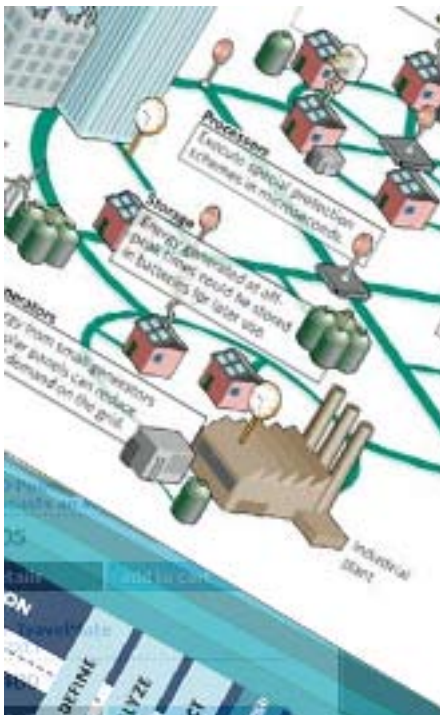


B E Y O N D

Think Beyond. Think Mphasis.

WHITE PAPER



EAI - A Strategy for the Practice

Kevin Rodrigues
Senior Architect

August, 2010

Table of Contents

1. EXECUTIVE SUMMARY	2
2. INTRODUCTION	2
3. BROAD OUTLINES OF THE STRATEGY	2
4. ANALYSIS AND UNDERSTANDING OF THE CURRENT APPLICATION LANDSCAPE	2
5. GLOBAL INTEGRATION ARCHITECTURE	3
6. THE INTEGRATION APPROACH – BOTTOM UP OR TOP DOWN	4
7. CORE TEAM CONSTITUTION	4
8. ENTERPRISE WIDE DATA MODEL	4
9. SOFTWARE DEVELOPMENT LIFE CYCLE AT VARIOUS LEVELS	5
10. APPLYING SDLC ACROSS EACH INTEGRATION LAYER	6
11. CONCLUSION	7

1. Executive Summary

In an earlier white paper titled 'J2EE And EAI – A Primer For The Practice', we touched upon technical aspects pertaining to integration. For an integration project to be successful, however, technical fundamentals must be complemented by a sound implementation strategy. This strategy should comprise of

- (a) An accurate plan,
- (b) A detailed integration related work breakdown structure and
- (c) Execution of the plan with near clinical precision

On the face of it, this seems no different from other software development kind of projects. An EAI project, however, has many more dimensions attached to it. Some of these may include the basic work culture of an organization, reluctance to change arising out of political motivations within the organization, dealing with a variety of stake holders such as end users, IT departments, security, developers, managers, etc; each with different points of view. These typically are clubbed under the umbrella of (SOA) governance.

There is a need to define best practices for successful EAI. This paper intends to define such an approach for successful integration.

2. Introduction

Automation is the key. Integration will eliminate the bureaucratic hurdles and inefficiencies that are generally associated with manual tasks or human workflows. Most IT developers within an organization realize this and inadvertently tend to build partial solutions towards integration. Partial integration solutions are also built as a quick fix response to critical situations that arise. Data level integration may be adopted to immediately douse the fires of a crisis. Gradually, over a sustained period, the organization finds itself in the throes of all the ills of a bottom-up integration approach leading to stovepipe systems. Sufficient resources in terms of time, money and developers are never really available; real authority to undertake integration has never really been granted and most importantly, the global vision towards integration has never been envisaged. Compounding this is another problem – that of mergers and acquisitions of companies that themselves have mushroomed into an integration mesh. Falling into the trap of such integration can be avoided if one adopts a sound strategy for integration

3. Broad Outlines Of The Strategy

Prior to undertaking any integration endeavor, the outlines broadly mentioned below must be followed –

- (a) Analyze the current application landscape from a functionality perspective.
- (b) Understand the current application landscape from a technical perspective.
- (c) Define the global integration architecture.
- (d) Decide upon the approach to integration¹.
- (e) Constitute a core team which will be responsible for the successful integration, but with reasonable executive authority vested in them.
- (f) Define the enterprise wide data model and the data migration approach.
- (g) Apply typical software development life cycle processes for integration at each of the levels.
- (h) Ensure early Management Funding and buy-in.

