Bloor Research believes that Resolution Accelerator provides a service that is essential to the successful widespread adoption of SOA

Peter Abrahams
Vitria Exception Management and Resolution

Introduction

Bloor Research is convinced that Service Oriented Architecture (SOA) is becoming the dominant IT model. Our certainty is based on the forces driving the implementation of SOA solutions:

- The business imperatives of flexibility and agility.
- The IT need to be more responsive to the business by implementing continuous improvement with minimal impact on existing systems.
- The delivery of standards and technology that effectively support SOA.

The heart of SOA is the loose coupling of services: this brings about the major benefits of SOA, including flexibility, agility and reuse. However, as SOA implementations have matured and grown in both scale and complexity, the following significant challenges have already become apparent:

- Managing exceptions thrown up by the business processes in a scaleable, auditable and robust fashion.
- Monitoring the health of the SOA infrastructure.
- Resolving errors in the infrastructure in a timely and effective way.

Bloor Research believes that Vitria’s Resolution Accelerator, combined with SOA governance products from Vitria’s partner AmberPoint and other vendors, provides robust, effective and straightforward answers to these major challenges.

The rest of this report expands on these conclusions by:

- Reviewing why SOA is essential to modern solutions and what the elements that make up an SOA solution should be.
- Explaining how exception management needs to work in an SOA environment.
- Showing how Vitria Resolution Accelerator implements exception management and resolution.
- Discussing the synergy between SOA governance and Resolution Accelerator.
- Providing examples of customer success stories.
Many companies have implemented Service Oriented Architecture (SOA) or are planning to do so. Vendors, including Vitria, are concentrating much of their effort in developing solutions in support of SOA.

This move to SOA is being driven by business and IT imperatives that are incapable of being resolved with traditional technology; these include:

- To remain competitive, businesses need to continually improve their flexibility and agility so that they can react quickly to business opportunities and competitive threats. This implies a need to be able to continuously change and improve their IT systems. This cannot be done using the older method of rip-and-replace, as this does not allow continuous incremental change. There is a need to change, or replace, small parts of the system without impacting the rest.

- Businesses need to be able to develop new business processes quickly to take advantage of changes in the environment. This can only be done if the new processes can be configured largely from existing generic services and technology.

- Businesses need to be able to outsource the provisioning of certain business services to third parties. The IT support for these services will move with the outsourcing and this move needs to happen with as little disruption to the remaining in-house systems and services. As business dynamics change it may become advantageous to bring some services back in-house; the advantage will be greater if the re-integration is straightforward.

- Mergers and acquisitions normally require the speedy integration of the new company; this can best be done by the speedy integration of the IT systems of the two parties.

- Third-party vendors are developing best-of-breed solutions for particular services. These are being built using SOA principles because that is the most effective way to offer them. An enterprise needs to implement SOA to take advantage of these new superior solutions.

- IT departments can see that SOA can improve their productivity by simplifying change and encouraging reuse.

- Business and IT need to be able to respond quickly to changes in business volumes, SOA enables extra resources to be allocated to a service easily and without impacting on other services.

SOA is the only IT model that provides the flexible framework to provide all of these benefits.
This section reviews the basic elements of SOA and the supporting functionality that is required to make SOA work in practice.

The basic ecosystem

Figure 1 shows basic elements that make up an SOA solution.

The services are exposed by IT and have SOA interfaces. Such services have well defined behaviors that can be invoked through a standards-based interface. If SOA is to be effective the services must be small and self-contained, they must have very well defined messages and interfaces and they cannot assume very much about their environment. They should not know what invoked them, and they definitely should not assume anything about the functionality of the invoker. Any services not built like this will be very difficult to test, very vulnerable to changes in the environment and will not provide the flexibility, adaptability and productivity that SOA can provide.

Services can invoke other services to create more complex services; but the new service still needs to have a limited set of well-defined behaviors and interfaces. The SOA model allows, and in fact encourages, the use of one service by multiple invokers, as can be seen in Figure 1.

