Metal Ions in Life Sciences

WebSphere Modernization in Healthcare IT Systems: A Practical Guide to Migrating Clinical Middleware to Open-Source JBoss for Interoperable Hospital Platforms

Dr. Sofia Marinova[1]*
Prof. Lucas Meyer[2]
[1] University of Athens, Department of Health Informatics and Clinical Systems Engineering, Athens, Greece
[2] University of Zurich, Institute of Biomedical Cloud Migration and Enterprise Health Architecture, Zurich, Switzerland

Abstract- Enterprise IT landscapes are increasingly seeking modernization pathways to reduce costs, enhance flexibility, and adopt cloud-native architectures. IBM WebSphere, long a cornerstone of enterprise middleware, presents challenges related to licensing costs, vendor lock-in, and scalability limitations. Red Hat JBoss (WildFly), as an open-source alternative, offers modular architecture, community-driven support, and adaptability for modern DevOps and containerized deployments. This review article provides a comprehensive guide for migrating from WebSphere to JBoss, covering architectural comparisons, migration planning, technical execution, integration strategies, testing methodologies, and post-migration operational considerations. Practical insights are drawn from case studies and lessons learned, emphasizing best practices in risk mitigation, automated deployment, security compliance, and performance optimization. By systematically analyzing the migration process, the article equips IT architects, system administrators, and enterprise decision-makers with actionable strategies to achieve a seamless transition while maintaining operational continuity, performance reliability, and long-term scalability.

Keywords - WebSphere modernization, JBoss migration, middleware migration, enterprise application refactoring, open-source middleware, application integration, CI/CD deployment, cloud-native architecture, performance optimization, IT modernization strategy

INTRODUCTION

Background and Motivation

Enterprise IT environments have long relied on IBM WebSphere Application Server (WAS) as a cornerstone for hosting critical business applications. WebSphere's robust architecture, transactional integrity, and support for complex enterprise workflows made it the preferred middleware for large organizations over the past decades. However, evolving business requirements, the increasing cost of proprietary licensing, and the growing emphasis on agile, cloud-native solutions have highlighted limitations inherent in traditional WebSphere deployments. Many enterprises are encountering challenges such as vendor lock-in, complex upgrade cycles, and difficulty scaling dynamically in hybrid or cloud-based environments. These pressures have driven the exploration of open-source alternatives, with Red Hat JBoss (WildFly) emerging as a leading candidate due to its flexibility, active community support, and cost-effectiveness. JBoss provides comparable enterprise-grade capabilities while offering a modernized approach to application deployment, containerization, and microservices integration. By adopting open-source middleware, organizations can reduce licensing costs, improve system agility, and position their IT landscape for cloud-native transformations.

Scope and Objectives

This review article aims to provide a comprehensive roadmap for enterprises planning to migrate from WebSphere to JBoss. The scope includes a detailed exploration of technical, operational, and strategic considerations, covering architectural differences, application refactoring, data migration, integration challenges, and performance validation. The primary objective is to offer a practical guide that balances theoretical insights with actionable steps for successful migration. Key goals include evaluating the migration risks, outlining best practices for planning and execution, and illustrating lessons learned from real-world case studies. Additionally, the review emphasizes the post-migration operational landscape, addressing deployment automation, monitoring, security, and continuous integration to ensure sustained system performance and resilience. By examining both the opportunities and challenges of WebSphere modernization, this article provides IT architects, system administrators, and enterprise decision-makers with the insights required to make informed migration decisions while minimizing operational disruptions and maximizing return on investment.

II. COMPARATIVE ANALYSIS: WEBSPHERE VS. JBOSS

Architectural Differences

The architectural landscape of IBM WebSphere and Red Hat JBoss exhibits both similarities and critical distinctions that directly impact migration planning. WebSphere is traditionally monolithic, with a tightly integrated application server environment that emphasizes transactional integrity, built-in clustering, and enterprise-level reliability. It supports advanced features such as sophisticated messaging (JMS), EJB containers, and complex session management. JBoss, on the other hand, follows a more modular, lightweight architecture. With its WildFly foundation, JBoss provides flexible deployment units, enabling microservices and containerized applications while supporting standard Java EE specifications. This modularity allows organizations to scale services independently, adopt cloud-native patterns, and integrate modern DevOps practices. From a configuration standpoint, WebSphere relies heavily on proprietary management tools, whereas JBoss leverages open-source configuration

mechanisms, including XML-based descriptors and CLI automation, offering greater flexibility but requiring detailed knowledge of system internals during migration.

