SCITECH
RESEARCH ORGANISATION|
Volume 4, Issue 1
May 29, 2015|
Journal of Information Sciences and Computing Technologies
www.scitecresearch.com

SOA-Based Distributed System in Online Transaction Processing

Abu Sarwar Zamani[1], Pankaj Kumar Gupta[2]
[1]Research Scholar, Pacific University, Udaipur, India
[2]Associate Professor,Tirupati College of Technical Education, Jaipur, India

Abstract

Service Oriented Architecture (SOA) with transactional workflow support is a state- of- the- art architectural style for constructing enterprise application. In this research, rate of progress activities raise distributed service in a coordinated manner, using transaction context propagating message, coordination protocol and compensation logic. We reviewed the past, present and future of transaction processing and transaction integrity. Most of the challenges and requirement that led to the development and evolution of transaction processing system are still applicable today and recently, we have some intriguing developments. We take an explorative approach to probe the theoretical and implementational feasibility of managing transaction in the web service world.

Keywords: Transaction; SOA; workflow; Distributed service/system.

1. Introduction

At present Service-oriented architecture (SOA) is no uniform definition. The more influential SOA is defined as: "essentially a collection of services. Mutual communication between services may be simple data transferred; it may be conduct two or more coordinating services active. It needs some method of inter-service connections. The so-called service is a function with a precise definition, perfect package, independent of other the environment and state serving. "Service-oriented architecture (SOA) is an organization based on service computing resources, with loosely coupled services and indirect addressing capability of the software architecture. In essence, SOA is service-oriented software architecture, to design and build a loosely coupled software solutions approach. The basic elements of SOA architecture is service for business processes as reusable components that simplify the information services or the state of the data migration process, to respond to customer requests and provide high quality services. In SOA- based system integration, long-running business transaction often involve incompatible trust domains, asynchrony and periods of inactivity, presenting challenges to traditional ACID-style transaction processing. We take an explorative approach to probe the theoretical and implementation feasibility of managing transaction in the web service world. Following the theoretical thread, we propose a mental reference model to adapt existing transaction theories, including the classical ACID models which are below:

a. **Atomicity:** The transaction executes completely or not at all.
b. **Consistency:** The transaction preserves the internal consistency of the database.
c. **Isolation:** The transaction executes as if it were running alone, with no other transactions.
d. **Durability:** The transaction's results will not be lost in a failure.

First, a transaction needs to be **atomic** (or **all-or-nothing**), meaning that it executes completely or not at all. There must not be any possibility that only part of a transaction program is executed. For example, suppose we have a transaction program that moves $100 from account A to account B. It takes $100 out of account A and adds it to account B. When this runs as a transaction, it has to be atomic — either both or neither of the updates execute. It must not be possible for it to execute one of the updates and not the other. The TP system guarantees atomicity through database mechanisms that track the execution of the transaction. If the transaction program should fail for some reason before it completes its work,

the TP system will undo the effects of any updates that the transaction program has already done. Only if it gets to the very end and performs all of its updates will the TP system allow the updates to become a permanent part of the database.

If the TP system fails, then as part of its recovery actions it undoes the effects of all updates by all transactions that were executing at the time of the failure. This ensures the database is returned to a known state following a failure, reducing the requirement for manual intervention during restart.

2. Business Relevant Information

The objective of our work is to support the design and development of *enterprise applications* that require transactional semantics. An example is a Customer Relationship Management (CRM) system that serves many concurrent users via multiple access channels and processes, including an Internet self-service and a call center. In This CRM, business-relevant customer profile information is persisted in databases and accessed via Web-accessible services; external systems also have to be integrated.

2.1 SOA and Web services

SOA reinforces general software architecture principles such as separation of concerns and logical layering. A defining element of SOA as an architectural style is the possibility to introduce a *Service Composition Layer(SCL)*, which promises to increase flexibility and agility and to provide better responsiveness to constantly changing business environments. (Re-)assembling workflows in the SCL does not cause changes on the underlying service and resource layers; computational logic and enterprise resource management are separated from the service composition. We refer to a SOA with such a SCL as a *process-enabled SOA*.

