NGE Solutions

Building the Next Generation Enterprises

PISA

(Planning, Integration, Security and Administration)

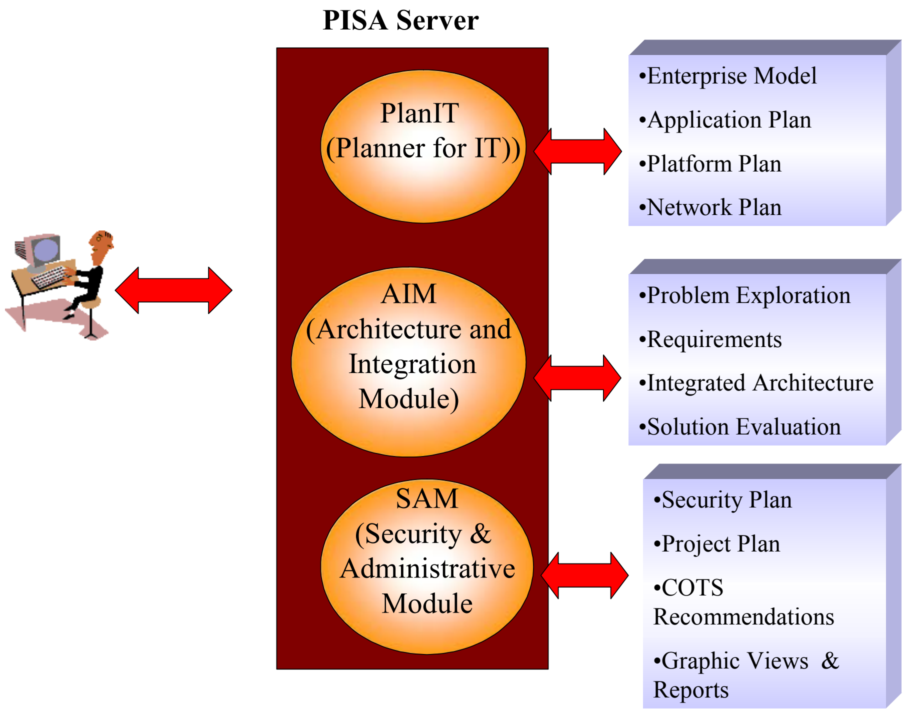
An Intelligent Decision Support Environment for IT Managers and Planners

SOA Planning Through PISA-AIM

Working Draft: November 20, 2007

NGE Solutions, Inc. (www.ngesolutions.com)

PISA at a Glance



- PISA is an intelligent decision support environment that helps IT managers and planners to plan, integrate and secure their IT (information technology) systems.
- PISA consists of the following modules that specialize in different IT planning tasks:
  - PlanIT for IT Infrastructure planning (application planning, platform planning, network planning)
  - AIM for SOA planning (development and evaluation of SOA-based integrated architectures)
  - SAM for security planning (based on attack trees and security patterns) and project planning with staffing estimates and WBS (work breakdown structure) that can be ported to MS Project
- The focus of this document is on AIM

IMPORTANT NOTE TO THE READERS

This document is a consolidation and expansion of the numerous short tutorials and explanations that are dispersed over different pages of the PISA-AIM online system. We received several suggestions from our users who indicated that they find it very tiring to read many short tutorials/explains online and suggested that one consolidated guide before working on AIM would be very useful. Hence this document.

After you have gone through this document, you can skip most of the AIM tutorials and explains and refer to them only to refresh your memory.

Table of Contents

Executive Summary

Modern enterprises continually architect their information systems (IS) for flexibility and integrate internal and external systems to survive and thrive in this rapidly fluctuating business landscape. The business motivations for architecture and integration projects are frequent mergers and acquisitions, outsourcing, business process re-engineering and streamlining of operations for improved services. Service Oriented Architectures (SOAs) provide a conceptual framework, supported by standards and enabling technologies, to help organizations develop integrated and flexible architectures. However, SOA projects are complex undertakings especially due to the evolving products, technologies, platforms, and specifications.

The objective of this document is to show how AIM (Architecture and Integration Module), part of the overall PISA Environment, can be used to develop and document thorough SOA plans quickly and repeatedly. The document shows how AIM can guide the IT managers, planners, and architects through the maze of business scenarios, strategic choices, technical interdependencies, and integration tradeoffs based on cost, performance and security issues in SOA projects. Examples of the questions that can be answered by using AIM are:

- What is the cost of transitioning to SOA and how can this cost be justified in business terms
- What are the security, performance, governance, and QoS impacts of transitioning to SOA
- Given a future business scenario, what type of SOA plan will be needed in terms of the services, configurations, platforms and commercial-off-the-shelf (COTS) products to be used
- How can the various business and technical scenarios be modeled and evaluated quickly to gain insights before massive deployments
- How can the multiple players (managers, users, business partners, contractors, developers) participate in SOA projects by understanding and influencing the decisions being made

The document starts with a quick overview of architectures, Service Oriented Architectures and Application Integration in Section 1 (this section can be skipped by people already familiar with these concepts). Section 2 gives an overview of PISA- AIM (henceforth referred to as AIM) and Section 3 explains the AIM methodology used to develop complete SOA plans. Section 4 concludes this document through a detailed case study that shows how an application can be transitioned to SOA by using the various AIM advisors.

1. Architectures, Service Oriented Architectures and Application Integration -- A Quick Overview

NOTE: Please skip this section if you are already familiar with the concepts of architectures, integration and SOA.

1.1. What is an Architecture?

Definition: An architecture of a system is a structure that describes three things:

- Components of the system (what are the pieces of a system?),
- Functions performed by the components (what do they do?), and
- Interfaces/interactions between the components (how do they work with each other?).

This definition is consistent with the IEEE 610.2 definition of an architecture: "The structure of the components, their properties, relationships, and the principles and guidelines governing their design and evolution over time."

Within the context of information systems, several types of architectures have emerged over the years (e.g., business architectures, database architectures, computing architectures, network architectures, software architectures). In all of these cases, it is useful to remember what are the components of the system, what they do, and how they interface/interact with each other. For example, a business architecture would show the business components (e.g., the business processes), what they do (e.g., satisfy customer needs) and how they interface/interact with each other (e.g., a business process flow). Our interest is in *solution architectures* that combine several applications and infrastructure components into a working solution to satisfy customer needs at enterprise level.

1.2. What is Application Integration?

In general, integration puts parts together into a whole *somehow*. Thus systems are integrated when they are to a lesser or greater extent seamlessly combined to support similar conventions or styles. At an enterprise level, enterprise application integration (EAI) software acts as a central command centre for coordination between corporate applications.

For our purposes, integration refers to the ease with which systems can be used – we primarily concentrate on the user benefits. *Integrated systems basically minimize the effort needed to the user of the system – the user may be a human being or another automated system.* This implies that two systems, S1 and S2, are integrated if they:

- Share and exchange information without external intervention,
- Are seamless in terms of operations, and
- Show consistency of behaviour and presentation.

A well known example of integrated applications is the Microsoft Office Suite that combines word processing (MS Word), presentation (MS PowerPoint), and spreadsheets (MS Excel) into a single package. The package allows information sharing/exchange with minimal intervention, is seamless in terms of operations, and shows consistency of behaviour and presentation. Basically, the MS Office Suite minimizes the human effort in developing documents, presentations, and spreadsheets.

Another example of integration are the “unified messaging” systems that seamlessly combine voice mail, fax, and email into a single package. In unified messaging, a voice mail system S1 and an email system S2 are integrated because the voice mail from S1 can be stored as email for S2 (and vice versa) easily, and S1 and S2 provide the same "look and feel" (same type of commands, icons, and screen formats). In the same vein, a legacy application S1 is integrated with a new application S2 if S2 can exchange information with S1 seamlessly and if S1 has the same "look and feel" as S2.

Let us extend this discussion to **enterprise application integration (EAI)**. EAI refers to the process of connecting different applications to allow information to flow between functions within an enterprise. The goal is to minimize the effort by the users of enterprise applications – the corporate personnel. Going beyond enterprises, the flow may include information flows between trading partners to minimize the effort needed for B2B trade. For example, EAI in support of an online order processing system should include the needed connections so that the order can be placed online, the availability of stock in inventory can be verified, the customer's credit can be checked, and the amount of the purchase can be approved. *EAI platforms* provide the middleware services that transform and route the data throughout an organization so that the individual applications can properly access the information they need. EAI platforms, also known as eAI (e-business application integration) platforms, provide the “bus” for different application to interact with one another in a meaningful way.

1.3. A Service Oriented View of Business and Service Oriented Architectures (SOA)

All businesses provide a set of services. Some services are provided to the customers (B2C), some to other businesses (B2B) and some to the employees (B2E). For example, Figure 1 shows a very high level view of a retail store that provides marketing, sales, customer support, and many other services (some are customer facing, some are supplier facing, and some are employee and management facing. In the highly fluid business environment of today, some of these services are provided by other service providers (outsourcing agencies, business partners, etc). For example, in this organization, customer services, marketing, human resource (HR) management, and finance and accounting (F&A) services are provided by other service providers (SPs). The task of the enterprise management is to find the best service providers (SPs) to run the firm. In addition, a company can change its business by adding new services from new SPs. For example, wired telephone company can add a wireless service provider, a manufacturing company can add a retail outlet provider, etc. In addition “service bundles” can be created by different SPs to meet user needs and to compete for user business. The idea is that companies may add, delete, change and merge SPs that provide the best services to compete.

