Managing risks on middleware projects – sharing the pain, and the gain

Nick Denning
Chief Technology Officer
Strategic Thought

Management introduction

Nick Denning is the Chief Technical Officer of Strategic Thought Group Plc (Wimbledon, UK) which he founded in 1987, having previously worked for Logica. Strategic Thought, since inception, has been involved with middleware, from Ingres through Tuxedo to the WebSphere portfolio and the services around these.

In this discussion, Mr. Denning:

- ***offers lessons from managing and delivering a wide variety of middleware projects for over 10 years***
- ***explores the problems that often make middleware projects difficult and different***
- ***identifies key characteristics of middleware projects***
- ***lists common risks relevant specifically to middleware projects***
- ***provides suggestions for managing risk on future middleware projects.***

Scope

There are many points where integration can occur. For example, you can integrate:

- **at the screen, using portal technologies**
- **between databases, by shipping data using replication technologies**
- **at the functional level, by passing messages between applications**
- **by coordinating messages into work flows, to bring together the work elements in an organization and build co-operating business processes.**

The effective exploitation of middleware is, therefore, critical to ensuring the increasing effectiveness and efficiency of activities. Being a competent master of middleware matters.

Some good news

The good news is that there is little fundamentally new to learn. Running a middleware project is largely the same as running any other large IT project. So do not expect to learn anything fundamentally new or shockingly revelatory in this analysis.

On the other hand there are some particular characteristics of middleware projects that are challenging but which, if identified, are readily addressed. Nevertheless the chances of incurring a failure in a middleware project are relatively high because we all seem to be institutionally predestined not to face up to the risks.

This may seem to be pointing out the blindingly obvious. Yet it matters. In order to discuss strategies, we must ensure that we are all looking in the right direction at the right moment to identify and react to the right risks.

Technology risk — more good news

You will (I hope) be pleased to hear that technology risk on middleware projects is relatively low. For instance a reasonably competent person can install products such as WebSphere MQ (WMQ, or the old MQSeries) and have messages being sent and received within an hour of starting. When the product is installed:

- **a default queue manager is created**
- **scripts can be run to build queues**
- **there are source code examples that illustrate most of the features of using the WMQ API**
- **compiled and built executable versions of various tools exist, which demonstrate message processing.**

Provision of a simple 'Getting Started' document assists a new user to obtain early confidence. (It is worth noting, however, that it took IBM almost 10 years to appreciate the power of this; in the early days of MQSeries you were expected to be way more understanding of the arcane issues that MQSeries brought with itself. It would be ever so agreeable to think that IBM, for other products, could think and act in a similar fashion.)

EIA hub products are, necessarily, more complex to install and for starting the building of applications. At Strategic Thought we advise client organizations to obtain an understanding of base messaging products before embarking on using hub technologies. That said, a new programmer with VB, SQL experience and a basic understanding of XML should get up to speed quickly if working as a member of a team where advice can be sought and the practicalities of decoupled (asynchronous) processing can be explained.

Although the J2EE-based technologies that now underpin middleware software products introduce their own complication, they do add new capabilities. These can also be exploited with effective training, that covering the J2EE environment.

Still more good news?

Middleware solutions can usually be decomposed into discrete process components. When this is accomplished, these can be implemented independently — and in parallel.

Estimating the time to implement middleware transactions is relatively straightforward. To build flows in IBM's WebSphere Message Broker we (at Strategic Thought) have a broad brush estimate of 5 days for a simple flow, 10 days for a medium flow and 20 days for a complex flow. This has stood us in good stead as a broad basis for 'first cut' estimations.

Thus, once development starts, a middleware project should be able to demonstrate a steady delivery stream of artifacts which can then be deployed and tested independently of the other system components. This is a largely unsung benefit — but it is a significant one.

Where is the catch? What stops all middleware projects being successful?

There are several catches. Perhaps they can best be summed up by saying that, while the technology is easy, the problem is difficult.

Why is the problem so difficult? Well primarily because middleware projects are all about linking together many existing systems to implement processes that span the enterprise. This means that all of the following:

- **the problem**
- **the perspective**
- **people**
- **the plan**

need to be managed. This is not trivial, as I describe below.

The problem

Generally, at a high level, it is possible to obtain a reasonable understanding of most business problems that need to be faced. It is, however, unlikely that any one person will possess a full view of the whole problem domain that needs to be encompassed. Collectively the problem may be understood (everyone can see a part) but this does not mean that there is a single coherent or central view of the complete problem.