4. Analysis and Understanding of the Current Application Landscape

The functionality of each of the applications needs to be understood not only in isolation, but also in the context of the business domain they operate in. It is important to note that, from a functionality perspective the current application landscape does not only include the generally visible applications across the organization. It also includes all the 'home grown tools and utilities' which individual users have developed to improve their own productivity. These tools and utilities after all became necessities to fulfill some needs that were lacking in the applications themselves.

From a technical perspective, an analysis of the variety of programming languages, operating systems and programming models that may have been used needs to be undertaken. This variety manifests itself as a potpourri of –

- Monolithic, client server as well as distributed applications.
- Procedural as well as object oriented applications.
- Hierarchical, relational as well as object oriented databases.

- Communication protocols used in the form of RPC, Message Oriented Middleware, ORBs, etc.
- Proprietary data formats used for data interchange and sharing.

5. Global Integration Architecture

To achieve integration, one could use either replacement or extension.

Replacement is a quicker solution, but it is an option if and only if there are commercially available applications that suit the needs of the organization. This rarely is the case. Commercially available applications never really comprehensively match the organization's custom applications. Applications such as Baan, People Soft and SAP were intended to be used as ERP systems that replaced custom applications. Nevertheless, they too had to provide interoperable interfaces so that the functionality they lacked could be provided by custom-built modules.

One could re-engineer the solution from scratch, if not purchase it, but the cost to benefit ratio in this case would be extremely high - years of development of the systems required a lot of knowledge and very rarely is this knowledge archived. Years of testing have led to system stability. Data migration issues would arise because historical data would need to be migrated from the old systems to the new systems. Personnel would need to be trained to use the new systems. All these suggest that replacement is simply infeasible.

Application extension does not take into consideration the global integration landscape. It simply modifies applications to suit short-term needs. It makes the systems tightly coupled and the overall integration landscape becomes extremely brittle. Application extension is very limited in nature and thus is not really a feasible option either.

The correct approach would be to have a global picture of the landscape and define a global level architecture for the integration approach. One ought to start with a grand vision for the final integration topology of the entire enterprise. The final integration architecture should be designed so as to reflect the business processes and not an assembly of application functionalities that force solutions. Arriving at the grand vision is central to successful EAI and is easier said than done. The vision will evolve iteratively, after a series of activities. Some of these activities, not exhaustive though, would involve –

- Making an assessment of the organizations business processes,

- Understanding the business domain that the business processes operate in,
- Review of documentation pertaining to various applications that need be integrated,
- Making an assessment of the IT infrastructure,
- Conducting inter-departmental workshops along with the involvement of IT department and subject matter experts
- Doing a cost-benefit analysis for return on investment,
- Identifying the stakeholders – decision makers who are the main beneficiaries of an EAI,
- Understanding global data in terms of an organization wide single unified data model,
- Identifying which off the shelf tools can be used for integration and then short listing appropriate vendors without entailing any vendor lock-in.

The global level integration architecture must address the various cross cutting concerns such as communication, routing, data transformation/mapping, business intelligence, rules, transactions, security, scalability, latency, auditing, extensibility, logging, exception handling, etc. The use of these cross cutting concerns must be standardized and this standardized usage must be enforced as part of the development best practices. The architecture also needs to identify the various integration infrastructure technologies that need to be used. Such an integration infrastructure is typically known as an integration broker, which nowadays is synonymous with what is known as an ESB.

The integration architecture would comprise of integration at various levels –

- data level,
- application level,
- business intelligence level and
- presentation level.