Licensing, Cost, and Support Implications
A major differentiator between the two platforms is licensing and total cost of ownership. WebSphere licenses are typically expensive, with ongoing support and upgrade fees adding to the operational burden. In contrast, JBoss is open-source, with Red Hat providing optional enterprise support subscriptions, significantly reducing upfront and recurring costs. This cost efficiency enables organizations to allocate resources toward modernization, performance optimization, and cloud adoption initiatives rather than licensing overhead. However, it is crucial to consider the trade-offs: while community support is robust, enterprises relying solely on open-source resources may face challenges in troubleshooting complex production issues, making a support subscription often essential for mission-critical deployments.

Performance and Scalability Considerations
Performance and scalability also differentiate WebSphere and JBoss deployments. WebSphere excels in high-throughput transactional environments, offering extensive clustering capabilities and sophisticated session replication mechanisms. JBoss, while capable of handling large-scale workloads, requires careful tuning of thread pools, connection pools, and clustering configurations to achieve comparable performance. Additionally, JBoss's lightweight footprint and container-friendly design make it highly suitable for dynamic cloud environments, enabling horizontal scaling and rapid resource provisioning. Monitoring and optimization tools differ as well, with WebSphere providing proprietary performance consoles, while JBoss integrates with open-source solutions like Prometheus and Grafana, allowing more customizable and extensible monitoring pipelines.

III. MIGRATION PLANNING AND ASSESSMENT

Current Environment Assessment
A successful migration from WebSphere to JBoss begins with a thorough assessment of the existing environment. Organizations must inventory all deployed applications, middleware components, and their interdependencies to understand the complexity of the migration. This includes evaluating enterprise services such as EJB modules, JMS queues, data sources, and web service endpoints. Dependencies on proprietary WebSphere features, such as advanced messaging patterns or integration with IBM-specific connectors, must be identified early, as they often require custom adaptation or alternative implementations in JBoss. Additionally, understanding resource utilization, server configurations, and clustering arrangements provides a baseline for capacity planning and ensures that the new environment can meet current and future performance requirements.

Risk Assessment and Mitigation
Migration inherently involves risks that, if unaddressed, can result in downtime, data loss, or operational disruption. Common risks include incompatibility of application code with the JBoss platform, data inconsistencies during migration, and integration failures with existing systems. A structured risk assessment should categorize risks by impact and probability, enabling organizations to prioritize mitigation strategies. Recommended approaches include conducting pilot migrations for critical applications, sandbox testing to validate configuration changes, and incremental migration to reduce exposure. Additionally, robust backup and rollback mechanisms should be established to ensure business continuity in the event of unforeseen issues.

Migration Strategy Selection
Choosing an appropriate migration strategy is critical for balancing speed, risk, and resource allocation. Lift-and-shift approaches involve moving applications with minimal changes, suitable for less complex environments but may miss opportunities for optimization. Incremental modernization focuses on refactoring and adapting applications in stages, reducing risk but requiring more extensive planning and development effort. Hybrid strategies combine elements of both, enabling phased migration while modernizing critical components simultaneously. The selection should consider factors such as application complexity, business priorities, downtime tolerance, and available technical expertise. Clear criteria for strategy selection help stakeholders align on objectives, allocate resources effectively, and ensure a structured migration roadmap.

IV. TECHNICAL MIGRATION STEPS

Application Refactoring and Code Changes
The technical execution of migrating from WebSphere to JBoss begins with thorough application refactoring. Many enterprise applications rely on WebSphere-specific APIs, EJB containers, and proprietary messaging constructs that are not natively supported in JBoss. Therefore, developers must analyze each module to identify code segments requiring modification, such as JMS queue implementations, EJB session beans, or security role mappings. Refactoring may include replacing proprietary APIs with standard Java EE equivalents or leveraging JBoss-supported libraries. Configuration files and deployment descriptors must also be adapted, translating WebSphere-specific settings into JBoss-compliant formats. For complex enterprise applications, automated code scanning tools can assist in identifying incompatible elements, while test-driven refactoring ensures that functional parity is maintained throughout the migration process.

