who seek to master the art of building distributed applications with COM and COM+. Join us on this enlightening journey into the realm of distributed object-oriented programming and unlock the full potential of COM and COM+.

Chapter 1: Embracing Distributed Object-Oriented Programming

Distributed Object Concepts

Distributed object-oriented programming (DOOP) is a programming paradigm that enables objects to be distributed across multiple machines and communicate with each other as if they were on the same machine. This allows for the creation of distributed applications that can scale to large numbers of users and data.

DOOP is based on the fundamental concept of distributed objects. A distributed object is an object that exists on a remote machine and can be accessed by other objects on different machines. Distributed objects communicate with each other by sending messages. These messages are typically sent over a network connection.

There are a number of benefits to using DOOP. These benefits include:

- **Scalability:** DOOP allows applications to scale to large numbers of users and data by distributing the objects across multiple machines. This can improve performance and reliability.
- **Modularity:** DOOP allows applications to be divided into smaller, more manageable components. These components can be developed and maintained independently, which can make it easier to update and maintain the application.
- **Transparency:** DOOP allows objects to be accessed in a transparent manner, regardless of their location. This means that developers can write code that accesses distributed objects in the same way that they would access local objects.

DOOP is a powerful programming paradigm that can be used to create scalable, modular, and transparent applications. It is a valuable tool for software developers who need to create distributed applications.

The Dance of Light and Shadows

DOOP can be used to create a variety of distributed applications. One example is a distributed game. In a distributed game, the game world is divided into multiple regions, and each region is hosted on a different server. Players can move their characters between regions, and the game server will automatically transfer the player's data to the new region. This allows players to interact with each other in a seamless manner, regardless of their location.

Another example of a distributed application is a distributed database. In a distributed database, the data is stored across multiple machines. This allows the database to scale to large amounts of data and improve performance. Distributed databases are often used by large organizations that need to store and manage large amounts of data.

DOOP is a powerful tool that can be used to create a variety of scalable, modular, and transparent

applications. It is a valuable asset for software developers who need to create distributed applications.

Chapter 1: Embracing Distributed Object-Oriented Programming

Benefits of Distributed Object-Oriented Programming

Distributed object-oriented programming (DOOP) offers a plethora of benefits that make it an attractive choice for developing complex and scalable distributed applications. Let's delve into some of the key advantages of DOOP:

Modularity and Reusability: DOOP promotes modularity by allowing you to decompose your application into independent, reusable components called objects. These objects can be easily combined and reused across different applications, reducing development time and increasing code maintainability.

Transparency: DOOP provides transparency in several aspects. It hides the underlying communication details

from the developer, enabling seamless interaction between objects regardless of their physical location. Additionally, DOOP allows you to access remote objects as if they were local, simplifying the development process.

Scalability: DOOP enables the construction of highly scalable applications that can seamlessly handle increasing loads and accommodate a growing number of users. By distributing objects across multiple servers, DOOP allows you to scale your application horizontally, improving performance and reliability.

Concurrency: DOOP supports concurrency by allowing multiple objects to execute simultaneously. This enables the development of responsive and efficient applications that can handle multiple requests concurrently, improving overall performance and user experience.

Flexibility: DOOP provides flexibility in designing and implementing distributed systems. It allows you to

choose the appropriate communication mechanisms and protocols based on the specific requirements of your application. This flexibility enables you to tailor your system to meet your unique needs and constraints.

Interoperability: DOOP promotes interoperability by allowing objects developed using different programming languages and technologies to communicate and interact seamlessly. This interoperability enables the integration of diverse components and services into a cohesive distributed system.

In essence, DOOP offers a powerful paradigm for developing robust, scalable, and maintainable distributed applications. Its benefits make it an ideal choice for building complex systems that require high performance, flexibility, and interoperability.

Chapter 1: Embracing Distributed Object-Oriented Programming

Architectural Overview of Distributed Object Systems

Distributed object systems (DOS) represent a paradigm shift in software architecture, enabling the construction of applications that seamlessly integrate objects located across a network. This architectural approach offers numerous benefits, including:

- **Transparency:** DOS provides a transparent layer of communication between objects, allowing developers to invoke methods on remote objects as if they were local. This transparency simplifies the development and maintenance of distributed applications.
- **Scalability:** DOS enables applications to scale horizontally by distributing objects across

multiple servers. This scalability allows applications to handle increasing loads and accommodate growing user bases.

- **Fault Tolerance:** DOS provides mechanisms for fault tolerance, such as replication and failover, ensuring that applications can continue to operate even in the event of server or network failures.

- **Interoperability:** DOS enables objects developed using different programming languages and running on different platforms to communicate and interact with each other. This interoperability promotes software reusability and simplifies the integration of heterogeneous systems.

The fundamental concepts underpinning DOS include:

- **Objects:** Objects are the basic building blocks of DOS. They encapsulate data and behavior and

communicate with each other through method invocations.

- **Interfaces:** Interfaces define the methods that an object provides. They allow objects to interact with each other without being concerned with the underlying implementation details.
- **Remote Procedure Calls (RPCs):** RPCs are the mechanism by which objects invoke methods on remote objects. RPCs transparently handle the details of message encoding, network communication, and error handling.

The architectural components of a DOS typically include:

- **Client:** The client is the application that initiates communication with a remote object.
- **Server:** The server is the application that hosts the remote object and processes method invocations from clients.

- **Object Request Broker (ORB):** The ORB is a middleware component that manages communication between clients and servers. It transparently handles the details of object location, message routing, and error handling.

Distributed object systems have revolutionized the way software applications are designed and deployed. By providing transparency, scalability, fault tolerance, and interoperability, DOS has enabled the development of complex and sophisticated distributed applications that seamlessly integrate objects across a network.

This extract presents the opening three sections of the first chapter.

Discover the complete 10 chapters and 50 sections by purchasing the book, now available in various formats.

Table of Contents

Chapter 8: Implementing Object Persistence with COM+ * Object Persistence Concepts * Implementing Object Persistence with COM+ * Configuring Object Persistence in COM+ * Troubleshooting Object Persistence Issues * Case Study: Implementing Object Persistence in a COM+ Application

Chapter 9: Enhancing Distributed Applications with COM+ Services * Using COM+ Services to Enhance Applications * Building Scalable and Reliable Distributed Applications with COM+ * Implementing Load Balancing and Failover with COM+ * Monitoring and Managing COM+ Applications * Case Study: Enhancing a Distributed Application with COM+ Services

Chapter 10: The Future of COM and COM+ * COM and COM+ in the Modern Era * Challenges and Limitations of COM and COM+ * Emerging Technologies and Their Impact on COM and COM+ * The Future of Distributed

Object-Oriented Programming * Case Study: Migration from COM/COM+ to Modern Technologies

This extract presents the opening three sections of the first chapter.

Discover the complete 10 chapters and 50 sections by purchasing the book, now available in various formats.