A good integration architecture should address the need for a unified enterprise wide data model in addition to componentizing the application via interfaces, managing asynchronous as well as synchronous communication, interoperability, message format translation, etc. The unified data model will ensure that there is a single version of truth, data will be entered only once into the system

One very important aspect overlooked in integration projects is that of information latency. In a distributed system, any change in data in one of the applications must be propagated immediately to the rest of the applications associated with that data. In other words - the latency of the system must be almost zero. This is an extremely difficult goal to achieve.

Another aspect to be kept in mind while defining the global level integration architecture is reuse. Each application being integrated serves a very specific purpose. One needs to think at a much higher level of abstraction about applications or parts of applications that can be grouped together to provide a business level solution. Such business level solutions become reusable within the enterprise as well as in B2B scenarios. Deep domain knowledge is essential to be able to visualize reusable solutions.

Apart from functionality reuse, there could be scope for object reuse. There might be some common objects that are relevant to many applications. Instead of any one single application owning this object and passing it via a bus to other applications, one might consider storing these global business objects centrally for coordinated access by all other applications – something akin to what naming services, directories, etc. do. In some EAI programs executed by MphasiS, this was referred to as the ‘Corporate Singleton’ and was used to store data such as time zones, currency, fiscal year, etc.

As part of the architecture, special focus must be paid to propagate an event driven paradigm. Each application being integrated needs to be analyzed with respect to its unpredictable behavior in the context of its business operation/domain. Events may be chained – one event may be handled by a sink (consuming application) partially, only to trigger another correlated event (or may even be an unrelated event) in another application.

Applications may terminate or crash mid way when a workflow is in progress or when a message is flowing through the system. The integration architecture must anticipate such possibilities and plan for a lossless system. Single points of failure must be eliminated and if not, then minimized and made tolerable. Repercussions through the system must be contained. The use of an effective failover configuration within the system should be considered.

Finally, the global level integration architecture must ensure extensibility. It must be designed keeping the future in mind.

6. The Integration Approach – Bottom Up or Top Down

The bottom up approach is very person-centric and not organization centric. It generally is highly ad-hoc in nature and is much uncoordinated. It looks at the specificity of problems, not at the global generality. With such an integration approach, we end up with a point-to-point mesh of applications. The maximum number of interconnections between N such applications would be $N(N-1)/2$. One can see that on adding just one new application to the system, the number of connections increases dramatically.

The top down approach on the other hand implies that one starts at the top with a global view of the integration landscape. From this vantage point, one can see the various existing application dependencies, one can define guidelines and priorities, and one can manage coordination. In essence, one can truly strategize the integration approach. Within the top-down approach some divide and conquer strategy is required – after all the EAI problem is too large to handle as a whole.

7. Core Team Constitution

Integration is not just a technical problem. It involves various stakeholders. Therefore, there is an issue of governance attached to it. Higher management buy-in, funding and ongoing support is essential for integration projects to be successful. This core team is responsible for conceptualizing and implementing the integration right from start to end.

8. Enterprise Wide Data Model

Data often is redundant and duplicated. Data maintenance becomes a nightmare. Generally, there is never a single version of truth for critical data that should have been considered master data. As part of the integration strategy, one needs to also adopt a revamp of the existing databases and rationalize it as much as possible. Not only should such master data comprise the golden source but this golden source might also need to have replicated copies that serve as golden references.

Data would then need to be migrated to these rationalized systems.