The complexity of problems is then made an order of magnitude more difficult if it crosses organizational boundaries — because the 'dimension' of the problem expands. Problems become increasingly more challenging as the number of separate functional areas involved increases.

The devil is in the detail. What seems simple at a high level often masks significant issues when implementing a process. One example is what is often referred to as an impedance mis-match — when we try to connect two subflows together and realize they do not obey the same protocol or have inconsistent data items in their payload.

This can mean that we have to reconcile ourselves that we cannot solve the complete problem. Instead we find ourselves:

- **matching the business requirements with the ease of implementation**
- **prioritizing**
- **then applying the 80:20 rule**
- **before constructing a development plan — with work-arounds to deal with scenarios it was not thought cost effective to implement.**

The perspective

It is common to find a misunderstanding of the nature of middleware. In one instance a client fondly imagined that, simply by installing WMQ, it had automatically message queue-enabled its SQL applications. The client had no concept that middleware is just an API which enables applications to inter-communicate and which requires communications code, using the API, to be written.

Crazy? Well, no. There are some software products supporting standard protocols which, when installed, can be detected by other programs and automatically exploited.

Perhaps even more fundamental, most developers building an application naturally fall back on familiar 'stovepipe process thinking' when building their solution:

- **screens, to demonstrate functionality**
- **a data model, to hold the business data**
- **code to glue the database to the screens.**

This stovepipe approach must be discarded in favor of a 'middle-out' design and implementation approach. Effective direction from an IT or enterprise architect can be successful in appropriately aligning a team.

People

There are many people involved in any middleware project — often with (or from) a wide range of systems, each with a vested interest to protect. Often those interests do not align.

Strong management and leadership is required to:

- **set common goals**
- **maintain the focus on achieving those goals.**

Imagine trying to put together a jigsaw puzzle when every separate piece has a mind of its own and is looking for the other parts of the jigsaw that have the right shape to fit together with it. (This is in fact easier than real life — because it pre-supposes that all the pieces are on the board and that there is a solution. In real life this will likely not be the case.)

You can extend this analogy to propose that you need a project manager to keep the bits of the puzzle in order and organize that each piece be in the right location. This would entail working with an architect who is first cataloging the pieces and then working out where they should go.

I hope this convinces you of the need for an appropriate management structure.

The plan

How do you monitor progress on a middleware project if there is nothing to see (middleware is terrifyingly intangible for those who like to 'see or feel' what they are buying)? How do you provide a demo when there are no screens? If you cannot monitor progress, how can you determine how much longer to continue if the project is delayed? When should you give up and pull the plug? Loss of confidence in a middleware team is a major cause for middleware projects to be canceled.

It is vital, therefore, to structure and plan a middleware project. Once the plan is signed off then, provided the staged deliverables identified in the plan are delivered on time, why would anyone question the project performance, regardless of the actual nature of the deliverables? The point is that you have to think very carefully about organizing projects to ensure confidence is maintained in the delivery approach.

Middleware design

Now let me consider how you might identify some of the special characteristics of a middleware project. I am going to focus on those that can make such projects different from a normal application development.

A middleware project requires a detailed design which is message oriented and identifies how messages must:

- **possess defined formats**
- **be consistent with an agreed protocol for messages generated by the source**
- **enable a target destination to be identified**
- **be routed to that destination**
- **be transmitted along the chosen route**
- **are transformed (so that they are in a format that is understood by the receiving application)**
- **be subject to an agreed protocol conversion (so that the resulting messages comply with the protocol of the receiving application)**
- **be received by an interface and consumed, causing business logic to be executed on the receiving application/system.**

In addition, it must be determined whether order matters. It may be necessary to preserve order so that messages are delivered in the same order as they are generated.

Then there are 'quality of service' considerations. A service level agreement should ensure that transmissions are reliable (so that messages are not lost) and that a stated percentage of message transmissions are completed end to end and within an agreed interval. It is highly desirable that it be possible to detect and manage any messages that become suspended within the middleware infrastructure — so that they can be either canceled, or restarted and transmission completed.

State management

A crucial difference between a middleware project and a standard business application is the need to focus on state management in a more sophisticated manner than just having an underlying relational database. Stateless integration means that the processing of messages is independent of any messages that have gone before or will go after. Identical messages are processed identically. Once the processing of stateless messages is completed the system has no knowledge of its existence other than perhaps in an audit log file. Typically a stateless message is completed in a single 2PC transaction and has no side effects — such as updating a database — which would cause the behavior of subsequent messages to change.