The services are all loosely coupled and are only bound at run-time. This means that the implementation of a service can be altered at any time as long as the interfaces and behaviors are preserved.

Services can be developed from scratch using any modern programming language such as Java or C#. However, many services will be developed by third parties and brought into an enterprise. Finally, many packaged applications and legacy applications are being repurposed by wrapping all or part of their functionality in services.

The ability to re-implement services and to source services in many different ways makes possible a much more fluid implementation environment that allows continuous incremental changes without adversely impacting the rest of the environment.

These generic services can then be used by various organizations for their specific purposes. The higher level services can then be combined into business processes with each step invoking a service.

Supporting functionality

The three elements above, which address the business functionality, might be sufficient if the environment never changed and never failed. This is obviously not realistic so the SOA ecosystem requires a further set of elements to facilitate change, governance, audit, exception management and monitoring.

The first requirement is for a robust and easy to use development environment. This should include a studio that provides an intuitive interface for the IT architects, developers and, crucially, for the business users and analysts. The studio will enable discovery of services and the creation of services and business processes. Eclipse has emerged as a popular foundation for this development studio, with many vendors offering specialized plug-ins to extend its capabilities.

Underlying the development process will be a registry/repository (usually based on the UDDI
standard) that can store the definition and documentation of all the artifacts. This will be used to:

- Help the discovery process.
- Allow the viewing of artifacts as they are being developed.
- Facilitate the promotion to production.
- Enable the viewing of existing elements for auditing and governance purposes.

Once artifacts have gone into production they individually, and as a whole, need to be monitored. The monitoring needs to include availability, performance, security and also should be able to signal any variation from service level agreements (SLA).

**SOA creates some new challenges of its own**

Some traditional monolithic applications include integrated capabilities for handling at least some types of errors and exceptions within their own application domain. For example, an application suite might respond to a missing input value by flagging an order for special attention, assembling all relevant data within the application, and passing everything to the relevant application module that can address the exception.

However, after a company adopts a loosely coupled architecture the number of separate software components can increase significantly. Moreover, many of these elements may be developed at different times by different people, be implemented with software from different vendors, and be under the control of different organizations within the enterprise.

This distributed, heterogeneous nature of service-based applications makes monitoring and inter-service error handling a significant challenge. It is unlikely that a service includes any consistent internal behavior related to monitoring, so consistent external monitoring is needed to gather all relevant information independently of the individual services.

To provide a complete picture the monitoring must be implemented independently of the services and structure under observation. It will interrogate the repositories to discover what it has to monitor and how it can gather the required information. There can be many levels of activity to monitor and well-established tools for that purpose—technical monitoring tools that watch systems, networks, programs and log files. In addition, vendors like AmberPoint offer specialized tools for web services monitoring and governance.

But, as many companies are now discovering, the loose coupling that enables the benefits of SOA also prevents tools that monitor lower level activities from reporting the higher-level business context of an error. This has profound consequences for exception management.

If a service discovers a problem; it cannot fix it and can only raise an exception in its response to the invoker. The response must be a message or document that identifies the exception and provides all the relevant information that the service has collected. The problem with this approach is that often the invoker is also just a service so it, too, cannot resolve the issue.

A similar problem exists when a monitoring facility signals an exception. The detection of an exception may be directly caused by a failure in a business application or its supporting technical infrastructure, but the root cause may in fact lie earlier in the process. This means that the monitoring function cannot hand the exception to any obvious owner to handle.

Luckily, SOA not only creates this problem; it also provides a more elegant mechanism for resolving it than is possible with the integrated application suites. The solution is to have specialized processes for resolving exceptions. The beauty of this approach is that the process to deal with a specific exception can be built up by configuring general-purpose services.