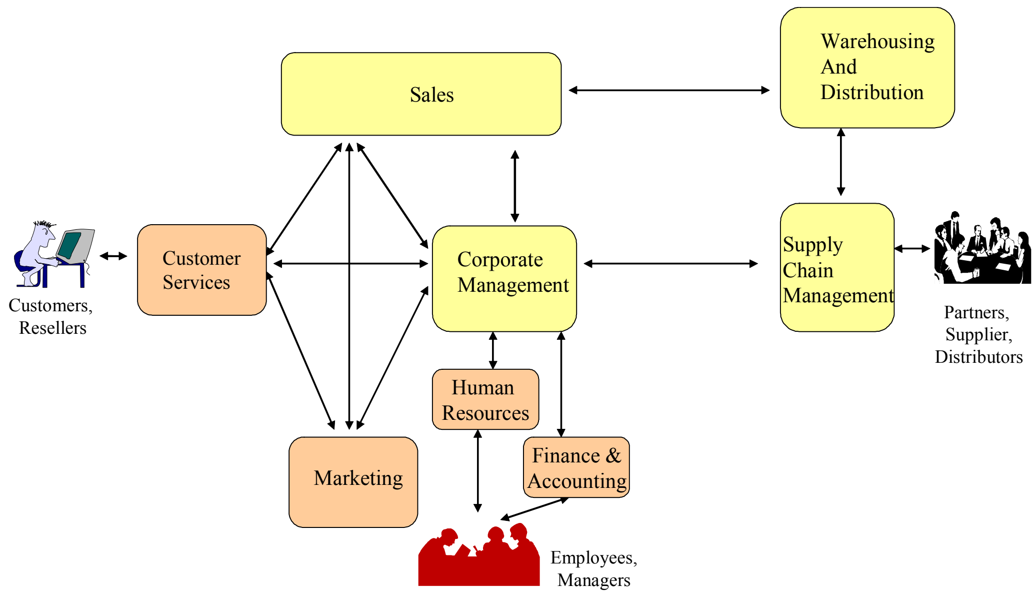


Figure 1: Service Oriented View of a Retail Store (Darker Blocks mean Outsourced/Rented Services)

How can enterprise software support this service-oriented business climate? The answer is that business software is developed as business components that can be assembled with other business components to provide business services. For example, a large grained business component (BC) -- a software package from PeopleSoft -- could provide the HR business service (BS). Similarly another BC from SAP could support the marketing BS and the like. A company could choose, assemble and run these BCs from different suppliers to support its BSs. A company could also replace a BC from PeopleSoft with a BC from SAP to provide better services, for example. More interestingly, an order processing BC residing in Atlanta could check the inventory managed by a BC in Detroit or Singapore. This implies the following:

- There is a BC that provides a set of business services -- this is the service provider
- The services are well defined so that other BCs can understand them
- BCs have well defined interfaces so that they can work with each other
- BCs from different suppliers can be used to provide a business service
- An IT infrastructure (middleware service) exists that allows services provided by components to be advertised, discovered, selected, and invoked over the Internet.

1.4. Benefits and Challenges of Service-Oriented Architecture

SOA promises to speed development and decrease integration time and effort but it poses several challenges that need to be addressed. The main benefits of adopting an SOA are:

- SOA makes it easier to integrate the IT environments found in most companies. SOA works very well in heterogeneous environments because developers don't have to spend an inordinate amount of time writing new lines of code to connect applications across an enterprise.
- SOA leverages the investment in existing systems – you can identify the capabilities of existing systems and leverage them by gradually moving to SOA and maximize the value of IT investments while minimizing the risk.
- SOA can help companies improve their ability to adapt to changing business requirements and shifting market conditions.

- Building services by using simple object access protocol (SOAP) and Web services description language (WSDL) facilitates the internal integration process and lets customers and business partners share information more easily across company firewalls.
- Businesspeople can think about the best ways to run their business, i.e., what type of business services and business components are needed to accommodate the customers and improve the level of customer service. By exposing and sharing information across multiple applications, companies can extract more business performance data in real-time, improving business intelligence.
- The benefits of easier integration and increased agility lead to greater ROI. Some companies have achieved a 200 percent return on their SOA investment. One of AXA Financial's most popular SOA-based services is Get Client, in which any front-end app can issue a command and, after probing around the legacy systems, come back with a complete picture of a customer's investments.

The main challenges presented by SOA are:

- Security is a big challenge because it is easier to secure a closed system than an open architecture. There is a lack of security standards for SOA and Web services. It may be important to transition first on business processes that do not require a high level of security.
- Management of the complexity of a services configuration is difficult and requires a good SOA governance model. For example, if 100 people are using a certain service through a WSDL interface, how do you communicate with those users if someone decides to change the interface?
- Network monitoring is another issue. Orchestration of complex interacting business processes in a service-oriented architecture creates complex monitoring and auditing requirements. For instance, when a transaction goes awry on a service-oriented network, which could involve multiple service providers, finding out what went wrong or where the transaction dropped or whether someone put incorrect information in the network can be a challenge. The current Web services technical standards are only beginning to address the service-oriented distributed collaboration, process orchestration and monitoring goals a practical reality.
- Cost can be higher because building an SOA is not cheap. In particular, reengineering of existing systems architecture to transition to SOA is expensive. It also requires significant human capital, including business analysts to lay out the business processes, systems architects to turn processes into specifications, software engineers to develop the new code and project managers to track it all.
- It is important but difficult to identify the right level of service to provide. The services should not have too fine a granularity, they should be a high business-process level because too narrow a focus creates a need for too many services which increases development time and may flood a network.
- The loosely coupled architecture of SOA is good for systems that do not require near-real-time responses. For example, an SOA-based air-traffic control system may not be a great idea.

1.5. Service Oriented Architecture (SOA)– A Closer Look

Service-oriented architectures (SOAs) rely on services and the components that provide the services as the fundamental elements for developing applications. The main idea of service oriented architectures is that the applications should be thought of in terms of the services they provide and the individual components that actually deliver the services. The services can be combined into aggregate services and similar components can be combined into applications (see the sidebar “What is a Service Oriented Architecture (SOA)?”). Thus a bank, for example, provides a set of services (e.g., deposits, withdrawals, fund transfers) and these services are provided through components that can be combined into banking applications. A service-oriented architecture has the following characteristics:

- **Applications must be developed as a set of services.** Applications must be decomposed into a number of smaller individual services that are easier to create and easier to maintain. Individual services should be supported through components.

- **Services and components must be as general purpose as possible.** It is important to decompose applications into components in such a fashion so that as many components as possible are general purpose and as few as possible are special purpose. It is best to create common services and components that can be used to serve many different requests.
- **Services must have well-defined interfaces.** The interfaces must be stored in a directory for discovery. Application clients must be able to query an interface directory or a server for a service and to ask for the current server functions. It is best to use an XML based service definition and use XML to exchange data.
- **Applications and their components must use Internet communication.** Application clients and servers, and any other components, must not communicate via any proprietary protocol. Clients must request services via a standard Internet protocol such as HTTP and service providers must respond via the same protocol. This should be carried to the component level, i.e., each component of an application should be Internet enabled and communicate with others over the Internet.
- **Applications must be able to interwork with other Applications**. Applications, in the form of business components, should be able to cooperate with other applications to form larger "enterprise applications".

Additional characteristics of SOA are:

- **Applications must be scaleable and portable**. Applications must be able to scale from supporting a few to thousand requests per day. Server services must also be portable from one platform to another without major re-engineering. It is important to use a IRL or a directory service to locate
- **Thin client model should be favoured.** Clients should use standard Internet browsers (e.g., Internet Explorer or Netscape Navigator) and adhere to a thin client model as much as possible. With thin client model, no application components are stored on client computers. This has the advantage that most business logic is server-based and thus can be easily managed, upgraded, and controlled instead of hundreds of different software components that reside on different client machines.
- **Service Providers should use standard Internet servers**. Service providers must use standard Internet servers running standard (or at least common) software such as Apache and Microsoft IIS. The server-side software must use standard and commonly used technologies such as JSP/ASP. This will help in the portability of services.

Web Services are becoming the key enablers of SOA.

1.6. SOA Architecture Patterns and Enterprise Service Bus

An SOA architecture pattern or just an SOA pattern defines the infrastructure services needed by the applications in SOA. These infrastructure services are typically defined in terms of an Enterprise Service Bus (ESB) that provides the main mechanism for integrating the internal applications. An SOA ESB provides a collection of technologies (middleware such as Web Services, adapters/gateways for protocol conversion, data transformers, transaction managers, and work/process flow systems) that allow diverse applications to talk to each other. At their best, ESB platforms hide all the complexity needed to enable interactions between applications that were developed at different times by using different middleware technologies. Thus ESB platform is not a new technology – rather, it is a combination of “well-known” technologies that can integrate multiple applications. All applications (business components) provide services that are invoked through well defined interfaces. Figure 2 shows a conceptual view of an ESB.