Data Migration
Data migration is a critical component that demands meticulous planning to preserve integrity and consistency. Organizations must evaluate differences in database drivers, transaction handling, and data source configurations between WebSphere and JBoss. Migration strategies often involve exporting schemas, synchronizing transactional data, and validating referential integrity to prevent inconsistencies. For high-availability environments, it may be necessary to implement incremental or staged data migration techniques, allowing the new JBoss environment to operate in parallel with WebSphere temporarily. This approach minimizes downtime and enables comprehensive validation of data correctness before full production cutover.

Deployment and Configuration in JBoss
Once applications are refactored and data migration strategies are defined, deployment and configuration in the JBoss environment begin. This includes setting up server instances,

clustering, load balancing, and high-availability configurations. JBoss offers a modular deployment model that supports containerized and cloud-native architectures, providing opportunities for automation using scripts or orchestration tools. Key considerations include tuning thread pools, configuring connection pools for databases, and establishing logging and monitoring pipelines. Automated deployment scripts and CI/CD pipelines enhance repeatability and reduce human errors, ensuring that environments are consistent across development, testing, and production stages. Comprehensive testing in the JBoss environment ensures that the migrated applications function correctly under expected load conditions.

V. INTEGRATION AND INTEROPERABILITY

Service Integration

A critical aspect of migrating from WebSphere to JBoss is ensuring seamless integration with existing enterprise systems and services. WebSphere environments often host applications that interact with databases, messaging systems, web services, and legacy applications through proprietary connectors. During migration, these integration points must be carefully analyzed and re-implemented to ensure continuity. JBoss provides a rich set of integration capabilities, including support for standard Java EE connectors, JMS messaging, and REST/SOAP web services. For messaging, JMS queues and topics may need to be reconfigured using JBoss-native providers, while database connections must be redefined with JBoss-compatible data sources and connection pools. Additionally, ensuring proper orchestration of microservices or modular components in JBoss is essential, particularly in environments adopting hybrid or cloud-native deployment models. Thorough testing of all service endpoints and interdependencies ensures that applications maintain their functional integrity post-migration.

Security and Identity Management

Migrating security frameworks from WebSphere to JBoss is a pivotal step, as enterprise applications rely heavily on authentication, authorization, and role-based access controls. WebSphere typically integrates with enterprise LDAP directories, Active Directory, or proprietary security modules. In JBoss, these integrations must be replicated or reconfigured to maintain consistent identity management. Role mappings, group policies, and permissions need careful translation to JBoss security domains and JAAS configurations. Furthermore, JBoss supports modern security standards, enabling integration with OAuth, SAML, and single sign-on (SSO) solutions. Ensuring compliance with corporate security policies and regulatory frameworks requires comprehensive validation of authentication flows, secure data transmission, and access control rules. Implementing robust security configurations not only protects enterprise assets but also builds confidence among stakeholders in the reliability of the migrated environment.

Interoperability Testing

Beyond integration and security, comprehensive interoperability testing is crucial to verify that migrated applications interact correctly with dependent systems. This includes simulating cross-application transactions, validating messaging sequences, and confirming web service responses. Automated testing tools can facilitate regression testing, ensuring that functional equivalence is maintained and any discrepancies are detected early. Additionally, monitoring tools can be leveraged to observe real-time interactions, measure latency, and identify potential bottlenecks in the new JBoss environment.

VI. TESTING AND VALIDATION

Functional and Regression Testing

Comprehensive testing is a cornerstone of any successful migration from WebSphere to JBoss. Functional testing ensures that each application module behaves as expected in the new environment, validating workflows, business logic, and service interactions. Regression testing is equally critical, as changes introduced during migration—such as code refactoring, configuration adjustments, and dependency replacements—may inadvertently affect existing functionality. Automated testing frameworks, such as JUnit or Arquillian for Java EE applications, provide repeatable and scalable testing processes. Test suites should cover all critical use cases, from core transactional flows to edge scenarios, ensuring that migrated applications maintain parity with their WebSphere counterparts. Detailed documentation of test cases and results enhances traceability and provides a benchmark for post-migration validation.

Performance Testing

Beyond functional correctness, performance testing evaluates the scalability, responsiveness, and throughput of applications deployed on JBoss. WebSphere environments often have finely tuned performance parameters, and replicating similar efficiency in JBoss requires careful configuration of thread pools, connection pools, caching, and clustering. Load testing simulates expected production workloads to assess system behavior under stress, while stress testing identifies thresholds where performance degradation occurs. Benchmarking against WebSphere performance metrics helps identify optimization opportunities and ensures that the migrated environment meets or exceeds existing service level agreements (SLAs). Tools such as Apache JMeter, Gatling, or JBoss-specific performance monitors can facilitate accurate measurement and reporting.