The details of SOA exception management and Vitria’s solution are explored in the following chapters.
The previous section discussed how exception management fits into the overall SOA ecosystem. At this point the reader might well ask ‘why spend so much time on exception resolution? Why not spend it on preventing exceptions altogether?’

Removing the root cause of exceptions is obviously important, so investing in preventive measures like data validation at the point of entry into a process is usually justified. However, there are many technical and business-related reasons why exceptions are still so common in real world processes.

- System-level errors are difficult to eliminate completely for a variety of reasons that most IT professionals understand all too well. (Examples: network problems, system availability issues, data corruption).

- Rapid business cycles don’t always allow time for thorough testing of new processes at a business or technology level. When a new process goes into production, it should be expected that some types of problems will not have surfaced in testing and will need to be managed as exceptions in production.

- Data quality is lower when separate organizations initiate, versus process, transactions. This problem is especially severe when separate companies are involved. (Examples: channel partners submit batched orders from their own customers, or some steps of a process are externally outsourced).

- The complexity of many products and processes leads to user misunderstandings about acceptable transaction configurations, and full validation logic may be too complex or volatile to embed in front-end systems. Consequently, problems may only be detected by backend systems (Example: order for incompatible products or services is first flagged by an order fulfillment or provisioning system).

- The process context within a company can create exceptions. (Example: ordered part is out of stock).

- External situations outside the control of the enterprise create exceptions (Example: severe weather conditions delay a shipment).

- Process changes upstream or downstream can cause many types of mismatch-related failures. These are often difficult to avoid, as many processes in modern systems are not aware of upstream users, particularly those in different organizations.

- After-the-fact changes to transactions requested by customers or suppliers are not only exceptions in their own right, but in a highly integrated and time-dependent chain, can also have ripple effects on other processes and hence generate further exceptions.

- Non-standard requests will always occur, either to meet a one off situation or as the first example of a new request type.

Even with a large investment in time, effort and technology, exceptions still occur frequently in the modern enterprise. Exception rates typically vary between 2–10% of transactions in mature, established, scalable processes, and 20–80% for new and rapidly changing processes.

Analyzing, automating and improving the exception management process offers significant benefits in terms of costs, revenues, and other factors. Cost savings result from reducing both the number of exceptions and the labor required to resolve them. In order management systems, for example, it’s not unusual for 50–70% of associated direct labor costs to be for exception processing. Less obvious but also important are indirect cost savings from avoiding unnecessary downstream activities, such as dispatching field personnel for onsite service requests that, unknown to them, had already been cancelled. All such reductions of cost add directly to the bottom line.

On the revenue side, eliminating processing delays caused by exceptions can reduce revenue leakage from customers who will take their business elsewhere if they wait too long. This is particularly true in highly competitive markets where customers expect immediate gratification and can easily switch to competitors.

Figure 2 shows the exception management and resolution processes.
Exception resolution

Most transaction flowing through the "normal" business process, seen at the top of Figure 2, will complete successfully without any errors or problems.

However, exceptions cannot be completely eliminated for the many reasons discussed in the previous section.

When exceptions do occur, there are a number of products that can help manage them, mainly by creating audit trails and reports that give the assigned staff the raw data needed to resolve each problem offline.

But the greatest benefits come from solutions that not only track data for managing exceptions, but also automate all or most of the exception resolution process. In high volume processes, or architectures with many levels of loosely coupled services, automated exception resolution may be the only scaleable way to deal with these problems.

This approach initiates a separate process to resolve each detected exception and includes its own series of activities, decisions, data and actions. The process goes through the following steps: discovery, classification, enrichment, routing, resolution (automatic and manual), restart and, potentially, external notification and action triggers.

Information is logged at all stages of resolution to enable subsequent auditing and process improvement.

Discovery
When a process discovers an exception, it first needs to gather all the relevant information that it is aware of including the identification of the process, the input transaction, the error message and any data gathered or created in processing the transaction so far. (Example: an incoming order is rejected by a CRM system because it is missing both the postal code in the shipping address and a preferred shipping option).