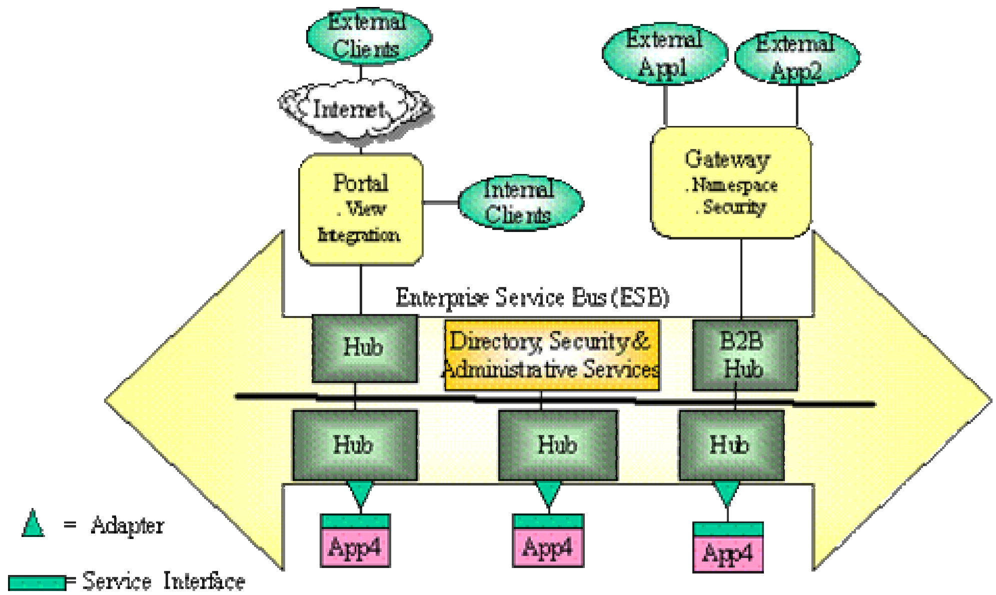


Figure 2: Enterprise Service Bus (ESB) – Conceptual View

While there is no industry-standard definition, an ESB is expected to posses the following common set of characteristics:

- Communication through a Broker. An ESB uses a software intermediary (a hub) between the sender and the receiver, providing a brokered communication between them.
- Intelligent Routing through Directory Services. ESBs typically use a directory service to resolve addresses at run time and may also route messages based on predefined rules (e.g., find a closest service provider). A Hub provides communications services between various service providers and consumers An ESB may consist of one or more hubs.
- Endpoint metadata. ESBs typically keep metadata about service interfaces and message schemas. This information is used to translate messages. .
- Message transformation. ESBs typically provide off-the-shelf adapters that are used for message and protocol translations.
- Basic Web services support. Most ESBs support basic Web Services standards including SOAP, WSDL, and XML. UDDI support for directory services is also becoming popular. stery service as well as foundational standards such as TCP/IP and XML.
- Some ESB vendors offer additional features including security, administration, software development, validation, logging, and auditing capabilities.

In short, ESB platforms are sophisticated mediators that provide an "application bus" for rapid and flexible integration of a very wide range of applications that may span technology vintages (past, present and future) as well as organizational boundaries (inter and intra organizational boundaries). ESB platforms are an outgrowth of earlier mediators such as application gateways and object wrappers and are intended to insulate the business from changes in the applications and business needs and help with combining systems from acquired companies ESB platforms may use different types of middleware technologies (e.g., CORBA, Message Queuing, etc). However Web Services are the most recent technologies of ESBs. ESBs may also exist as EAI (enterprise application integration) platforms or message brokers.

ESB software is commercially available from vendors such as IBM (Websphere ESB), Microsoft (Biztalk Server 2006) and others (e.g., Sonic Software, Systinet, Tibco, Fiorano, IONA).

1.7. Sources of additional information (on the Web):

- IBM SOA Website: www.ibm.com/soa
- Sun SOA Website: www.sun.com/soa
- IEEE Computer Society Technical Committee on Services Computing - www.servicescomputing.org. This is a very good website on SOA information..
- SOA Portal at http://www.service-architecture.com/

2. SOA PLANNING THROUGH PISA-AIM

AIM (Architecture and Integration Module), part of the overall PISA Environment, is designed to help the users develop detailed and well documented SOA plans for a wide range of real life scenarios. AIM consists of a family of advisors that systematically guide the IT managers, planners, and architects through a series of steps that help the users understand the complexity of the problem. Based on the complexity and the nature of the problem, AIM then produces reports that provide the following pieces of information for a solid business case and a blueprint:

- The cost of transitioning to SOA
- The security, performance, governance, and QoS impacts of transitioning to SOA
- Detailed SOA plan that shows the services, configurations, platforms and commercial-off-the-shelf (COTS) products needed for a given business scenario.
- Detailed requirements and architecture reports that can be used in implementing the SOA plan

2.1. AIM as Part of PISA

PISA (Planning, Integration, Security & Administration) Environment (Figure 4) is a research prototype for IT planners, architects and managers that attempts yo satisfy the aforementioned requirements. PISA consists of a set of automated consultants ("advisors") – each advisor supports a specific stage and collaborates with other advisors to produce plans. PISA advisors are segmented into three modules (see Figure 5 for an overview of PISA capabilities):

- **PlanIT (Planner for IT)** concentrates on IT planning projects and develops a plan at the enterprise level.
- **SAM (Security and Administration Module)** provides security and administrative services to the entire PISA system.
- **Architecture and Integration Module (AIM)** deals with the more detailed issues of how specific components of the plan can be architected and integrated to form a functioning system.

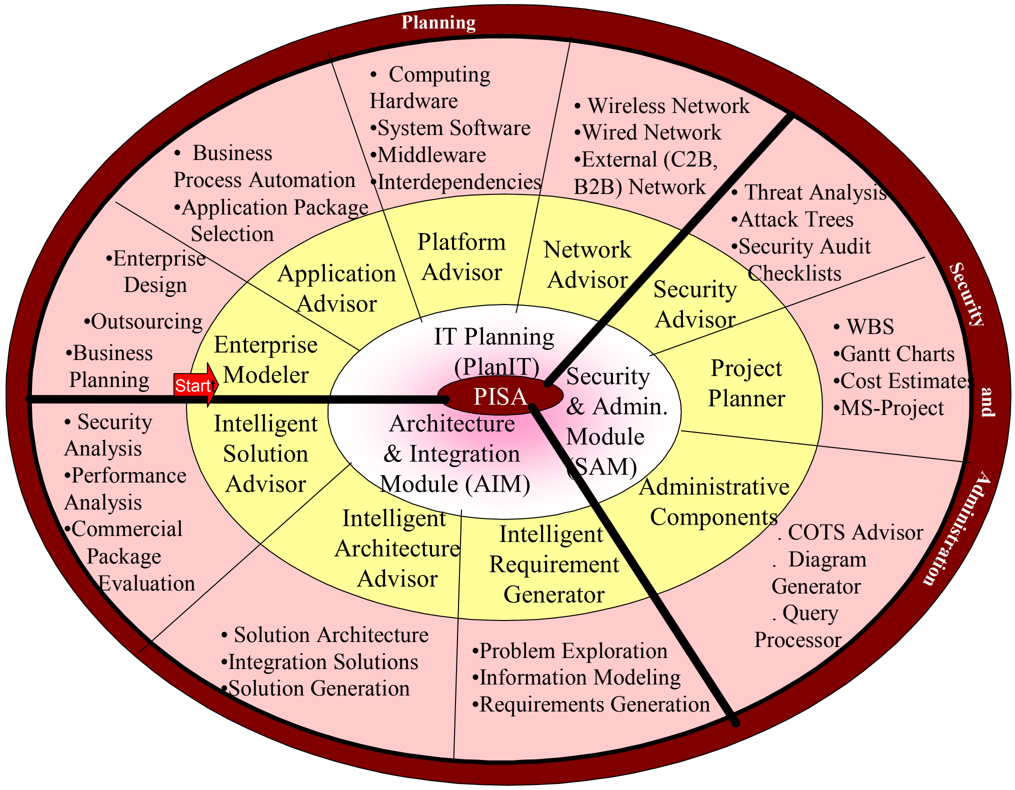


Figure 3: Overview of PISA Capabilities

The PISA environment uses a systematic methodology that spans three major phases (see Figure 4):

- IT Planning Phase (handled by PlanIT): develop a solution sketch at enterprise level, with rough estimates and separation of issues. This sketch is refined in the next phase.
- Architecture and Integration Phase (handled by AIM): concentrate on more detailed issues of how specific components of the plan will fit together to form an SOA-based functioning system. AIM guides the user through these difficult but crucial issues.
- Security and Administration Phase (handled by SAM): provides security and administrative services (e.g., project planning) to the entire PISA system.

All phases are essential, albeit with different emphasis. For example, if a business wanted to offer new services for a new business unit, then PlanIT would be used to identify the applications, platforms and networks needed to support these services, and SAM would be used to secure the needed IT assets and to develop a project plan. AIM, the focus of this document, would be use to determine how the new services will integrate with the existing ones.