9. Software Development Life Cycle At Various Levels

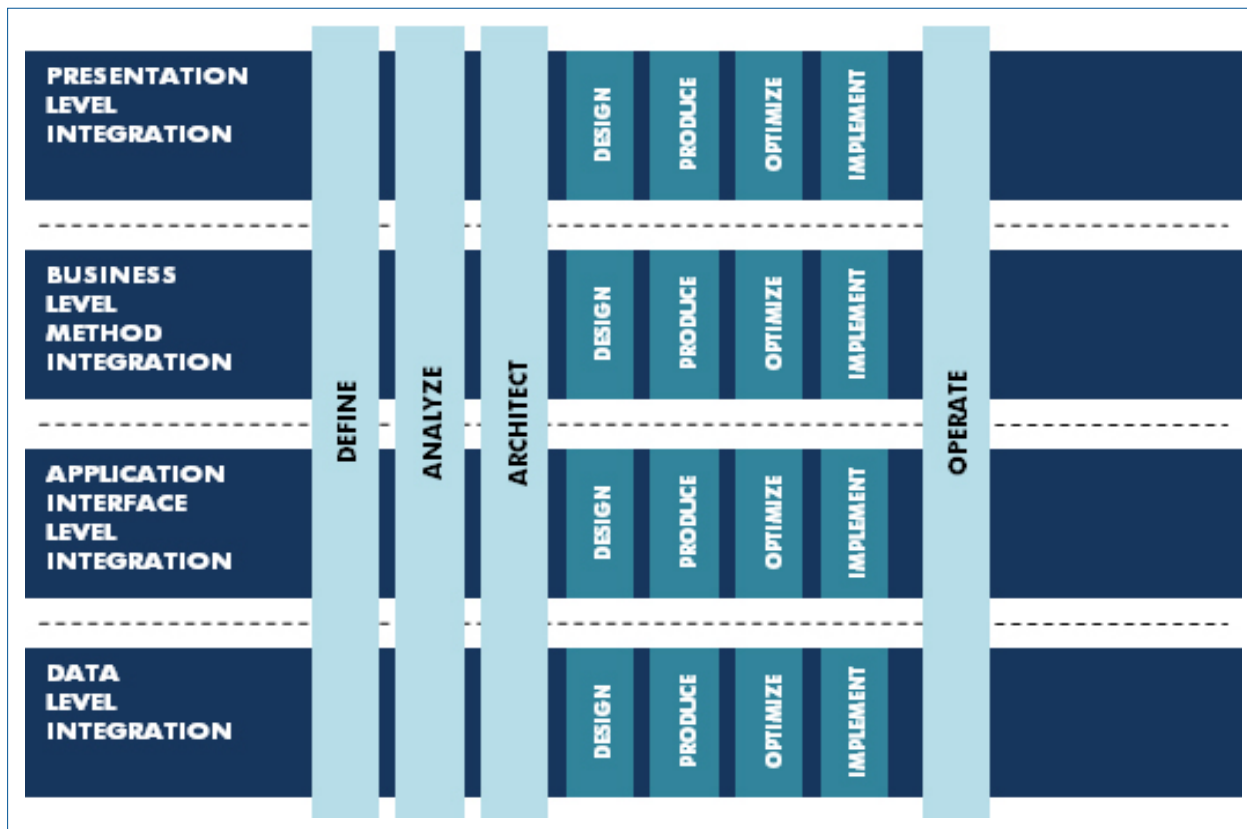
SDLC is quite a common practice for any software development project and is not much different for an EAI project. Below is a depiction of the SDLC used within Mphasis Software Services. One could define a better SDLC, but that, however, is not central to this paper. What is important is the application of various phases of the SDLC across each of the integration levels.



The SDLC consists of the basic high level activities – Define, Analyze, Architect, Design, Produce, Optimize, Implement and Operate. The end of each of these phases constitutes a checkpoint. At each checkpoint, various artifacts (key deliverables) must exist. For example, a project plan, a metrics plan, a communication plan and an Integration Requirements document are some of the artifacts at the checkpoint pertaining to the Define phase of the SDLC. Project Management, Technical Governance and Intellectual Capital (resources) are some activities and resources that span across the SDLC space.

