



# Development intelligence: business intelligence for software development

Bola Rotibi

April 2007



# Development intelligence: business intelligence for software development

*There is currently pressure within organisations to apply quality management, metrics, intelligence and insight to the delivery and management of IT solutions. Why? Because, while IT and the supporting technologies have come a long way both in maturity and sophistication in enabling businesses to achieve and connect more with their customers, supply chains and partners, its delivery, management and the value generated have been too inconsistent for its role as a well managed and integral business process. This paper looks at how intelligence gathering within the software development and delivery process can help in the task of making IT more of a managed and reliable business process. We put forward the concept of development intelligence as a process and framework for business intelligence for software delivery. Our paper offers a guidance framework for helping organisations to address the importance of obtaining better development intelligence as part of an overall strategy towards improved management and delivery of IT solutions, and a fully integrated quality management and business intelligence framework.*

## **A lack of intelligence... the reality behind IT failure**

With the increase in packaged applications from office suites to CRM, it must seem odd to a non-IT person that there is still so much in-house software development. This is because software packages are targeted at wide markets. For a business to gain a commercial advantage from a software package, it needs to be customised for the way that they work.

The problem is not the need to customise software or build new software applications from scratch; it is that each software project is like a separate journey of learning. Despite the vast sums of money spent on methodologies and best practices, very little analysis of a project takes place, and that means that lessons are not learned.

Software development needs to be reliable and deliver quality solutions. Ask users about reliability and quality of software and the response will be more negative than positive.

To turn this around successfully and reverse the user perception, greater insight and intelligence into the IT organisation and its processes is required. The same is also true for the IT organisation if it is to improve



the quality of its IT solutions, ensure that they are better equipped to meet the needs of the business, and optimise the delivery process so that they can react more quickly, confidently and predictably.

While a large amount of information is created, very little of it is stored and presented in a useful way. The IT organisation, particularly the delivery team, lack accurate information on all steps involved in the software process, and without that they are going to continue to struggle to improve.

### **Something old... something borrowed... something BI**

The notion of development intelligence and the concepts behind it are not new. Understanding what works and what doesn't is the basis of most intelligence solutions. Getting access to the information in a clear and consistent way can be a challenge but it is not insurmountable. Take the example of business intelligence (BI).

One of the most important things that BI has given to organisations is the ability to remove the artificial, inter- and intra-departmental silos that surround data. With users able to search for and analyse data across the entire business, or as much as security allows, correlations between departments and business processes are being exploited for commercial and financial gain.

Today, BI is talked about as part of mainstream computing, with vendors competing to put BI tools onto the desktop of every user.

The development organisation is much further behind in applying business intelligence to its processes, failing more often than not to provide answers such as the ability to determine an application's state of readiness, what projects are at risk, how to optimise development processes, and conformance to SLAs for offshore/outsourced development.

We consider *development intelligence* to be a feature of the overall business intelligence framework that should exist inside organisations and which promotes the most effective and beneficial relationships between customers, suppliers and an organisation's internal workforce. BI uses tools to aggregate and correlate data from multiple sources. This allows very specific questions to be asked of the data and patterns identified. As development intelligence (DI) is a subset of BI, the same environment and criteria apply to the concepts behind its application (for example, build systems, SCM systems and various testing tools may each represent separate data silos that need to be leveraged). Putting in place a BI-based framework into the software delivery process should not be hard. Many within the IT organisation are already keen to welcome such a strategy. However, to be successful, a DI strategy will also require a cultural shift, as there will be a need to embrace change and a willingness to learn from previous projects and results on a continuous basis.



## Development intelligence explored

### Defining development intelligence

There are many facets to the definition of development intelligence.

Therefore, by definition:

*development intelligence is the collection, aggregation and analysis of all data generated by the software delivery process and the means by which the IT delivery team is able to continually identify the strengths and weaknesses of the process early, in order to improve the assessment, management, implementation and quality of existing and future software projects.*

Development intelligence is:

- a means of eliciting objective information about the health and status of development projects to support management decisions
- a way to provide early visibility and trend data so that one is able to evaluate risk and steer the process in the right way to ensure quality and the effectiveness of the solution delivered (or, alternatively, reduce wasted cost and effort by cancelling unrecoverable projects early)
- a way to gather information to identify skills gaps and resource dependencies or shortages, allowing project planning teams to understand what is at their disposal and where there is risk, especially when teams are distributed
- dynamic and continuous, reducing the variances by providing more accurate data in a faster timeframe, allowing for better fine-tuning
- a feature of an overall process control framework that needs to be in place to better manage the overall quality of the application lifecycle
- not just about the process, it is also about better managing the dynamics of the IT organisation so that it is able to react quickly to the changes and regulations placed on it.