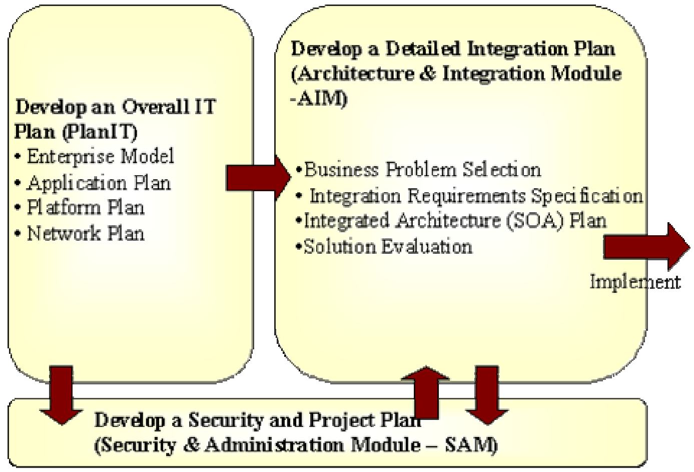


Figure 4: High Level View of IT Planning, Integration, Security & Administration (PISA) Methodology

2.2. AIM -- A Closer Look

AIM (Architecture and Integration Module) deals with the architecture and integration issues based on SOA. AIM consists of the following main advisors, shown in Figure 5, that help a user through life cycle of an SOA project:

- **Business Problem Explorer (BPE)** helps the user to select and define an integration project in terms of participating applications. For large scale enterprise integration projects, the user goes through this process iteratively.
- **Intelligent Requirements Generator (IRG)** helps the user to quickly generate requirements documents that capture the essence of the integration problem for the selected business area.
- **Integrated Architecture Advisor (IAA)** attempts to capture the complexity of the problem and suggests an SOA architecture based on the requirements. This advisor walks the user through strategic decisions and scenarios of outsourcing, migrations, and data warehousing. This advisor also maps the technical architecture produced by IAA to COTS (commercial-off-the-shelf) solutions and captures the main decisions in an architecture document.
- **Intelligent Solution Advisor (ISA)** guides the user through the process of cost, performance and security estimates and produces cost-benefit analysis of the integration project. The user can now go back and revaluate the same problem for different architectural configurations or pick another business problem by going back to the BPE.

The starting point of AIM is the IT plan that is generated by working with the planning advisors (Application Advisor, Platform Advisor, Network Advisor, etc.).

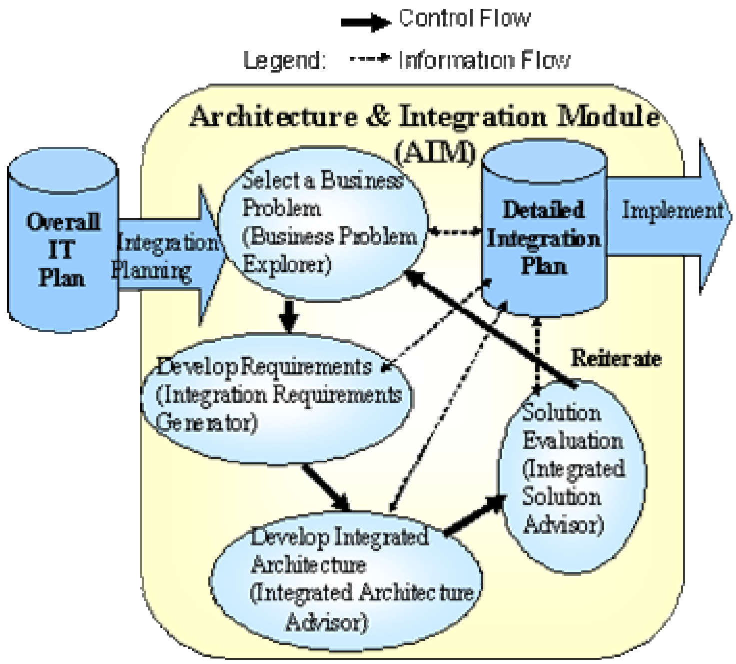


Figure 5: Conceptual View of AIM

2.3. Who should use AIM and why?

AIM is intended for IT managers, IT planners, and IT architects to work through the following business scenarios:

- Determine how a given application (called ***Target Application***) can interwork with its surrounding applications by using SOA. The target application may be a new application, an existing application that is being extended-migrated, or a critical app that needs to (re)evaluated.
- Estimate the cost of transitioning to SOA and how can this cost be justified in business terms.
- Develop and evaluate an SOA plan in terms of cost, performance and security.
- Transition an application or a group of apps to SOA through replacement or gradual migration strategies.
- Outsource an application through rental, remote hosting, or other outsourcing scenarios and evaluating the tradeoffs.
- Develop a data warehouse for business intelligence and understand how it fits into the overall enterprise architecture.
- Do gap analysis between your current method of operation (PMO) and future method of operation (FMO) based on SOA. In other words, given a future business scenario, what type of SOA services, configurations and platform should be used in terms of outsourcing, renting, remote delivery of services, and extended enterprises scenarios.

3. AIM Methodology

Development of an integration plan is a complicated task with many challenges. Instead of a 'big bang' approach where all enterprise systems are converted to SOA in an afternoon, AIM supports a gradual approach where the enterprise achieves an integrated architecture one business (application) area at a time. The AIM methodology, discussed later, guides the user through the iterative process of choosing a business problem and then developing and evaluating integrated architectures for the chosen problem.

Integration projects can be large scale enterprise-wide undertakings that may involve numerous applications. The methodology displayed in Figure 6 allows the users to break large scale integration projects into smaller pieces that can be understood, integrated and then composed into enterprise wide solutions.

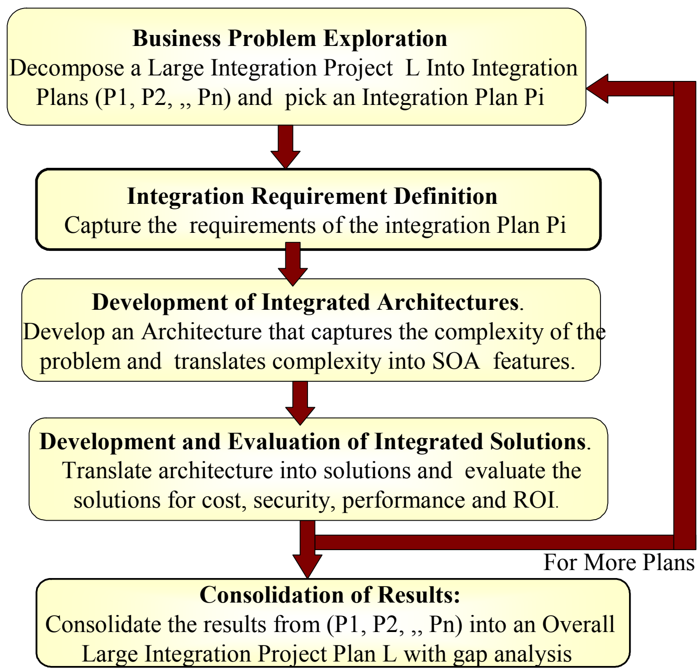


Figure 6: AIM Methodology

The main steps of this methodology are:

- **Business Problem Exploration: D**efine a Large Integration Project L that covers the overall SOA project and decompose it into Integration Plans (P1, P2,,, Pn), i.e., L= (P1, P2,,, Pn). For a small project, L = (P1). The integration plan may be defined at a Business Process (lower granularity) or Business Function (large granularity) level. For small projects involving a few applications, it is a good idea to stay at low granularity. For enterprise-wide integration projects, large granularities are better. The output is an integration project that identifies critical applications and a decomposition of the plan. **Business Problem Explorer (BPE)** supports this stage by helping the user to select and define an integration project Pi in terms of participating applications. For large scale enterprise integration projects, the user goes through this process iteratively.
- **Integration Requirement Definition.** Use the PISA Application Repository (AR) to define the integration requirements of the selected 'Target Applications'. At the core of each integration plan is a critical (target) application that is important to the business. This approach is based on the well known Critical Success Factors (CSF) methodology (Rockart, J.F. "Chief executives define their own data needs", Harvard Business Review", Vol. 57, pp. 81-93, 1979). The CSF methodology

concentrates on a core set of critical issues and addresses them instead of analyzing every thing in detail. By using CSF, a user can concentrate on the apps that are critical to the success of the business and understand how their transition to SOA will impact the surrounding applications. The PISA Application Repository shows interactions between various business processes (BPs), business functions (BFs) and applications (automated BPs and BFs) and thus helps in determining critical applications. The interacting applications for each integration plan define an "Application Group" (a group of applications that will participate in an integration project). For example, if an order processing (OP) application is to be integrated in an integration plan, then the application group consists of OP plus interacting applications such as inventory, payment and shipping. **Intelligent Requirements Generator (IRG)** supports this stage by helping the user to quickly generate requirements documents that capture the essence of the integration problem for the selected business area.