XML-based Web services are a state-of-the-art implementation option for process enabled SOAs. The Web Services Description Language(WSDL) describes service interfaces, SOAP service invocation messages. BPEL is a workflow language with operational semantics that can be used to realize the SCL. Component models for the implementation of services are emerging; SCA is such a model. Service components in SCA are defined from several perspectives: an *interface* describing the input and out put parameters of the operations of a component, *references* to other components, and component *implementations*. Via *imports*, a component implementation can reference external services.

2.2 Transactional Workflow

Transactional workflows coordinate the outcome of the local and remote service invocations that access and manipulate the enterprise resources. Transactional workflows in process-enabled SOA are particularly challenging to design due to the potentially long-lived nature of processes, the loose coupling and autonomy of services, the existence of non-transactional resources, and the diversity in coordination and communication protocols synchronous and asynchronous message exchange patterns). Traditional system transactions alone are not directly applicable in a SOA setting; a more decentralized *coordination* model and application-level *compensation* strategies have to be added.

3. Web service oriented transaction processing Model

Concentrates on adapting these models to meet transaction processing needs in a service-oriented architecture. SOA impacts the understanding of transactions, and why the ACID-style transaction models are inadequate for SOA. Major challenges in web service transaction management are emphasized, and the analyses of how to meet these challenges are operationalized into a set of eight design criteria. This is followed by a brief introduction of the WS-TX protocol family, which are web service standards governing the transaction space. Based on the design criteria and the web service standards, a reference model for web service transaction management is constructed and presented.

Solving transaction problems at the web service level introduces an extra degree of complexity, which makes it a less efficient solution compared to transaction management at the application or database level. When used in system integration, web services usually wrap around coarse-grained functionality in legacy systems which otherwise would not be able to communicate. These legacy systems typically run in different execution environments with platform-specific transaction support, usually an implementation of the strict ACID-style two-phase or three-phase commit. If all services in a transaction are managed entirely within a single execution environment, web service transaction management will have no role to play. Transaction management at the web service level should only be considered when using web services to integrate disparate systems, as shown in Figure 1.