Classification
Based on the information collected the exception is classified. Classification categories will determine how enrichment and routing are carried out. (Example: the CRM error codes for missing postal code and missing shipping option are identified and understood).

Enrichment
To accelerate the resolution of an exception, relevant data may need to be retrieved from other systems. (Example: retrieve customer address data from past transactions in the same CRM system as well as in a separate billing system).

Routing
Defined rules for routing the exception need to be applied based on the type of the exception, the data collected during the enrichment, and other factors such as the client’s status or the transaction value which may effect who it is routed to and its priority. (Example: Route the missing postal code problem to an automated system for resolution, and route the missing shipping option problem to a call center employee for resolution).

Resolution—automatic
Many of the recurring exceptions can be resolved automatically. A set of rules can be created that attempts to resolve the exception. The rules may work in all cases or there may be a residual that still requires human intervention. (Example: get the missing postal code from any matching records in the CRM or billing system if available, otherwise invoke a web service to look up the postal code based on street address and city in the order. If that doesn’t work, send the problem to a call center employee to resolve).

Resolution—manual
Some exceptions inherently need human intervention to resolve. This might be because a subjective management decision is needed, information from person-to-person contact is required, information from a non-integrated system is needed, or the exception management system does not know how to classify a new type of exception. When human intervention is required, there is a need for a workflow system that can schedule, route, prioritize, monitor and escalate to ensure that the resolution occurs in a timely and effective fashion. (Example: a call center employee is prompted to call the customer contact listed on an order to get the missing information on shipping preference and, if still needed, the postal code).
Restart
When the resolution has been completed, the updated data on a transaction can be used to restart (or abort) the business process.
(Example: the repaired order with the correct postal code and shipping option is re-inserted back into the order management process).

External notifications and action triggers
Sometimes an exception or its associated resolution needs to trigger events outside the actual resolution process, such as sending advisory alerts or initiating external processes.
(Example: exception processing for an order affected by an out-of-stock part could automatically notify the relevant sales representative for that customer, even if no action is required).

Improving exception handling
Although having a well-managed and efficient exception resolution process is vital to a well-run and successful business, the really exciting and beneficial part of exception management in the long term is finding ways to:

- Prevent exceptions by improving the process to remove their root causes.
- Expedite the resolutions by improving the resolution processes themselves.

The key is to collect the relevant information. Without historical information on the number and type of exceptions that occurred and how they were ultimately resolved, it is extremely unlikely that the right process improvements will be made in a timely manner. Similarly, without specific “before and after” measurements it is very difficult to measure the effect of changes; so a central log of the exceptions and an end-to-end audit trail for the resolution process are critical to any improvement process.

The first analysis should just be a report of the current situation, so management can see if things are getting better or worse and if there are any specific teams, departments, processes or partners that appear to need assistance. If things look good, there may be no need for further analysis but in general some areas will be highlighted for attention. The low hanging fruit to look for is either a very common exception where the total resolution cost is high, or a high value transaction type where the removal of just a few problems could influence profitability and customer satisfaction.

Further analysis will be needed to understand why the exception occurs, which organization or process caused the exception and why the resolution process is difficult. Based on the analysis a decision can be made to:

- Eliminate the root cause by making a change in the upstream process.
- Improve the resolution process by further enrichment or by modifying the resolution rules.

Modifying exception processes must be done in a careful, controlled, and transparent manner so the responsible management can understand the changes, and verify that the new processes are working to specification. This suggests the need for:

- A central repository for all the rules driving the resolution process.
- A standard language describing the rules that can be understood by people and machines.
- A user interface for monitoring and reporting that can be viewed directly by management.
- Complete cradle-to-grave tracking of all applications of resolution logic to every exception processed, plus all changes to exception resolution logic itself.