- **Development of Integrated Architectures**. For each chosen integration plan, develop an integrated architecture based on SOA principles. The main objective is to capture the complexity of the problem and translate the complexity into SOA features. The process used in this stage is: a) develop a logical architecture to capture the basic complexity, b) select integration strategies (e.g., migration versus integration-in-place) for specialized considerations, and c) construct a physical service oriented architecture (SOA) based on SOA patterns that captures the key features. The output is a detailed SOA architecture that highlights the key features needed. **Integrated Architecture Advisor (IAA)** helps the user through various steps of this stage.
- **Development and Evaluation of Integrated Solutions.** The objective of this stage is to translate architecture into solutions and evaluate the solutions based on metrics (e.g., costs, security, performance and return on investment -- ROI). The process used in this step is: a) translate the selected architecture *A* into plausible solutions (*S1, S2,,Sn*) by using different product mappings in terms of COTS (commercial off-the-shelf product), and b) evaluate the solutions (*S1, S2,,Sn*) in terms of metrics such as cost, security and performance. Cost estimates due to the chosen architecture are based on the complexity of the ESB selected, the type and number of adapters needed, platforms to be bought/used, commercial-off-the-shelf (COTS) packages to be used, etc. Security implications are based on security patterns chosen in the architecture and performance implications are based on the configurations and allocations. **Intelligent Solution Advisor (ISA)** supports this stage by guiding the user through the process of cost, performance and security estimates and producing ROI (return on investment) analysis of the integration project.
- **Consolidation of Results**. After evaluating the solution by ISA, the user can go back and re-evaluate the same problem for different architectural configurations or pick another integration plan by going back to the BPE. After reiterating through the individual integration plans (P1, P2,,, Pn) of the large integration project L, now consolidate the results into an overall project document (Grand Consolidated Report). The objective of this stage is to re-iterate, consolidate the results from different projects and do gap analysis. The consolidation effort may be zero for small single application projects but may be considerable for enterprise-wide application integration projects. The process used in this stage consists of several steps: a) for large scale projects, re-iterate to pick another critical application and go through previous stages, b) consolidate results at the end of iterations to develop a business case, including total ROI and c) develop gap analysis by determining an FMO (Future Method of Operation), a PMO (Present method of Operation) and developing a transition plan for going from PMO to FMO.

4. Illustrative Case Study: SOA Planning Through AIM

To illustrate the main issues addressed by AIM, let us consider the following case study about a retail store (Xshop). To improves sales, the company needs a very flexible online purchasing (OP) application that is based on SOA. The company needs help in addressing the following issues: what other applications interface with OP, how will they be impacted if OP is transitioned to SOA, what happens if OP is outsourced and hosted elsewhere, how will OP be accessed from a wide range of user devices, what type of integration technologies will be most suitable, and what will be the cost of transitioning OP to SOA?. Additional issues include: are there commercial-off-the-shelf products that can be used for OP, what type of middleware technologies are needed to support different architectures, which ESB (enterprise service bus) platform should be used, what are the performance and security tradeoffs when different components of this application participate in B2B trade, and what type of cost/benefit analysis need to be considered while evaluating these alternatives. These are non-trivial questions that require a great deal of time and effort to answer in a purely manual approach. In the following sections, we will illustrate how AIM can possibly help.

4.1. Stage 1: Business Problem Exploration - Understanding the Problem

This stage is supported through the Business Problem Explorer (BPE) that allows the users to browse through the AIM knowledgebase to select applications that will participate in an integration project. For example, the user selects OP by using the PISA knowledgebase. The knowledgebase consists of 3 parts: pattern[1] repository, object models, and COTS database. The Pattern Repository (PR) plays a central role in AIM because we heavily use patterns to quickly develop solutions. In particular, i*ndustry patterns (IPs)* are the main starting point for this stage. Figure 7 shows example of an industry pattern (IP) that captures a high level view of a retail company, similar to XShop, in terms of enterprise functional areas (e.g., sales, corporate management, back-office operations, supply chain management), the major business processes in each functional area (e.g., purchasing and payment within sales) and the key interactions between these processes. We have created IPs for 12 industry segments that include manufacturing, healthcare, telecom, and others. These patterns are stored in the Patterns Repository, part of the planning knowledgebase, as XML documents so that they can be analyzed and modified based on a simple interview.

The user starts by invoking the BPE to choose an industry segment and thus fetching appropriate IP for that industry. The user reviews the IP, modifies it if needed, and selects the critical applications that drive the SOA projects. For example, Figure 8 shows the result of choosing order processing (OP) as a critical application. This screenshot of BPE shows the external interfaces of order processing application such as selling chain management, purchasing, customer payment, These are the applications that will be affected if OP was transitioned to SOA and thus help in understanding the complexity and the impact of transition OP to SOA.

---

[1] A pattern, simply stated, is a sketch that can be refined and specialized for different situations.

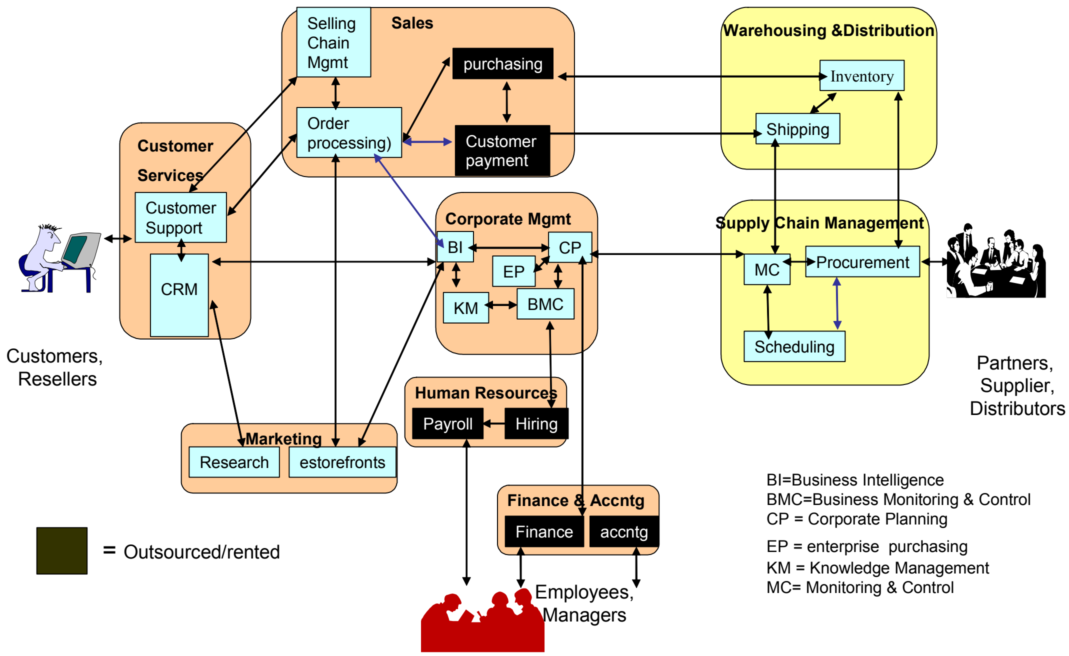


Figure 7: Industry Pattern (IP) for a Retail Company

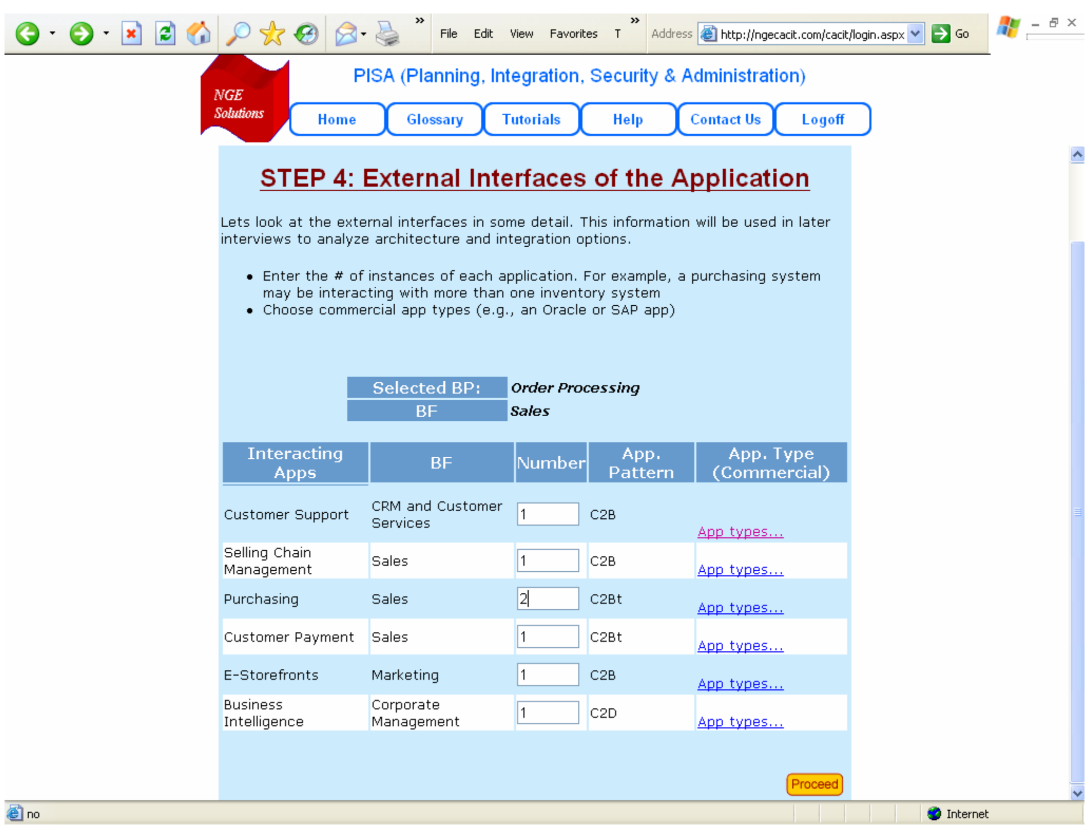


Figure 8: Selection of Order Processing as a Target (Critical) Application

4.2. Stage 2: Integration Requirements Generation – Documenting the Problem

Development of an integration plan is a complicated task with many challenges. Instead of a 'big bang' approach where all enterprise systems are converted to SOA in an afternoon, a gradual approach is better where the enterprise achieves an integrated architecture one application area at a time. Before proceeding with technical decisions, it is important to develop an understanding of the problem, establish a business case by identifying the applications that will actually participate in an SOA project and capture the key integration requirements.