It may be necessary, therefore, to hold information within message systems to facilitate any of the stages of message processing — such as translation look up tables or routing tables or other business rules data that controls how messages are processed. This data may be dynamically changing, for instance fed by real time data feeds (that update routing tables, transformation data or other business rules). At Strategic Thought we treat the management of this information separately from the 'statefulness' of the messages themselves.

Stateful messages encompass all other messages — where the processing of a message:

- **depends on messages that have been processed previously**
- **can impact the processing of messages that are processed subsequently.**

By definition business processes are stateful. Executing a business process will have a side effect of causing a business transaction that does work and creates value to complete. A business process will generally invoke sub-processes — some of which will be stateful and others which will be stateless.

In this context, an example of a stateless message is a

request for a price. Firing a 'price enquiry' message would find the item price but would have no side effects. In contrast, a 'place order' request would change the database and hence have a side effect. Subsequent processes may behave differently as future orders may not be accepted if stock levels have been exhausted.

On the other hand the design may not consider the message boundary at the adapter to adapter level and not include the database update. In this instance the 'place order' message might be considered stateless.

Another scenario for a stateful message is in financial services messages. A FIX message, for instance, participates in a protocol where by the FIX adapter maintains state on each message received and transmitted. Each FIX message contains the transaction number of the last message received so that FIX adapters communicating can verify that there have been no messages lost between the two connecting applications (the potential consequences of losing a message requesting a highly valuable transaction to be performed can be significant).

The important conclusion from this is that communications requires you to use a software stack — such as the ISO 7 layer model — and, depending on the quality of service, you will choose to maintain state in any of those layers that ensure the functional and quality of service obligations are met.

Evolution of middleware

Middleware continues to evolve. It is not static and is becoming ever richer. This needs to be taken into account along with the many lessons that have bean learnt as the industry has progressed:

- **from exchanging files (FTP, for example).**
- **to point to point messaging**
- **to EAI hub integration (or message broking)**
- **to stateful applications — where application messages involve multiple transactions to deliver a business process**
- **to the merging of automated application to application messaging with work flow (to consolidate EAI and work flow technologies into a process-driven Service Oriented Architecture).**

It is necessary to understand this path of evolution because with such understanding has come much experience with solving increasingly complex middleware problems. Many of these may not be apparent to designers on their first implementation.

Middleware projects — an organization structure

Middleware projects connect the business processes of an organization. Typically such projects will introduce organizational change issues.

It is therefore vital that the project sponsor is sufficiently senior to direct the resolution of issues across the affected departments. It also matters that this occur promptly, when issues arise.

In effect, the middleware team sits at the center of a web that represents the organization's business processes. It is vital that the responsibility of each team is identified.

At Strategic Thought we have met scenarios where individual projects were not inclined to respond promptly. As a consequence the middleware team tried to implement a capability that would best have been implemented in one of the underlying solutions — potentially compromising the architecture.

In other scenarios the middleware team must direct the individual systems. For instance they may require that static data is made consistent across the organization — to avoid transactions succeeding in one part of the flow but being found to be invalid in another and stalling in a third as a result of using consistent business rules but inconsistent data.

Architecture and design

It is vital that, before any development work starts, an architecture is produced. From this a set of lower level designs can be derived.

Such a design needs to concentrate on a middle-out approach which considers all of the following:

- **the business process to be implemented**
- **the routing and decision rules on which to base the paths of messages across the enterprise**
- **the message protocols — in effect how messages will flow through a system**
- **the message structure(s)**
- **the interfaces to external systems**
- **the transformation of messages across interfaces (to match protocols and the message structures)**
- **the maintenance of state**

- **the handling of exceptions, which always arise (exception handling is a separate business process in its own right — so go back to the top of the list, and start again)**
- **flexibility within processes, plus the ability to facilitate human intervention**
- **non–functional issues such as security (to address a range of issues such as privacy, integrity/tamper protection, denial of service, performance, business continuity and audit): all have to be considered and addressed.**

A measured approach which is structured around delivery of an appropriate solution — and which addressses the business need — will provide the return on investment.

Implementation approach

It is essential that a middleware project has a common and agreed starting point. To that end answers to the following questions are critical:

- **what is the business need that is to be addressed and has a clear business case been signed off?**
- **what is the investment needed to deliver the business case?**
- **can a realistic return on investment be identified and achieved, and what are the factors that might impact the business case positively (or negatively)?**
- **does a glossary of common terms exist which ensures that all participant use the same descriptions for the same 'things'?**

Identifying risks

Having described what a middleware project requires, let me now consider a schedule of risks that a middleware project might encounter.