The next chapter describes how exception management is implemented with Vitria Resolution Accelerator.
Building an exception management solution as a set of services, which can be initiated using standard SOA protocols, has enabled Vitria to produce a general purpose, configurable and scalable solution. It has also enabled Vitria to integrate with other SOA ecosystem solutions such as AmberPoint.

A simplified architecture is shown in Figure 3. At the top is the exception flow that was described earlier. Below that can be seen the Resolution Accelerator run-time environment supported by a variety of generalized services and the metadata that defines the individual processing.

**Run time**

The run time is a standard SOA service and can be accessed by any system or process that generates an exception, including the infrastructure monitoring facilities from AmberPoint.

When the exception is passed to the run time environment the metadata is accessed to determine what processing is required to resolve the exception.

**Metadata**

The metadata is the key to the flexibility of the solution. By using a structured “dictionary” of metadata instead of simply coding how the resolution process should work, the logic is kept in a form that is much easier to understand, implement, maintain, and modify later, particularly if multiple people are involved. The metadata defines and describes different types of exceptions, the rules governing selection of resolution processes, the processes themselves, and any documents referred to in manual resolution processes.

By building the resolution processes from standard rules and services, new exceptions can be defined quickly and accurately. As the analysis of past exceptions brings out ideas for elimination, automation or acceleration they can be speedily defined and implemented.

Having all the definitions stored in metadata means that an audit trail of changes is kept. This information can then be used when management, or the compliance officers, need to understand how exceptions were being handled during a particular period.

A high function user interface is provided for the entry and management of all types of metadata. It is designed to be used by managers who are responsible for the accuracy of the exception handling. The managers may delegate some of the detailed input but can still verify its accuracy by directly viewing the metadata.

Many industries, including telecommunications, healthcare, insurance, and banking, use specialized data vocabularies for information exchanged between systems and organizations. Metadata on these vocabularies can be important to exception detection, classification, and routing. Vitria offers several product options for specialized industry-specific vocabularies in telecommunications, healthcare, insurance, and other sectors.
Services

Resolution Accelerator has been developed based on the experience and requirements of Vitria customers across several industry sectors. These customers identified standard services including automatic resolution, human workflow and integration services.

Automatic resolution
The automatic resolution of an exception is likely to require a process with several steps and logic between the steps. Data may be required from several different systems and computation carried out on the information. Resolution Accelerator provides all the services required to implement these process flows.

Human workflow
To build a solution that assists human resolution requires a set of services that support human workflow. These include services that:

- Decide on the routing of an exception.
- Queue requests to individuals or teams.
- Determine the relevant information to display to the user.
- Prioritize individual transactions.
- Monitor the status of the queues.
- Escalate out-of-line situations.
- Accept and log changes.

Resolution Accelerator includes all these services and the metadata defines how they should be configured.

Integration
Exception management requires integration with the routines that pass exceptions to it; but it also needs to be able to access external systems, particularly in the enrichment process.

If the other systems support SOA directly then the integration is simple. If they are older systems then Vitria can supply integration technology that provides an SOA interface to Resolution Accelerator but will use whatever interface is available to connect to the other system.
As discussed earlier, SOA solutions are made up of loosely coupled services that are connected and orchestrated to support business processes (as depicted at the top of the above diagram). Non-trivial SOA solutions also require a governance mechanism for defining acceptable and unacceptable service behaviors, then monitoring and enforcing these policies at runtime. This governance function may also include monitoring performance against Service Level Agreements (SLAs) negotiated between the business side and the IT provider, and generating alerts when these agreements are violated or at risk.

Specialized SOA runtime governance products from AmberPoint, and other vendors, have emerged to enable policy-based management of service networks, including monitoring and control of services-based environments. These capabilities typically include:

- Maintaining information on services and service endpoints in a centralized repository, which includes or interoperates with one or more UDDI-compliant service registries.
- Defining and centrally managing policies for service execution, including those related to endpoint management, access control, related security issues, and business policies.
- Enforcing these policies in runtime via distributed agents that check all service invocations and their associated message contents.
- Tracking and reporting the status of all services and the requests they process. This includes detailed reporting on service execution failures caused by exceptions, including those resulting from policy violations.
- Monitoring and reporting performance against defined SLAs. This may include response times for individual services or higher-level processes, number of business transactions completed versus expected or required, and other key parameters.