Development of integration requirements is an important but extremely time consuming process. An Integration Requirement Generator (IRG) helps the user to quickly develop a requirements document. The heart of IRG is an interview that starts with the information already captured by the BPE in the previous stage. Additional information is gathered through the interview that considers factors such as user access, back-end apps, B2B apps, transaction value, transaction volume, number of partners, mobility, personalization, etc. Figure 9 shows a partial snapshot of the interview. The outputs of this interview are used to populate the requirements document. In short, to develop a requirements document for integration of online purchasing application, the user basically fills out an interview form shown in Figure 9. As a result of this interview, IRG selects appropriate integration patterns from the Pattern Repository and customizes them based on the results of the interview.

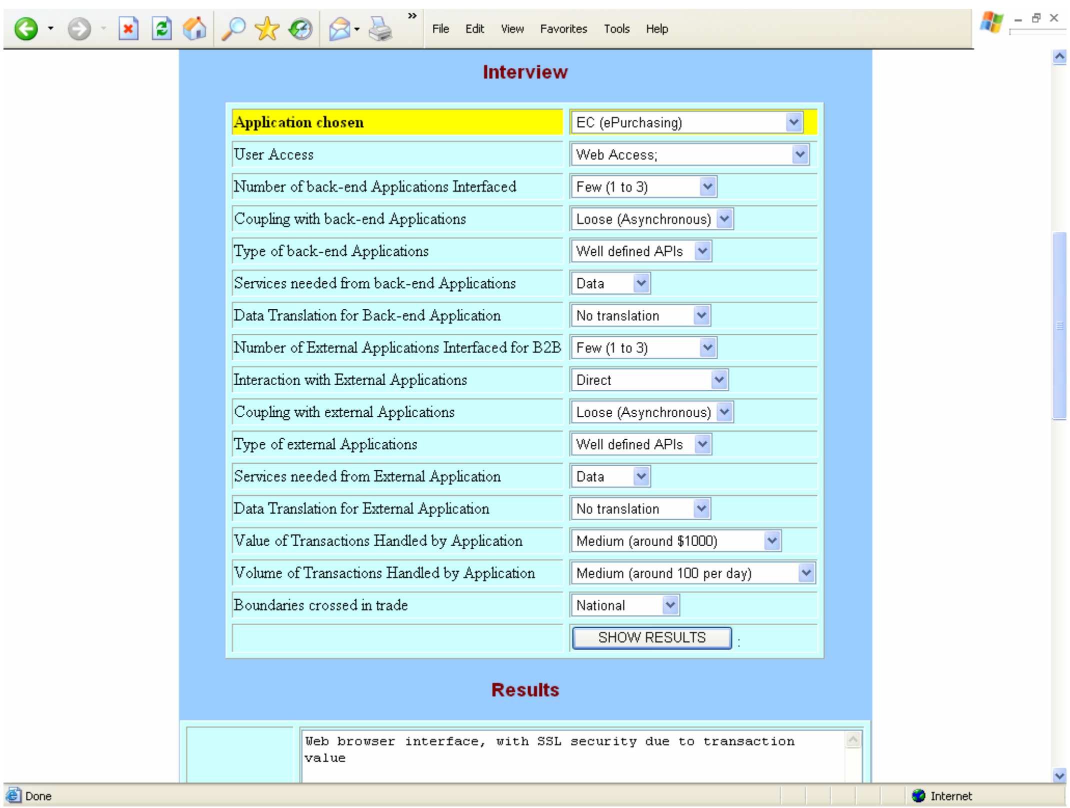


Figure 9: Sample Interview

4.3. Stage 3: Integrated Architecture (SOA) Stage – Capturing the Complexity

This stage translates the requirements model created in the previous stage into a component based Service Oriented Architecture (SOA). The output of this key stage is a detailed architecture document that captures the complexity of the problem and translates it into SOA features by using the following steps (see Figure 10):

- **Development of a Logical Architecture based on SOA (shown in** Figure 10**a):** Assumes that an application consists of N large grained components, each providing a set of business services. The components are arranged in several tiers: front-end integration, business logic, etc. This logical architecture can be used to determine the different types of adapters needed for different tiers.
- **Selection of Integration strategies (shown in** Figure 10**b):** The user chooses an integration strategy such as integration in place (i.e., integrate existing systems without changing any), data warehouses (develop a common database to be shared by multiple applications), migration (gradual or sudden replacement of existing apps) or composite (all of the above). This helps in selection of the SOA patterns for different integration strategies.
- **Construction of a Physical Service Oriented Architecture (shown in** Figure 10c**)**. In this step, the logical architecture is translated into a physical architecture by using SOA patterns. . The appropriate SOA ESB (Enterprise Service Bus) configuration plus the infrastructure components (adapters, registry, hubs, zones, etc) are chosen to support the different integration strategies shown in Figure 3a.

The Integrated Architecture Advisors (IAA) supports this stage by invoking three different interviews support the aforementioned three steps. These interviews gradually capture the complexity of the integration problem. Figure 11 displays a sample interview that shows the type of information (e.g., type of platform, types of services needed, and the type of data translation) needed for each application that interacts with order processing (e.g., customer support, selling chain management, and purchasing). This interview identifies the types of integration adapters that will be needed to integrate order processing with its interacting applications. As a result of the interviews in this stage, a detailed architecture document is generated that contains the adapter information, the ESB features needed, and overall SOA-based architecture.