If the project team does not have suitable experience of middleware, when problems are encountered it is common for people to fall back on their previous experience. There is a risk here that the result is a stovepipe application rather than a process oriented solution. The risk is that the potential benefits of middleware will not be obtained.

If there are insufficient design standards and methods then there is a risk that connecting flows at integration testing can be expected to fail and require additional design and re-work. Remember what happened with the Airbus A380 wiring.

In a team comprising a program manager and architect, even working in conjunction, there is a risk that it will not be possible to control the overall program of work. This is necessary to ensure that all comply with agreed design standards.

If the architecture and design are performed by a single person (or team) then there is a risk that the long term architecture will be compromised by the need to manage immediate detailed design issues.

If the interfaces between each component are not completely defined — and are not tightly managed under contractual change control — then there is a risk that ad-hoc change requests and uncontrolled changes will be made by individual teams. This may happen without regard to the knock on consequences for other teams — resulting in a continuous cycle of re-work, additional costs and time overruns.

If the systems to be interconnected are not yet complete (and particularly if there is any weakness in the design approach of any of these) then there is a significant risk of excessive change to interfaces. Expect a knock-on impact on middleware development costs.

If there is no central design authority then there is a risk that competing design authorities will not be able to agree on changes to interfaces in a timely fashion. This is a particular risk when delivering a solution that goes across organizational boundaries.

If excessive analysis and design is undertaken without testing through pilot implementations there is a risk that the approach will have to be modified to take into account factors only identified during implementation. There is an associated risk that confidence in the team is lost by the rest of the business.

If staff are brought onto a project before the work for them is properly specified there is a risk that they will start developing in advance of the final designs being provided and that they will not follow a middle-out approach. There is a further risk that the integrity of the design will be compromised by decisions that are made too quickly and without sufficient consideration — ones that may have to be reversed in the future.

Because middleware runs in the background there is often nothing to demonstrate. If there is nothing to demonstrate progress and the project is then delayed there is a distinct risk that it will be difficult to maintain the confidence of the project's business sponsor.

If the first middleware project taken on is difficult and it fails, the risk exists that this will discredit the adoption of middleware technology in the future. If justification of the cost of middleware technologies is placed on a single project there is a risk that this will present a significant obstacle to the commissioning of the project.

If an organization develops its own middleware technology there is a risk that the costs of this will discredit the use of externally-sourced middleware. This may obstruct the subsequent take up of industry standard products.

If a middleware project is not driven by a business requirement (supported by a return on investment case) there is a serious risk of over-engineering a middleware solution. This can result in the satisfaction of higher quality of service objectives than are required for such a first project.

If there is not strong project sponsorship at a sufficiently senior level then there is a risk that there will not be sufficient commitment from all functional departments affected. If there is no mechanism for measuring the benefits of deploying middleware then there is a risk that it will be difficult to make future business cases for enhancing the middleware and extending it across the business.

If there is no architect taking overall responsibility for the long term planning and approach then there is a risk of anarchy as individual projects seek to implement their own chosen middleware technologies in an uncoordinated fashion. This will result in significant subsequent costs to re-engineer solutions.

If there is no common glossary or agreed taxonomy of terms for the project then there is a significant risk of misunderstanding arising:

- **between teams**
- **across functional departments**
- **between partners**

for those intending to communicate via middleware.

If any one project introduces too many new elements then there is a risk that the project team will not be able to manage the additional complexity and that the project will be late or of reduced quality. Three new capabilities in any one project is probably a reasonable limit.

If an early project is undertaken that questions a team's reputation there is a risk that team confidence internally is reduced. Trust from the business can be damaged — thereby stalling future phases.

If the development methods and standards are not thought through and proven there is a risk that the middleware approach will not be scalable. This can manifest itself in multiple ways — including single threading, protocol bottlenecks and deadlocking when running under load.

When the solution is moved from development to production there is a risk that apparently minor differences in environments can introduce failures. A classic 'error' is relying on an implicit message order — which may not be valid in alternative configurations.

If the technical architecture of the middleware solution is tightly coupled and it is difficult to change one component without a major impact on the others then there is a risk that the progress of upgrading the environment will be both difficult and expensive. Products may also go out of support, thereby incurring higher maintenance costs.