With its broad capabilities for resolving exceptions (not just monitoring them), Resolution Accelerator strongly complements SOA runtime governance products.

Runtime governance agents are well suited to detect exceptions for Resolution Accelerator to resolve, in both initial service invocations and their associated responses. These agents can check the actual contents of messages as well as the service-related context.

Similarly, governance policies can include centralized exception-triggering logic beyond that in individual applications. One possible application of this approach is feeding Resolution Accelerator with exceptions that are not errors or failures, but instead are business opportunities worthy of special treatment (e.g., divert an order with certain characteristics to a call center representative who will try to up-sell the customer to a higher-value product).

Resolution Accelerator provides sophisticated resolution capabilities for policy violations that SOA governance products alone can only block from continuing. For example, a new user’s attempt to access a service might simply be blocked by a governance product if a new user had not yet been given the proper access rights. Resolution Accelerator, on the other hand, might automatically retrieve background data on the user from other systems, include this in a multi-step workflow process to determine if the user’s access rights should be changed, and modify the governance policy itself, if appropriate.

Governance products monitoring SLA compliance can use Resolution Accelerator to manage the response to early indications that an SLA is at risk of violation, or to actual violations themselves. For example, reaching 90% of allowable service unavailability halfway through a defined time period could trigger Resolution Accelerator to manage a process to add more resources to share the processing load.

Together, SOA governance products and Resolution Accelerator provide a more complete data set for root cause analysis of exceptions than either could by itself. For example, the former can provide more insight into where, in multiple layers of service invocations, an exception initially appeared; while Resolution Accelerator can extract relevant process context data from sources outside the narrow scope of the affected service. This enables more complete root cause analysis, which in turn leads to better process enhancements to prevent future exceptions.

This combination of SOA monitoring and exception resolution capabilities thus provides a well-integrated and robust solution to governance of SOA systems.
Resolution Accelerator has been selected by Vitria customers in several sectors in North America, South America, and Europe. Here are some examples.

- A large telecommunications company in South America is using Resolution Accelerator to manage exceptions in its order management process for wireless services. 85% of exceptions that had previously been resolved manually are now resolved automatically, and reduced customer turnover from the dramatic reduction in exception-related delays is expected to increase revenues over $2 million per year. In addition, exception-related labor costs have been reduced by 60%.

- A rapidly growing digital entertainment company in the United States is using Resolution Accelerator in its order processes and CRM operations. A key objective attained for this project was maintaining high customer satisfaction with the service activation experience and preventing delays even when order volumes reach unusually high peak levels.

- A manufacturing company in Europe selected Resolution Accelerator to resolve exceptions in its operational support systems and electronic interactions with supply chain partners.

- A European telecommunications services provider selected Resolution Accelerator to resolve exceptions from its process for clearing and settling wireless roaming charges with other companies.

- A local telecommunications company in the United States implemented Resolution Accelerator to centralize the management of exceptions from multiple application services in its order management and provisioning processes.

- A large financial services company in the United States selected Resolution Accelerator as part of its SOA-driven integration solution for processing insurance-related transactions.
The benefits of Vitria Resolution Accelerator for exception management are multiple:

- Automated exception resolution and context-sensitive guided workflow can greatly reduce delays and manual effort caused by exceptions in business processes. This in turn can increase revenues and reduce operational costs. These benefits are particularly relevant to SOA projects, which are more vulnerable to exceptions because services are loosely coupled.

- Centralized logging of exceptions and their resolutions can improve the auditing, monitoring and control of exceptions and the overall business process itself. This logging is essential to ensure business compliance, as ad hoc fixing of exceptions is a real opening for unauthorized changes to occur.