User Acceptance and Operational Readiness

Successful migration is not only a technical achievement but also an operational milestone. User acceptance testing (UAT) validates the applications from the end-user perspective, ensuring that functionality, usability, and workflows meet business expectations. Engaging stakeholders during UAT helps identify gaps that may not surface during technical testing. Additionally, operational readiness assessment evaluates monitoring, logging, and alerting mechanisms, confirming that the production environment is prepared for day-to-day operations. Documentation, training, and knowledge transfer for administrators and support teams further enhance readiness, ensuring smooth transition and minimal disruption to business operations.

VII. DEPLOYMENT AND POST-MIGRATION CONSIDERATIONS

Continuous Integration and Continuous Deployment (CI/CD)

After successfully migrating applications from WebSphere to JBoss, establishing robust CI/CD pipelines is essential to

streamline ongoing development, testing, and deployment. JBoss's modular architecture and compatibility with modern DevOps tools, such as Jenkins, GitLab CI, or Red Hat OpenShift, facilitate automated builds, testing, and deployment. By integrating source control, automated unit tests, and deployment scripts, organizations can ensure consistent and repeatable releases across development, staging, and production environments. CI/CD pipelines also support rapid iteration, allowing teams to address defects, apply updates, or introduce new features without disrupting service continuity. This approach not only accelerates release cycles but also reduces human error and operational overhead, aligning the migrated JBoss environment with contemporary agile practices.

Monitoring and Troubleshooting

Monitoring the post-migration environment is critical to maintaining application performance, availability, and reliability. JBoss supports integration with both open-source and commercial monitoring tools, including Prometheus, Grafana, ELK Stack, and JBoss Operations Network (JON). These tools enable real-time visibility into resource utilization, transaction throughput, error rates, and cluster health. Proactive monitoring helps detect performance bottlenecks or configuration issues early, allowing administrators to implement corrective actions before user experience is impacted. Additionally, establishing structured logging and alerting mechanisms ensures that critical events are tracked and resolved promptly. Regular analysis of monitoring data also informs capacity planning, helping organizations optimize infrastructure and scale efficiently.

Maintenance and Future-Proofing

Ongoing maintenance is vital for sustaining the benefits of the migration. This includes patch management, version upgrades, and applying security updates promptly. JBoss's open-source model encourages community engagement, providing access to updates and enhancements that support long-term stability and innovation. Organizations should also implement structured backup and disaster recovery plans to protect against unexpected failures. Future-proofing involves planning for scalability, cloud integration, and adoption of containerized or microservices architectures, ensuring that the JBoss environment remains adaptable to evolving business needs. By combining operational diligence with forward-looking architectural strategies, enterprises can maximize ROI, reduce risk, and ensure that the migrated environment continues to support critical business operations effectively.

VIII. CASE STUDIES AND PRACTICAL EXAMPLES

Successful Migration Case Study

A leading financial services organization recently undertook a migration of its critical WebSphere-based applications to Red Hat JBoss to reduce licensing costs and modernize its middleware infrastructure. The migration involved over 50 enterprise applications, including core banking, customer relationship management, and reporting modules. The organization began with a comprehensive assessment to identify dependencies, proprietary APIs, and integration points with legacy systems. Pilot migrations were conducted for low-risk applications, allowing the team to refine migration scripts, configuration templates, and testing procedures. Application refactoring focused on replacing WebSphere-specific constructs with JBoss-compliant standards, while data migration was executed incrementally to minimize downtime. By leveraging JBoss's modular architecture and automated deployment pipelines, the enterprise achieved functional parity and enhanced scalability. Post-migration performance metrics indicated improved transaction throughput and reduced server overhead, validating the benefits of the migration strategy.

Lessons Learned and Best Practices

The case study highlights several critical lessons and best practices. First, comprehensive environment assessment and risk evaluation are essential to identify potential blockers and avoid costly surprises during migration. Second, adopting a phased migration approach mitigates risk by allowing incremental validation and adaptation, rather than attempting a full-scale cutover in a single step. Third, investment in automated testing and deployment pipelines ensures consistency, repeatability, and rapid issue resolution. Fourth, aligning security configurations and identity management early in the migration prevents operational disruptions and ensures compliance with enterprise policies. Additionally, engaging stakeholders throughout the process, including developers, system administrators, and business users, fosters collaboration and ensures that functional expectations are met. Finally, leveraging monitoring tools and performance benchmarking post-migration provides actionable insights, enabling optimization and fine-tuning for both immediate and long-term performance goals.