If a regression test environment and associated test harnesses are not developed up front as part of the standard development approach there is a risk that it will not be possible economically to regression test the new environment. This is likely to result in an unacceptable level of bugs being introduced into production.

If the design approach does not rely on sub-components that can be engineered into top level flows there is a risk the solution will be brittle and vulnerable to introducing errors or incurring excessive development effort when being changed. There is also a risk that it will not be able to react to business needs.

If the order of delivery is not managed and, in particular, if it is delivered asynchronously there is a risk that causal effects can result in updates being delivered and processed out of order resulting in lost updates.

If error detection and management have not been considered then there is a risk that messages might fail yet be held in the middleware environment. This can 'suspend' business processes, with sales and revenue effects being felt. There may not even be the right tools in place to restart them.

Middleware solutions enable the business to route work dynamically to those able and qualified to carry out that work. If the number of specialists is reduced to too low a level there is a risk that the business will not cope with a return to manual mechanisms in a crisis.

If middleware exposes the business to electronic transactions with partners there is a risk that, if these fail or if the

business accepts orders that cannot be delivered, there will be a significant impact on the business' reputation. Similarly, if electronic business has insufficient security mechanisms and there is a loss of key data through a system failure or fraud and the business cannot demonstrate that it has proper controls in place then there is a risk that the business will expose itself to legal redress.

If an organization does not implement appropriate change mechanisms then there is a risk that the project will be obstructed by those who are affected by the change in organizational structures when moving from a functional hierarchy to a business process approach.

There is a risk that business requirements will change and that an organization will not be able to respond to those requirements. The need may be to have a capability to upgrade infrastructure at least (say) every three months.

If appropriate BC/DR approach is not designed and implemented there is a risk that it may be impossible to take and store backups and change logs and then recover the enterprise systems to a known and consistent state.

If middleware is particularly reliable then there is a risk that the operational expertise will gradually be eroded by time. If middleware skills are not maintained or practised, it will be difficult to resolve any issues when they do arise.

If a traditional capital model is used to finance implementing middleware solutions then there is a risk that middleware expenditure will not be consistent with capital expenditure rules. Middleware costs should be written off immediately even if they impact immediate business profits.

So what are the key risks?

Having processed all of the above, it is possible to group the schedule of risks into the following high level risks:

- **if you do not organize yourselves effectively there is a risk that you will not be clear about you want, that you will not obtain value for money from what is delivered and that the solution will not be sufficiently flexible to support the business into the future**
- **if you do not provide effective project sponsorship, governance, management and technical architecture resources to a middleware project there is a risk that the cost and time to deliver will be extend and extend — and result in a failure to achieve the business case**
- **if you do not work within the structure of a phased approach to implementation of middleware systems then there is a risk that the implementation team will lose the confidence of business sponsors and/or will not be able to evolve or learn from the experience of first phases; in addition it may not be possible to react quickly enough to changing business requirements**
- **if you do not take into account the special factors that apply to middleware projects there is a risk that these will fail to deliver the middleware benefits to the business which implement flexible and reliable automated business processes across the business (and the efficiencies that arise from these)**
- **once installed and working, there is a risk that any significant failure of the middleware systems will have a major detrimental impact on the business**
- **as the exploitation of middleware systems extends across the boundaries of the organization there is a risk that reputations can be damaged in the event of a serious failure of the middleware/system (or from a breach in security) resulting from failing to implement the necessary protection mechanisms for the overall solution**
- **as the use of middleware becomes a core enabler of the organization there is a risk that the balance of requirements against cost benefits will change, resulting in 'gold plating' and with the costs of operational management escalating.**

Opportunities

Yes, organizations face a wide range of risks when they implement middleware solutions. But they also create significant opportunities for the business.

Rome was not built in a day. The essential aspect of any middleware implementation is to develop:

- **an architecture**
- **an implementation road map**
- **an organizational structure**

that will enable the organization's business objectives to be achieved in the required timeframe.

If an organization is sufficiently capable, then taking on increased levels of risk to achieve competitive advantage

and then exploit that advantage in a timely manner is an excellent strategy.