- Changes to how specific exceptions should be resolved can be implemented quickly by modifying Resolution Accelerator’s metadata dictionary and handlers. This can significantly improve business productivity by reducing the number of errors that have to be resolved by human intervention.

- Exceptions can be handled in a variety of different ways, all under the control of a generalized service. In order of increasing value, potential approached include:
  - Just passing a problem to a human for resolution.
  - Guided workflow where the system suggests possible resolutions to a human user.
  - Automation where the system fixes the problem without needing any human intervention.
  - Prevention where a system can catch and require correction of errors at the point of entry into a process before the service raises an exception.

- As the resolution life-cycle of different types of exceptions becomes better understood, the action enabled by Resolution Accelerator can evolve through the four approaches above. This can be done for one exception type at a time so the low hanging fruit with the biggest benefits can be tackled first without compromising the processing of more unusual errors.

- Institutionalization of Best Practices—what used to be “tribal knowledge” of how to deal with an exception now can be captured in a well-defined “service” that can be easily modified and adapted. Such well-documented practices are an essential prerequisite to meeting compliance regulations.

- Resolution Accelerator can offer even more value when combined with SOA governance and monitoring products from AmberPoint and other vendors.

The beauty of SOA is that it enables solutions like this to be developed and provided independently as services, thus providing all the benefits at a fraction of the cost of trying to build a bespoke exception handler.

Bloor Research believes that Resolution Accelerator provides a service that is essential to the successful widespread adoption of SOA and should be considered very seriously by anyone moving into this area.

About Vitria

Vitria Technology, Inc., has been providing award-winning business process integration products and solutions for over 12 years. They combine technology leadership in SOA and event-driven integration with expertise in business process management in telecommunications, manufacturing and supply chain, healthcare and insurance, and financial services.

Vitria operates in North and South America, Europe, Asia, and Australia.
Bloor Research overview

Bloor Research has spent the last decade developing what is recognised as Europe’s leading independent IT research organisation. With its core research activities underpinning a range of services, from research and consulting to events and publishing, Bloor Research is committed to turning knowledge into client value across all of its products and engagements. Our objectives are:

- Save clients’ time by providing comparison and analysis that is clear and succinct.
- Update clients’ expertise, enabling them to have a clear understanding of IT issues and facts and validate existing technology strategies.
- Bring an independent perspective, minimising the inherent risks of product selection and decision-making.
- Communicate our visionary perspective of the future of IT.

Founded in 1989, Bloor Research is one of the world’s leading IT research, analysis and consultancy organisations—distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services and consultancy projects.

About the author

Peter Abrahams

Peter started in IT as a sandwich student 36 years ago. He worked for IBM continuously through that period. In a company known especially for its hardware Peter saw the importance of software and especially transactional processing. He installed the first IMS on-line system in the UK and early versions of DB2.

His role as both pre and post sales technical support involved him in the design, implementation and integration of complex systems in many of the major banks, retailers and manufacturers in the UK, Israel and Europe.

He spent three years representing IBM software to the analyst community which is when he first met Robin Bloor.

Most recently he was a Consultant Architect in IBM’s Financial Markets Practice. His speciality was STP, which enables financial institutions to connect diverse systems within their own organisation and in their partner’s to process complex transactions without human intervention. This experience reinforced his belief in the need for continuous evolutionary change of application systems and therefore the necessity of an intelligent infrastructure to bind them together in a robust but flexible way.
This document is subject to copyright. No part of this publication may be reproduced by any method whatsoever without the prior consent of Bloor Research.

Due to the nature of this material, numerous hardware and software products have been mentioned by name. In the majority, if not all, of the cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not Bloor Research’s intent to claim these names or trademarks as our own.

Whilst every care has been taken in the preparation of this document to ensure that the information is correct, the publishers cannot accept responsibility for any errors or omissions.