IX. CONCLUSION

The modernization of enterprise middleware from IBM WebSphere to Red Hat JBoss represents a strategic initiative that balances cost efficiency, operational agility, and future readiness. Throughout this review, the comparative analysis highlighted key architectural, licensing, and performance differences between the two platforms, emphasizing JBoss's modular, lightweight architecture and its suitability for cloud-native and containerized environments. Migration planning and assessment were shown to be critical steps, requiring comprehensive inventorying of applications, careful risk evaluation, and the selection of an appropriate migration strategy tailored to organizational needs. Technical migration steps, including application refactoring, data migration, and deployment configuration, were identified as essential processes to ensure functional parity and maintain system reliability. Integration and interoperability considerations underscore the importance of seamless service connectivity, robust identity management, and security compliance, all of which are necessary to maintain enterprise operational continuity. Rigorous testing and validation, encompassing functional, performance, and user acceptance testing, further ensure that migrated applications perform reliably under real-world conditions. Post-migration deployment practices, including CI/CD pipelines, proactive monitoring, and structured maintenance strategies, enable organizations to achieve operational stability while laying the groundwork for continuous improvement and scalability. Case studies reinforce the practical benefits of a structured migration approach, demonstrating that phased execution, automated tooling, and stakeholder engagement significantly mitigate risk and enhance overall success. Lessons learned from these real-world

examples provide actionable insights for enterprises contemplating similar modernization initiatives, highlighting best practices in planning, execution, and post-migration optimization. In conclusion, migrating from WebSphere to JBoss is not merely a technical exercise but a transformative strategy that enables organizations to reduce costs, enhance system flexibility, and embrace modern IT practices. By combining careful planning, methodical execution, and a focus on operational resilience, enterprises can successfully modernize their middleware stack, ensuring that their applications remain agile, secure, and scalable in the evolving landscape of enterprise IT. This modernization journey ultimately positions organizations to capitalize on emerging technologies, adopt cloud-native strategies, and maintain a competitive edge in a rapidly changing digital environment.