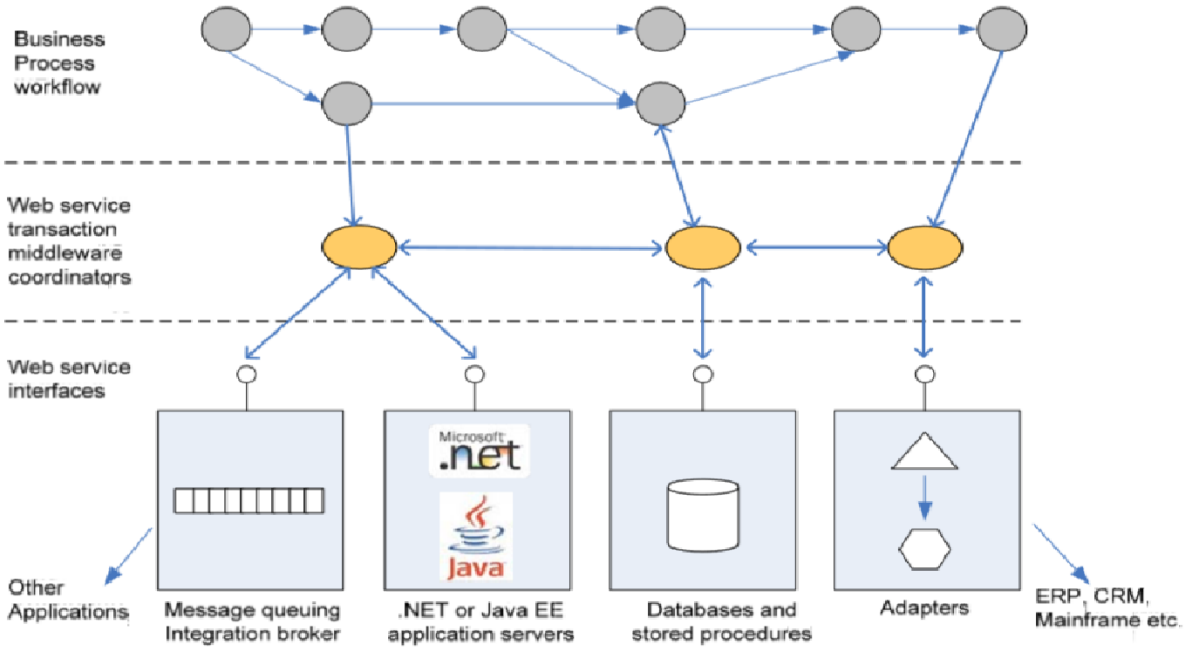

Figure 1: Integrating transaction domains in SOA

3.1 Interoperability

A web service transaction middleware consists of, among other things, transaction coordinators. The purpose is to coordinate the system to reach common decision on whether to commit, rollback or compensate the changes done by the business workflow. Interoperability is provided by wrapping legacy systems behind a web service interface, between otherwise incompatible systems. Examples of such system are SAP, .NET, EJB, CORBA, JMS, etc.

3.2 Stateless Service vs. Statefull Transaction

As statelessness is a key service-oriented principle, requiring individual services to retain state for the sake of transaction management seems to be a step back from the service-oriented ideal. The paradox between statefull transactions and stateless services is exacerbated by the necessity of treating legacy business services as "black-boxes" in system integration.

The composition principle in SOA makes it possible to aggregate several business services in a composite service. In this research, the three terms – composite service, service orchestration and business activity – will be used as synonyms. The *subsidy application service* in the CAP case is an example of a composite service, representing the entry point to a long-running activity and a semi-automated business process. Every service orchestration introduces a level of context into an application runtime environment. The more complex a business activity, the more context information it tends to bring with it. Managing transaction contexts in service orchestrations inevitably requires managing and propagating these contexts, or transactional states.

3.3 Heterogeneous transactional requirements in SOA

Orchestrated services have varying transaction requirements, the relationship of a transaction to a legacy web service might be as simple as delegating a transaction to an existing transaction execution environment for the legacy service. It may also be as complex as coordinating a single transaction across multiple Participants in a long-running business process, across arbitrary execution environments. A variety of transaction protocols should be available for use with different transactional requirements, ranging from tighter-coupled, short-lived strict ACID transactions to loosely-coupled and long-running automated business process executions.

3.4 Composable Transaction Model

In some complex web service workflows, certain stages of a service orchestration require a strict atomic outcome, while other stages can have more relaxed ACID requirements. This is the case in our *subsidy application service* example, where the top-level service is a long-running business transaction, which entails three subtractions of different types - a flat atomic transaction (the *account payable* service), a nested atomic transaction (the *case registration* service) and a long-running business transaction (*eligibility evaluation* service). These heterogeneous nested transaction types represent the requirement which gives rise to our sixth design criterion:

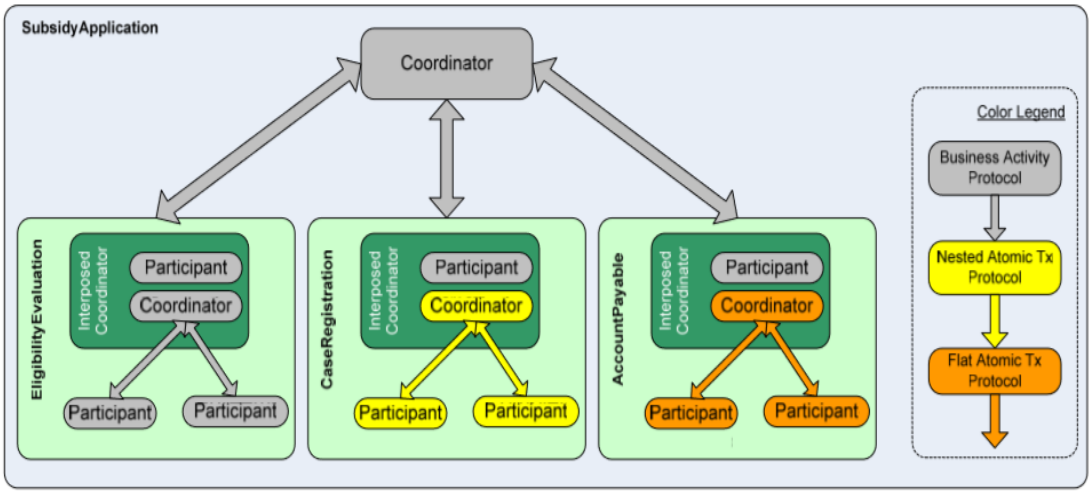

Figure 2: A coordination hierarchy with embedded transaction models

4. WS(Web Services) Transaction Standard

Several competing web service transaction specifications have been proposed. We have chosen to base our prototype design upon the WS-Transaction specifications, *i.e.* the WS-TX (Web Services Transaction) protocol family. The WS-TX protocol family comprises three specifications: WSCOOR (Coordination Framework, part of WS-TX), WS-AT(Web Services Atomic Transaction) and WS-BA (Web Services Business Activity). These specifications are, from our vantage point, a good match to what we believe should be captured by a good web service transaction processing model.

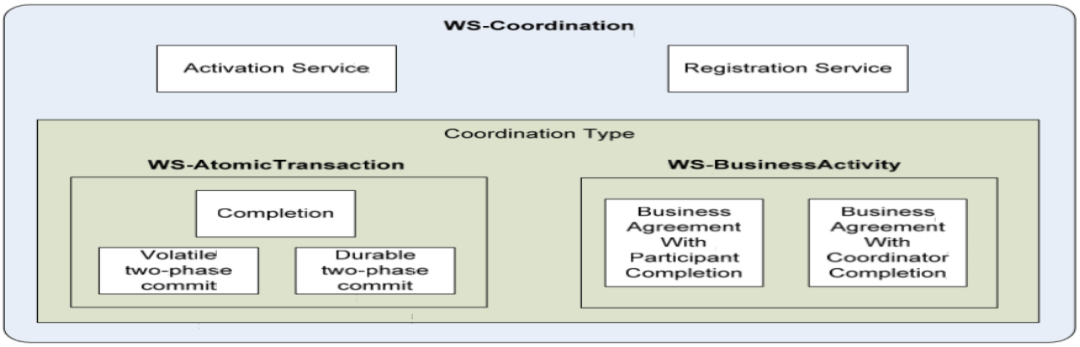

Figure 3: WS-Transaction services and protocols

5. Managing data in Distributed Online Transaction Processing Environment

There is huge database bottleneck that cause major problems of retailers and other organizations with highly distributed online environments. Databases containing customers and product data must be read from and written to constantly and in near real time to support the quality and timelines of each transaction. As for example, after a customer places his selections in a shopping cart, data on product price and availability in addition to customer address and credit must be instantly accessed, analyzed and updated before the transaction can be completed. In a distributed IT environment, various data elements required for this transaction are likely to be located in multiple nodes across the entire system. Making sure that the right data is available at the right time to provide a seamless and high quality experience for the user is a goal that gets increasingly harder to achieve for some companies as they grow their business.

Sometimes a customer is ready to buy only to determine that shipping costs are much higher than they had anticipated putting the purchase at risk. If product inventory data across multiple warehouses is not well integrated, a buyer might be told the product is on backorder when it is actually available in an alternative warehouse. This type of problem happens because there is too much data that needs to be analyzed, accessed and coordinated in real time.

How can companies deal with these complex real times transaction management issues? In essence, companies follow one of the three available options described below:

a. Use of in-memory data grids

This method supports the caching of frequently used data in a distributed, in-memory data grid. The quantity and variety of data that must be cached in-memory is often expansive including data related to customers, individual products, shipping locations, third party suppliers, warehouse, and inventory. While this approach helps to eliminate data bottlenecks and can lead to significant improvements in transaction performance, it comes with the real risk of data loss when nodes fail. Because data and transactions are stored in memory only, loss of a node can mean failed transactions and loss of revenue. Even with the deployment of elaborate backup and recovery plans, most companies find this approach insufficient to support business-critical OLTP applications.

b. Use of in-memory data grids combined with manual management of persistence

This method uses advanced software development designed to improve a company's success with in-memory data grids. One important aspect of this method is the implementation of the two-phase commit protocol to eliminate some of the risk of data loss. The two-phase commit protocol is a technical algorithm that provides the required coordination between all the processes that must take place to persist a transaction accurately to permanent disk. This approach can take many months to implement in a distributed environment and requires significant development knowledge and on going manual software code updates. The software engineers need to know the server location of all the data elements required for a transaction so that the data can quickly be read, reviewed, and then written in the appropriate location to keep up with the desired transaction speed. This approach requires highly skilled engineers to successfully handle moderate transaction loads with software that is written for a specific use case. Most companies find that, persisting data in this way results in systems insufficient to support more extreme loads.