10. Applying SDLC Across Each Integration Layer

In an EAI project, integration is possible at various levels as depicted in the diagram below, and hence the SDLC must be applied concurrently at each level of integration. Integration at each level ought to be considered as a sub project in itself. Define, Analyze and Architect phases of the SDLC can be common to all levels. The Design, Produce, Optimize and Implement phases may be local to each level. The operate phase would again span all levels.



The SDLC process at each of the levels could follow a spiral model. The entire integration landscape needs to be partitioned and each partition developed incrementally. Each set of applications contributing to a business process or workflow would constitute a partition. Starting with one partition, at each stepwise refinement of the development process, additional partitions may be added until the entire integrated topology has been achieved. At each milestone in the incremental process, one would have a working prototype that can not only be tested but also can also be profiled and compared.

One often missed activity in the define and analyze phases of the SDLC is that of thorough investigation of existing applications. One tends to get on with the integration act in a hurry and therefore pays only cursory attention to the functionality of existing applications in terms of basic functionality, communication protocols used, interfaces and data. One often omits taking into consideration any

documentation pertaining to the application, undocumented functionality, complexity of internal algorithms of the applications, configuration information of the applications, impact of integration on data integrity and on security, etc. In the design phase of the SDLC, one tends to overlook issues such as transactional behaviors (atomicity, consistency, isolation and durability) synchronization, deadlocks and circular transitions. Achieving these in a distributed system is extremely challenging. The design should aim to use EAI patterns such as aggregator, content enricher, splitter, content-based routers, etc.

The implementation phases are relatively easier in an EAI project compared to developing an application because no new functionality is being developed. However, this phase does not preclude new development to extend the functionality of an existing application. The need for such new functionality should be actively sought out in the design phase. Extending application functionality should be treated as a separate project, distinct from the EAI project.

The implementation phase should not be considered completed unless there is elaborate documentation. Maintenance of an EAI project without documentation can be a nightmare on a much grander scale as compared to maintenance of applications per se without documentation.

In the testing phase, one needs to make sure that regression testing is carried out. Any change in one area of the integration topology can have a distinct ripple effect in another area. Early in the architecture phase tracing should be incorporated. Tracing should be considered on par with other cross cutting concerns such as exception handling, logging, auditing, etc. This would help debug integrated applications. The testing and deployment approach otherwise is more or less similar to that in application development projects.

11. Conclusion

Many integration projects have failed not because of lack of technical capability, but simply because of the inability to breakdown the problem to be solved and then supplement it with sound integration strategies. Initial development costs of an EAI solution can be exorbitant. EAI solutions can be time and resource intensive. This is even more reason why an EAI strategy must be put in place.

Though this whitepaper is not exhaustive, it has attempted to contribute towards the success of any EAI initiative.

Contact us

USA

MphasiS
460 Park Avenue South
Suite # 1101, New York
NY 10016, U.S.A.
Tel: +1 212 686 6655
Fax: +1 212 686 2422

UK

MphasiS
88 Wood Street
London EC2V 7RS, UK
Tel: +44 208 528 1000
Fax: +44 208 528 1001

AUSTRALIA

MphasiS
410 Concord Road
Rhodes, NSW 2138, Australia
Tel: +61 290 221 146
Fax: +61 290 221 134

INDIA

MphasiS
Bagmane Technology Park
Byrasandra
C.V. Raman Nagar
Bangalore 560 093, India
Tel: +91 80 4042 6000
Fax: +91 80 2534 6760

About MphasiS

MphasiS is a leading Applications, Business Process Outsourcing, and Infrastructure services provider. The company delivers real improvements in business performance for clients through a combination of technology knowhow, domain, and process expertise. With currently over 37,000+ people, MphasiS services clients in Financial Services, Communications, Media & Entertainment, Healthcare & Life Sciences, Manufacturing, Transportation & Logistics, Retail & Consumer Packaged Goods, Energy & Utilities, and Governments around the world. To know more, visit www.mphasis.com