REFERENCE

1. Battula, V. (2015). Next-generation LAMP stack governance: Embedding predictive analytics and automated configuration into enterprise Unix/Linux architectures. International Journal of Research and Analytical Reviews, 2(3).
2. Battula, V. (2016). Adaptive hybrid infrastructures: Cross-platform automation and governance across virtual and bare metal Unix/Linux systems using modern toolchains. International Journal of Trend in Scientific Research and Development, 1(1).
3. Battula, V. (2017). Unified Unix/Linux operations: Automating governance with Satellite, Kickstart, and Jumpstart across enterprise infrastructures. International Journal of Creative Research Thoughts, 5(1). Retrieved from http://www.ijcrt.org
4. Battula, V. (2018). Securing and automating Red Hat, Solaris, and AIX: Provisioning-to-performance frameworks with LDAP/AD integration. International Journal of Current Science, 8(1). Retrieved from http://www.ijcspub.org
5. Cunico, H.A., Fernández, L.A., Hellsten, C., & Kharkovski, R. (2005). Migrating applications from weblogic, jboss and tomcat to websphere v6.
6. Edgeworth, B., Prall, D., Barozet, J.M., Lockhart, A., & Ben-Dvora, N. (2016). Cisco Intelligent WAN (IWAN).
7. Gill, P.E., Wong, E., Murray, W., & Saunders, M.A. (2016). User's Guide for SQOPT Version 7.5: Software for Large-Scale Linear and Quadratic Programming.
8. Gowda, H. G. (2017). Container intelligence at scale: Harmonizing Kubernetes, Helm, and OpenShift for enterprise resilience. International Journal of Scientific Research & Engineering Trends, 2(4), 1–6.
9. Kota, A. K. (2017). Cross-platform BI migrations: Strategies for seamlessly transitioning dashboards between Qlik, Tableau, and Power BI. International Journal of Scientific Development and Research, 3(?). Retrieved from http://www.ijsdr.org
10. Kota, A. K. (2018). Dimensional modeling reimagined: Enhancing performance and security with section access in enterprise BI environments. International Journal of Science, Engineering and Technology, 6(2).
11. Kota, A. K. (2018). Unifying MDM and data warehousing: Governance-driven architectures for trustworthy analytics across BI platforms. International Journal of Creative Research Thoughts, 6(?). Retrieved from http://www.ijcrt.org
12. Madamanchi, S. R. (2015). Adaptive Unix ecosystems: Integrating AI-driven security and automation for next-generation hybrid infrastructures. International Journal of Science, Engineering and Technology, 3(2).
13. Madamanchi, S. R. (2017). From compliance to cognition: Reimagining enterprise governance with AI-augmented Linux and Solaris frameworks. International Journal of Scientific Research & Engineering Trends, 3(3).
14. Madamanchi, S. R. (2018). Intelligent enterprise server operations: Leveraging Python, Perl, and shell automation across Sun Fire, HP Integrity, and IBM pSeries platforms. International Journal of Trend in Research and Development, 5(6).
15. Maddineni, S. K. (2016). Aligning data and decisions through secure Workday integrations with EIB Cloud Connect and WD Studio. Journal of Emerging Technologies and Innovative Research, 3(9), 610–617. Retrieved from http://www.jetir.org
16. Maddineni, S. K. (2017). Comparative analysis of compensation review deployments across different industries using Workday. International Journal of Trend in Scientific Research and Development, 2(1), 1900–1904.
17. Maddineni, S. K. (2017). Dynamic accrual management in Workday: Leveraging calculated fields and eligibility rules for precision leave planning. International Journal of Current Science, 7(1), 50–55. Retrieved from http://www.ijcspub.org
18. Maddineni, S. K. (2017). From transactions to intelligence by unlocking advanced reporting and security capabilities across Workday platforms. TIJER – International Research Journal, 4(12), a9–a16. Retrieved from http://www.tijer.org
19. Maddineni, S. K. (2017). Implementing Workday for contractual workforces: A case study on letter generation and experience letters. International Journal of Trend in Scientific Research and Development, 1(6), 1477–1480.
20. Maddineni, S. K. (2018). Automated change detection and resolution in payroll integrations using Workday Studio. International Journal of Trend in Research and Development, 5(2), 778–780.
21. Maddineni, S. K. (2018). Governance driven payroll transformation by embedding PECI and PI into resilient Workday delivery frameworks. International Journal of Scientific Development and Research, 3(9), 236–243. Retrieved from http://www.ijsdr.org
22. Maddineni, S. K. (2018). Multi-format file handling in Workday: Strategies to manage CSV, XML, JSON, and EDI-based integrations. International Journal of Science, Engineering and Technology, 6(2).
23. Maddineni, S. K. (2018). XSLT and document transformation in Workday integrations: Patterns for accurate outbound data transmission. International Journal of Science, Engineering and Technology, 6(2).
24. Marchioni, F. (2009). JBoss AS 5 Development.
25. Marrs, T., & Davis, S. (2005). Jboss at Work: A Practical Guide.
26. Matthews, J.N., Dow, E.M., Deshane, T., Hu, W., Bongio, J., Wilbur, P.F., & Johnson, B. (2008). Running Xen: A Hands-On Guide to the Art of Virtualization.
27. Mulpuri, R. (2016). Conversational enterprises: LLM-augmented Salesforce for dynamic decisioning. International Journal of Scientific Research & Engineering Trends, 2(1).

28. Mulpuri, R. (2017). Sustainable Salesforce CRM: Embedding ESG metrics into automation loops to enable carbon-aware, responsible, and agile business practices. International Journal of Trend in Research and Development, 4(6). Retrieved from http://www.ijtrd.com
29. Mulpuri, R. (2018). Federated Salesforce ecosystems across poly cloud CRM architectures: Enabling enterprise agility, scalability, and seamless digital transformation. International Journal of Scientific Development and Research, 3(6). Retrieved from http://www.ijsdr.org
30. Rao, R.A. (2009). JBoss Portal Server Development.
31. Snyder, B.A., Bosanac, D., & Davies, R. (2011). ActiveMQ in Action.
32. Takemura, C., & Crawford, L.S. (2009). The Book of Xen: A Practical Guide for the System Administrator.
33. Tochel, C., Haig, A., Hesketh, A., Cadzow, A., Beggs, K., Colthart, I., & Peacock, H. (2009). The effectiveness of portfolios for post-graduate assessment and education: BEME Guide No 12. Medical Teacher, 31, 299 - 318.