c. Keep with the status quo and react to exceptions

This approach is very common in retail and other distributed transaction environments. Sometimes retailers in highly competitive markets will cut corners when it comes to designing fault tolerant architectures and quality control measures for data because they are under pressure to get to market quickly with new features and functionality. They may also eliminate some system safeguards that might slow down the system because they want to ensure that the online shopping environment performs well during spikes in demand. As a result, the system may maintain performance levels, but there are lots of places where errors may occur. In systems like this, the engineering teams analyze the product and customer databases every night looking for places where the integrity of the data has been disturbed and they make corrections as needed.

6. Summary and Outlook

Transaction models are becoming the predominant web service transaction management protocol in lieu of the ACID-style transactions. Appropriate countermeasures must be used to provide alternative reliability guarantees for compensation-based transaction protocols. Various possible combinations of web services within a transaction often require the use of multiple protocols and an external Coordinator capable of bridging disparate execution environments. We introduced a new analysis and design method leveraging architectural decision models and patterns in support of the full lifecycle of designing transactional workflows, a particularly challenging problem in the construction of process-enabled SOA. We then defined three conceptual patterns, Process activity, Communication Infrastructure and Service Provider level. Future work includes documenting more variations and pattern selection guidance for our three patterns. The three primitives can be mapped to more runtime platforms such as the Spring framework. To extend the method, architectural patterns for other recurring decisions, for example business-level compensation, should be documented.

References

[1] Bhiri, S., Gaaloul, K., Perrin, O., Godart, C.: Overview of Transactional Patterns: Combining Workflow Flexibility and Transactional Reliability for Composite Web Services. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, Springer, Heidelberg (2005).
[2] Fowler, M.: Patterns of Enterprise Application Architecture. Addison Wesley, Reading (2003).
[3] Gray, J., Reuter, A.: Transaction Processing: Concepts and Techniques. Morgan Kaufman Publishers, San Francisco (1993).
[4] Abu Sarwar Zamani, Mohammad Jawed Miandad & Shakir Khan, "Data Center- Based Service Oriented Architecture (SOA) in Cloud Computing", International Journal of Computing Science and Information Technology (IJCSIT), Volume 1, January 2013 in India.
[5] Papazoglou, M., Kratz, B.: A Business-aware Web Services Transaction Model. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, Springer, Heidelberg (2006).
[6] Olaf Zimmermann, Jonas Grundler, Stefan Tai, and Frank Leymann, "Architectural Decisions and Patterns for Transactional Workflows in SOA".
[7] WS-Atomic Transaction: WS-Business Activity Framework, WS-Coordination, http://www.ibm.com/developerworks/library/specification/ws-tx