Managing risks

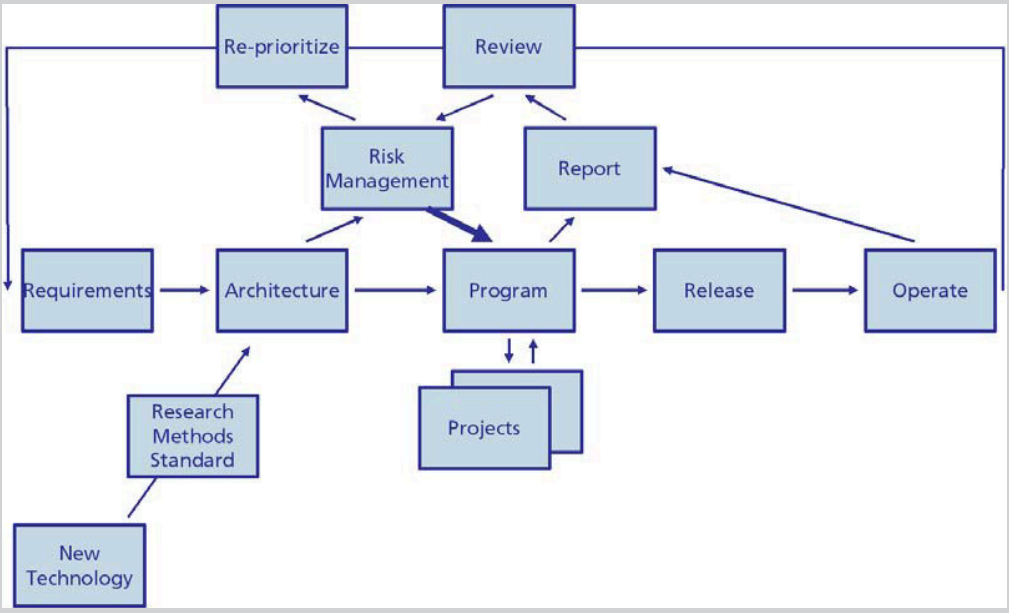
Managing IT risk (Figure 5.1) is no different from managing any other risk. The following need consideration:

- **having identified the potential risk, what do you need to do to understand fully the nature and the scope of the risk together with the potential impact on a project?**
- **what contingency planning can you do to deal with the impact of this risk if it occurs?**
- **what mitigation activities can you undertake in advance to reduce the probability of the risk occurring and minimize the impact of the risk if it occurs?**
- **what project, commercial and contractual protocols and agreements need to be put in place to encourage effective and timely risk management, including provision to transfer risk (and reward) to the party most able to manage it?**
- **what management framework and associated business processes do you need to put in place to assign responsibility to manage, monitor, control, report, escalate and resolve tisks during the course of a middleware project?**

The generic approaches for managing IT risk apply equally well to middleware as to any other IT project:

- **ensure that the responsibilities for risk management are assigned to staff trained, experienced and with the authority to manage them**
- **focus on the architecture and design, and identify the points of weakness, uncertainty or dependency on any unproven technology**
- **discriminate between a time risk and a killer risk (or 'show stopper'); if the killer risk occurs the project cannot proceed as specified, as opposed to an 'ordinary' risk which has a solution and the only uncertainty is the time required to resolve it**
- **do the difficult first, not what is easy: the easy might provide quick wins but may need revision if the resolution of the difficult risk identifies that an alternative strategy is required**
- **identify any aspects of the solution on which the approach is completely dependent and modify the design to introduce alternative approaches as contingency or fall back plans**
- **include additional features within the solution to provide a mitigation plan for any failures that occur so that the solution has strength in depth; if one defense is breached there should always be an alternative line of defense**

Figure 5.1: Managing IT risk



- **never include in a design any technology that has not yet been proven in the organization**
- **ensure that standards incorporate good practices**
- **implement a phased approach to delivery — proving the technology model all the way though to production — in order to identify any challenges as early as possible**
- **ensure that good work once completed is looked after, for instance through change control procedures and configuration management tools so that the good work put into the foundations is not destroyed by careless implementation.**

Management conclusion

From Mr. Denning's perspective, middleware projects introduce considerable levels of risk when compared with traditional IT projects. This difference occurs as a result of a number of factors. Few of these factors are fundamentally different from the challenges facing IT projects over many years, though the impacts of them manifest themselves in new ways.

What remains true is that because the problem being addressed is bigger the impact of a failure is greater. However, the opportunities arising from successful implementation, deployment and operational management of middleware solutions provide the scope to create highly significant returns on investment because, for a relatively small cost, they can leverage substantial historical investments.

Alternatively, by clearing specific expensive obstacles to the way that one does business, organizations can facilitate large reductions in transaction costs of doing business. The proof is in the eating. One Strategic Thought client achieved a positive return on investment from a US$60M middleware project in as little as 18 months.