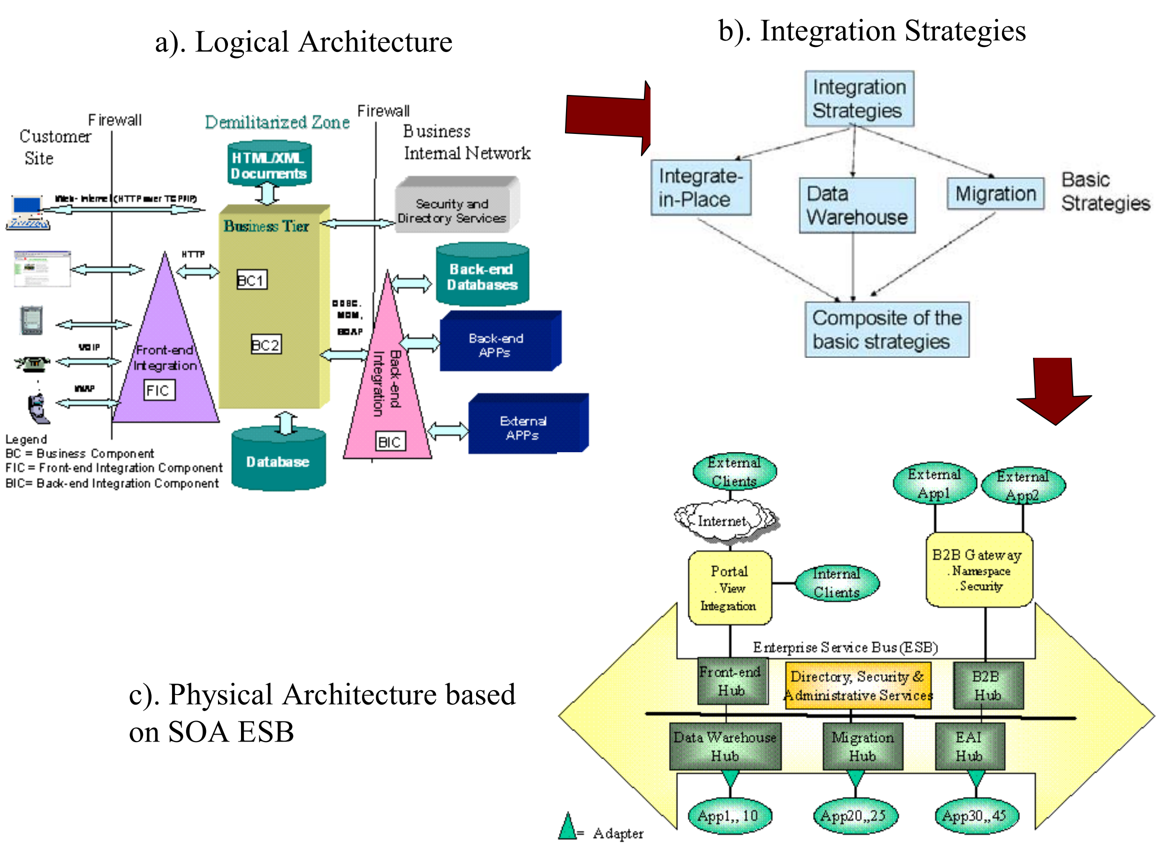


Figure 10: Gradual Development of an Architecture

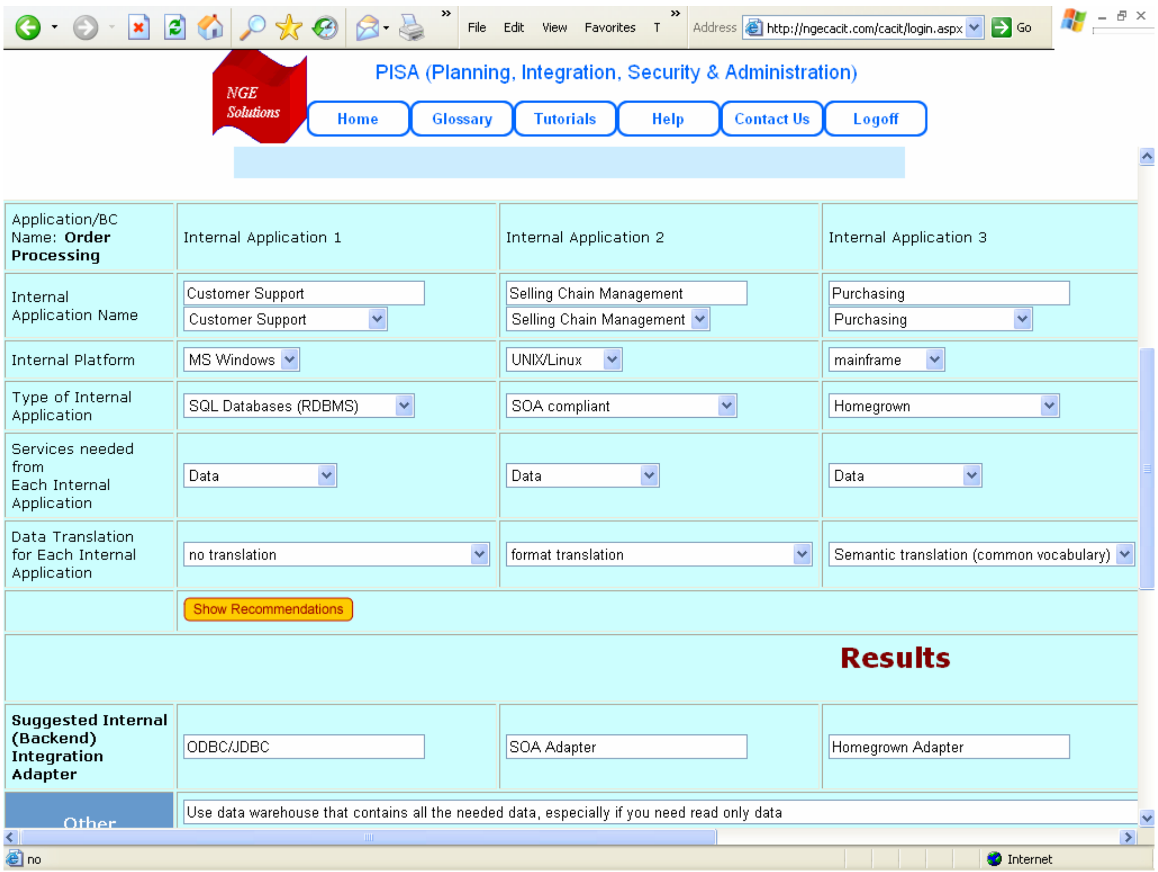


Figure 11: Interview to Capture the Complexity of the Problem

4.4. Stage 4: Solution Evaluation – Cost, Security and Performance Analysis

This is the most important stage from a management point of view because it involves estimation of costs, performance and security issues for each architecture solution. Specifically, this stage goes into further details by translating the SOA architecture *A,* produced previously, into plausible integrated solutions (*S1, S2,,Sn*) with appropriate commercial-off-the-shelf (COTS) packages and cost/ performance/security evaluations. The main activity of this stage is to evaluate the solutions (*S1, S2,,Sn*) in terms of the following:

a) **Cost estimates** due to the chosen architecture. The cost estimates are based on the complexity of the ESB chosen, the type and number of adapters needed, platforms to be bought/used, commercial-off-the-shelf (COTS) packages to be used, etc. Cost estimates include:

- Platform costs that show the ESBs, front-end portals, B2B gateways, adapters, and other platform component costs
- Development costs that show the development costs (e.g., developing an adapter) and installation/maintenance costs
- Miscellaneous costs that include training costs and the costs of rework due to errors

b) **Security implications** based on SOA and other technologies. The security issues due of SOA are investigated by using attack trees and security patterns. In particular, the following security issues are noted:

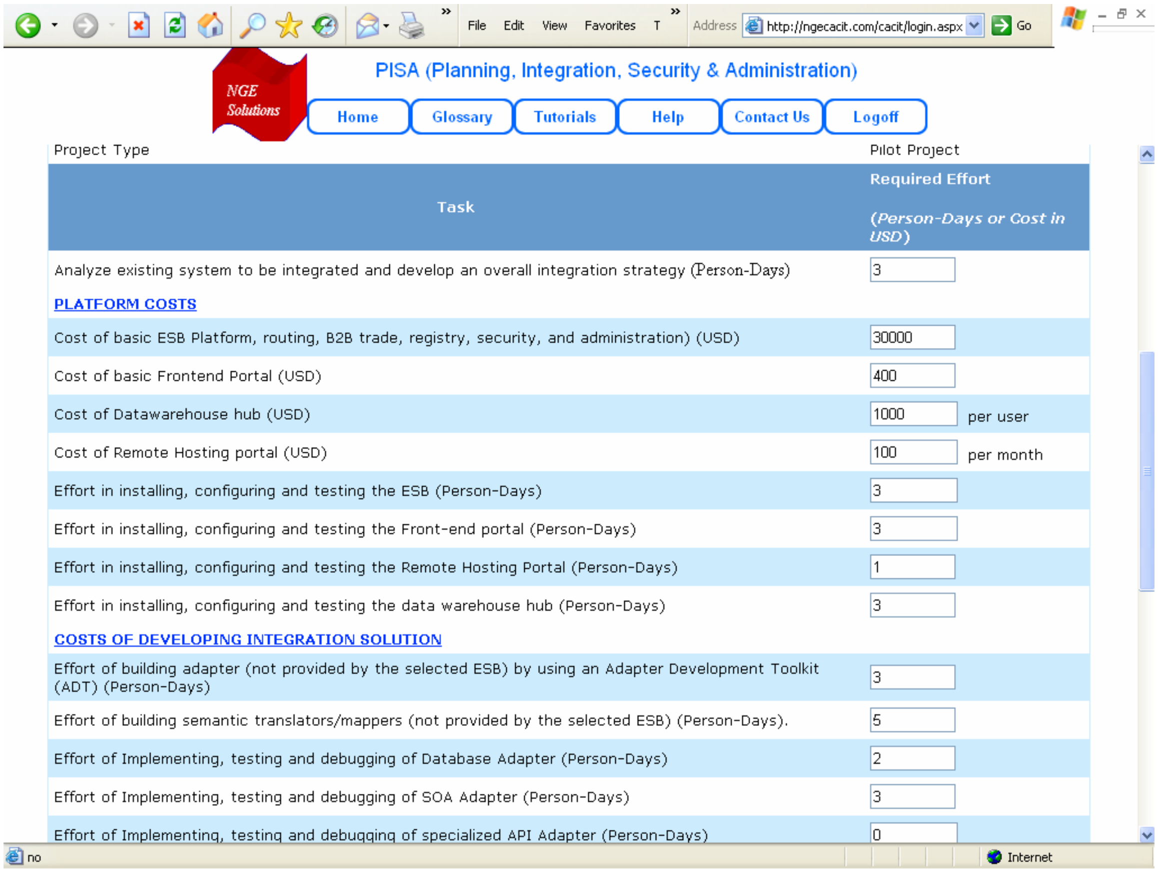
- Security of the ESB facilities (e.g., protecting the ESB directory)
- Security of the service providers and service consumers that use the ESB

c) **Performance implications** based on the configurations and allocations. An analytical queuing model is used to estimate performance bottlenecks. The main focus is to determine how many servers will be needed to support the ESB.

These steps produce a table (Table 1) showing the evaluations for different solutions for the order processing application. This stage produces several details reports. Figure 12 shows partial view of a sample cost estimation report produced by the Integrated Solution Advisor (ISA) that supports this stage. The sample report shows the platform as well as development costs.