[8] Zimmermann, O., Doubrovski, V., Grundler, J., Hogg, K.: Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario. In: OOPSLA 2005 Conference Companion, ACM Press, New York (2005).
[9] Zimmermann, O., Gschwind, T., Küster, J., Leymann, F., Schuster, N.: Reusable Architectural Decision Models for Enterprise Application Development. In: Overhage, S., Szyperski, C. (eds.) Proc. of QoSA 2007. LNCS, Springer, Heidelberg (2007).
[10] Marcia Kaufman, COO and Principal Analyst, The Challenge of Managing On-line Transaction Processing Applications in the Cloud Computing World, Hurwitz & Associates.
[11] Arne John Glenstrup, "Distributed Transaction Management in SOA-based System Integration", IT University of Copenhagen, 2007.
[12] [BA-Interop, 2006] OASIS. *WS-BusinessActivity Interop Scenarios 1.1(Working Draft).* November, 2006.
[13] [Bernstein, 1997] Philip A. Bernstein and Eric Newcomer. *Principles of Transaction Processing.* Morgan Kaufman.
[14] [BTP] Business Transaction Protocol (BTP) Committee Specification (2002). OASIS http://www.oasis-open.org/committees/business-transactions.
[15] [Casavant, 1990] and . *A Communicating Finite Automata Approach to Modeling Distributed Computation and its Application to Distributed Decision-Making.* Thomas L. Casavant Jon G. Kuhl IEEE Computer Society, Washington, DC, USA.
[16] [Erl, 2004] Thomas Erl. *Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services.* Prentice Hall. New Jersey, USA.
[17] [Erl, 2005] Thomas Erl. *Service-Oriented Architecture – Concepts, Technology, and Design.* Prentice Hall. New Jersey, USA.
[18] [Erven, 2007] H. Erven, H. Hicker, C. Huemer and M. Zapletal. *The Web Services BusinessActivity-Initiator (WS-BA-I) Protocol: an Extension to the Web Services BusinessActivity Specification* Presented at theIEEE Int'l. Conference on Web Services, March 2007.
[19] [Frank, 2006] Lars Frank. *Databases with Relaxed ACID Properties.* Copenhagen Business School Press.
[20] [JBoss, 2006] JBoss. *JBoss Transaction Service 4.2.2 – Web Service Transaction Programmer's Guide*. http://www.redhat.com/docs/manuals/jboss/jboss-eap-4.2/doc/jbossts/JTA_Programmers_Guide.pdf
[21] [Little, 2003-1] Mark Little and Thomas Freund. *A Comparison of Web services transaction protocols*. http://www-128.ibm.com/developerworks/webservices/library/ws-comproto/
[22] [Newcomer, 2004] Eric Newcomer, Greg Lomow.*Understanding SOA with Web Services.Chapter 10. Transaction Processing.* Addison-Wesley Professional, UK.
[23] [Tanenbaum, 2006] Andres S. Tanenbaum and Maarten Van Steen. *Distibuted Systems – Principles and Paradigms.* Pearson Prentice Hall, New Jersey, USA.
[24] [Weerawarana, 2005] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey and Donald F. Ferguson.*Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Chapter 11. Transactions*. Pearson Prentice Hall, New Jersey, USA.
[25] [WS-AT] Web Services Atomic Transaction (WS-AtomicTransaction) v1.1. OASIS, April 2007 http://docs.oasis-open.org/ws-tx/wsat/2006/06
[26] [WS-BA] Web Services Business Activity (WS-BusinessActivity) v1.1. OASIS, April 2007 http://docs.oasis-open.org/ws-tx/wsba/2006/06
[27] Muhammad Younas and Kuo-Ming Chao. *A tentative commit protocol for composite web services*. In *Journal of Computer and System Sciences*, Volume 72, Issue 7, Pages 1226-1237, November 2006.
[28] [Zhao, 2005] Wenbing Zhao, L. E. Moser and P. M. Melliar-Smith. *A Reservation Based Coordination Protocol for Web Services*. In *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. IEEE Computer Society, 2005.

Authors' Biography

Abu Sarwar Zamani is doing PhD form Pacific University, Udaipur, India. He has received his Master of Science in Computer Science from Jamia Hamdard (Hamdard University), New Delhi, India in the year of 2007, and later he did Master of Philosophy in Computer Science from Vinayak Mission University, Chennai, India in 2009.He is a Member of International Association of Computer Science and Information Technology- IACSIT, (Singapore, Member No:80341225), Member of International Journal of Computer Science & Emerging Technology-IJCSET (UK), Program Committee Member of Academy & Industry Research Collaboration Center (AIRCC), Member of International Association of Engineers-IAE (Hong Kong, Member No:113797) and Member of IEEE. He has actively attended and published various research papers in National as well as International.

Dr. Pankaj Gupta is associated as Professor in Department of IT and Management in Tirupati College of Technical Education,Jaipur (India), he has completed his Ph.D research in the field of E-Commerce from the University of Rajasthan. He has masters in Information Technology and MBA in Information Systems. His more than 15 years of experienceincludes teaching, guiding in research, and working in various research and development projects.

Dr. Gupta has also written a number of articles and research papers in prestigious national and international journals and magazines. He has also written a number of books on 'Software Engineering', 'Computer Fundamentals','Network Technologies and TCP/IP', 'Object Oriented Programming', 'E-Commerce' and 'Management Information Systems'.He is always ready to welcome new ideas and innovations to update his knowledge and improve the quality of work.