Table 1: Example of Solution Evaluations for a Small Company

Choices	Estimated Costs ($)	Performance	Security Issues	Comments
Integrate	$120K (it is relatively cheaper to install an ESB and adapters)	2 seconds. (adapters introduce delays)	ESB & adapters may be targets for attacks & need to be secured	May need to migrate in future
Migrate and replace with an ERP	$500K million (it is expensive to completely replace a system with an ERP system)	1 second (no adapters are needed, hopefully, for an integrated ERP system)	Security can be designed for the new system from scratch	Migrations are typically expensive and require staff training
Data Warehouse	$200K (it is expensive to convert data and construct a data warehouse)	0.7 seconds (data level access is usually faster due to no overhead)	ETL needs to be protected, data level access needs protection	Data warehouses create duplicate data that needs to be synchronized



Project Type	Pilot Project
Task	Required Effort (Person-Days or Cost in USD)
Analyze existing system to be integrated and develop an overall integration strategy (Person-Days)	3
PLATFORM COSTS	
Cost of basic ESB Platform, routing, B2B trade, registry, security, and administration) (USD)	30000
Cost of basic Frontend Portal (USD)	400
Cost of Datawarehouse hub (USD)	1000 per user
Cost of Remote Hosting portal (USD)	100 per month
Effort in installing, configuring and testing the ESB (Person-Days)	3
Effort in installing, configuring and testing the Front-end portal (Person-Days)	3
Effort in installing, configuring and testing the Remote Hosting Portal (Person-Days)	1
Effort in installing, configuring and testing the data warehouse hub (Person-Days)	3
COSTS OF DEVELOPING INTEGRATION SOLUTION	
Effort of building adapter (not provided by the selected ESB) by using an Adapter Development Toolkit (ADT) (Person-Days)	3
Effort of building semantic translators/mappers (not provided by the selected ESB) (Person-Days).	5
Effort of Implementing, testing and debugging of Database Adapter (Person-Days)	2
Effort of Implementing, testing and debugging of SOA Adapter (Person-Days)	3
Effort of Implementing, testing and debugging of specialized API Adapter (Person-Days)	0

Figure 12: Sample Cost Estimates

4.5. Stage 5: Reiterate and Consolidate Results at Conclusion

The objective of this stage is to re-iterate, consolidate the results from different projects and do gap analysis. The consolidation effort may be zero for small single application projects but may be considerable for enterprise-wide application integration projects that require many applications to be integrated. For large scale projects, each iteration handles only a few applications so several iterations are needed. There is a need to consolidate results at the end of iterations and to develop an overall business case, including ROI (return on investment) and gap analysis. *G*ap analysis can be conducted by using the following approach:

- From the solutions (*S1, S2,,Sn*) produced in iterations 1 through n, respectively, choose the best solution S* based on evaluation
- Use S* as the FMO (future Method of operation)
- Use S0, the current system, as the PMO (Present method of Operation)
- Do a cost-benefit and ROI analysis of transitioning from PMO to FMO This includes tangible as well as intangible costs as well as benefits of the PMO, the PMO and the transitions.

Figure 13 shows a sample of gap analysis. Although some of these gap analysis and ROI calculations can be done manually, it is virtually impossible to do a good gap analysis without having a clear picture of the FMO based on a systematic analysis and an automated tool as suggested by the various stages of this model.

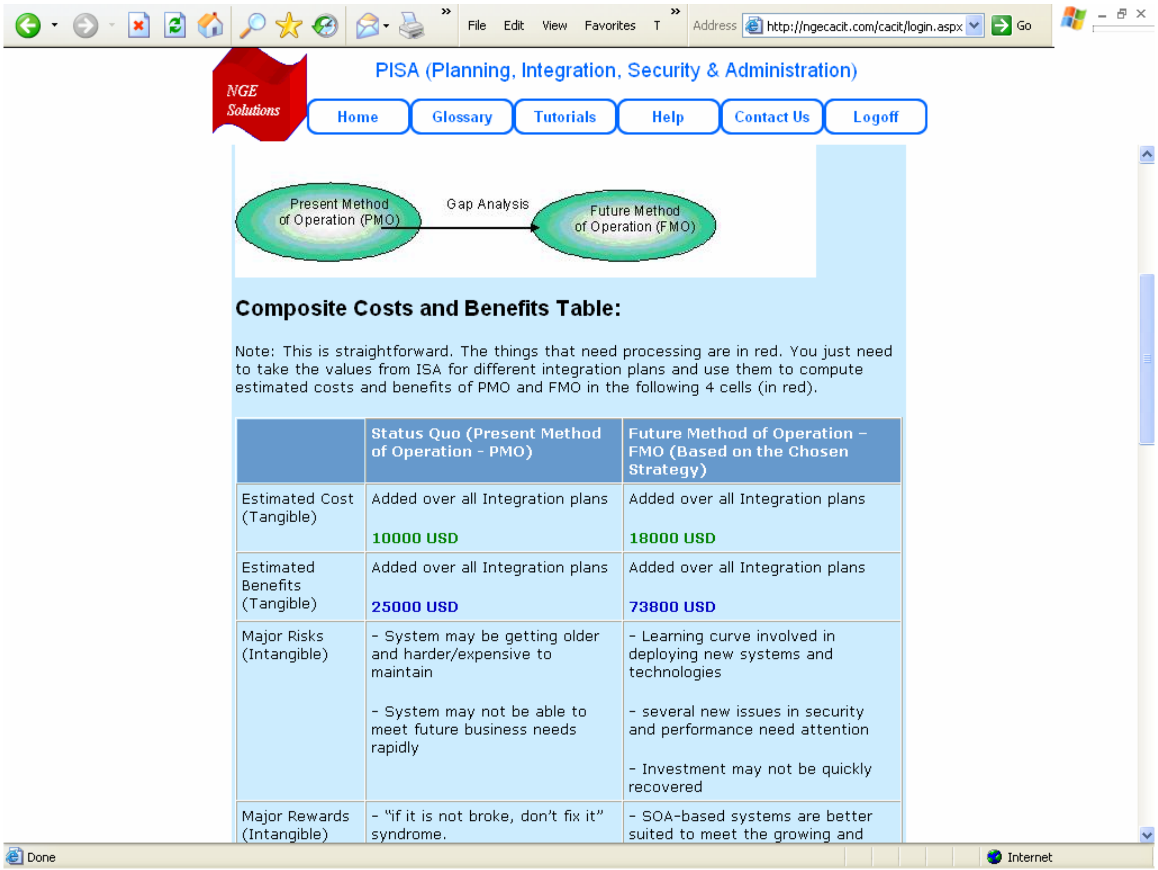


Figure 13: Sample Gap Analysis

5. CONCLUDING COMMENTS

SOA projects are complex undertakings that require detailed analysis. Instead of high level assertions, PISA-AIM guides the users to quickly develop SOA plans based on patterns, best practices, inferences, and collaboration. A system of this nature has not been reported previously in the literature. The PISA-AIM system, operational as a beta version at present, has been used in six consulting assignments (others are in progress) in which the users (mostly IT managers from small to medium firms) developed business cases for SOA and SOA plans before embarking on an integration project. In addition, AIM has been used to teach five systems design, enterprise architecture and integration courses so far with very encouraging results. In each course, the students were assigned three projects: 1) manually develop an integrated architecture for an SMB that is going through a major re-engineering effort, 2) use AIM to solve the same problem, and 3) use AIM for a project of their own interest. Most students had a great deal of fun with the third project -- they built models of different businesses and developed integrated architectures by using AIM for "what-if" analysis of different scenarios. We are currently negotiating with several universities and businesses for additional experiments and are using our current experiences and lessons learned to guide future research and development directions.

SELECTED REFERENCES

[1] Adams, J., et al., Patterns for e-Business: A Strategy for Reuse, IBM Press, October 2001.

[2] Alexander, C. et al , *A Pattern Language*, Oxford University Press, 1977

[3] Buschmann, E., et al., Pattern-Oriented Software Architecture, Vol. 1: A System of Patterns, John Wiley, 1996.

[4] Fox,M. and Gruninger, M., "On Ontologies and Enterprise Modeling", International Conference on Enterprise Integration Modeling, 1997, pp. 40-46

[5] Gamma, E., et al., Design Patterns, Addison Wesley, 1994.

[6] IBM e-Business Framework website http://www-106.ibm.com/developerworks/patterns/

[7] IDEAS (Interoperability Development for Enterprise Application and Software) Roadmap: “State of the Art Summary”, www.ideas-roadmap.net, deliverable D1.1, Mat 2003.

[8] Linthicum, L., "Enterprise Application Integration", Addison Wesley, 1999

[9] Linthicum, L, “ B2B Application Integration: e-Business-Enable Your Enterprise”, Addison Wesley, 2001

[10] Shore, B., “Enterprise Integration in Globally Disbursed Service Organization, Comm. Of ACM, June 2006, pp. 102-106.

[11] Umar, A., "Intelligent Decision Support for Enterprise Architecture and Integration", Informatica Journal, May, 2007.

[12] Umar, A., "Application (Re)Engineering in Distributed Environments", Prentice Hall, 1997.

[13] Umar, A., e-Business and Distributed Systems Handbook, Architecture and Integration Modules, NGE Solutions, August 2004. (Umar 2004a)

[14] Umar, A., Mobile Computing and Wireless Communications, NGE Solutions, July 2004 (Umar 2004b).

[15] Umar, A., Information Security and Auditing in the Digital Age, NGE Solutions, August 2004 (Umar 2004c).

[16] Umar, A., “IT Infrastructure to Enable Next Generation Enterprises", ISF (Information Systems Frontiers) Journal, Volume 7, Number 3, July 2005, pp: 217 - 256. (Umar 2005c).