All of this means that DI requires the collaboration of the key roles involved and seamless integration of knowledge from the tasks and functions carried out by the key roles responsible for the delivery, deployment and operation of software applications – whether they are custom-built, packaged or externally sourced.

### The importance of development intelligence

***Better software, improved risk assessment, more timely intervention, lower cost.***

Software delivery without intelligence is a little like blundering around in the dark. It's hard to see any other industry taking such a cavalier



approach to its products or customers accepting it. Banks and loans, governments and macro-economic policy, drug companies and new products, supermarkets and product placement are all examples of how large datasets are captured, queried and used to make better ongoing business decisions.

Intelligence, and more importantly, historical intelligence and appropriate metrics, give a means to setting risk and predictability. The more trend analysis on a process the more predictable and reusable it becomes. Even when one enters a new project where the risks are not yet quantified, there is still a certain amount of protection that can be gleaned from historical experience and realtime assessment.

However, the software delivery process has a long way to go before it can claim to be responsive to realtime and historical data.

Quantifying the waste from different parts of the development process is difficult. There are very few studies that have taken place over the years to evaluate losses due to poor requirements analysis, incomplete software design and bad coding practices.

What research has been published has tended to be around the issue of testing, both from a quality perspective and an ongoing developer responsibility.

In June 2002, the US National Institute of Standards and Technology (NIST) said 'software bugs are so prevalent and detrimental that they cost the US economy an estimated \$59.5 billion annually, or about 0.6 percent of the gross domestic product'. It went on to say that 'although all errors cannot be removed, more than a third of these costs, or an estimated \$22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects'.

DI proposes an approach whereby success and failure can be respectively reinforced and reduced early in the delivery process. There is already a vast amount of information known about software projects, but it needs to be captured and turned into useful, actionable information.

## **Breaking down the IT silos for development intelligence**

For DI to move forward, companies need to rethink their entire approach to the information they have and how to extract the intelligence from the silos that exist in IT. Whilst breaking the silos is important, a bigger challenge is integrating both unstructured and structured data sources.

### **Development silos**

Every software project can, and does, create vast amounts of data, but it is often ineffective because:



- the data is generated and held in isolated silos
- it is not always saved for further analysis and comparison
- there is no automated process for collection thereby preventing errors, purposeful omission or misinterpretation.

We consider five main silos covering software development.

- Analysis and business requirements are rarely stored in electronic form until a specification is drawn up and passed to the architect.
- Application architecture is increasingly done through design tools. This information is starting to be stored in repositories, but this is not a common practice.
- The majority of developers use Source Code Control Systems (SCCS). Not only is code stored here, but it is generally linked to some form of bug-tracking solution.
- QA and test data is split between unit testing and static code analysis carried out by developers and formal tests done through a QA process. In disciplined processes, tests are built from information gleaned from the analysis and architectural phases. Data is then shared between developers and the QA team through the use of a common repository. However, more often than not, when data is generated from unit tests and static code analysis, it is either not saved or saved in a manner that is not very useful or manageable, resulting in additional data silos that provide little opportunity for shareable analysis and comparison. In some cases the sharing of information is actively discouraged to hide embarrassing results or bad practices (test failures and poor coverage).
- Operations and deployment is a post development role. There is little integration between the two teams with the exception of bug reporting.

### **Impacting factors**

Sitting alongside these silos are other influencers.

- Compliance has started to have a significant impact on software development. The problem is that it is focused on process not on quality. Tools for developing compliance needs are in short supply and those that do exist tend to be more rules-based solutions. This makes it hard to directly apply to software development.
- Security has always been an issue with software development. It can be hard enough aligning current security practice to ongoing development but few companies, including software vendors, have the budget to deal with existing code. This means that security alerts or other discovered vulnerabilities, particularly those from third parties, are rarely, if ever, tracked in existing applications until that code is revisited for new features.



- Project Management tends to hold a lot of data about a project – initial timescales, resources, what was changed and why it was changed. Little of that information is ever used to identify skills shortages within a software development organisation or to see what resources are underused.

## Putting in place a viable DI framework

To really change the way we deliver data we need to begin collecting and considering information at the start of the process. Much of the information we generate today is 'after the fact' and is not fed back into the next project, as well as making it almost impossible to capitalise on it effectively for existing projects. The end goal with DI is not just to be able to learn from a project or to use historical data to see how effective QA tests were, but to create and constantly update an institutional knowledge base that allows the delivery team to discover trends, issues and risks early and act on them at a time when costs can be minimised and scope and resources optimised.

Intelligence can be taken from the data or processes we already have. The most pressing need is for some way to capture and extract meaningful data from raw information and ongoing activities. Some of that will be in existing repositories but much will be unstructured, un-indexed data (often hiding in email or lost among the hard drives of various developer workstations). The latter provides a serious challenge, not only in gathering and preparing it, but also in making sure it exists at all. A lot of key data, such as the results of unit tests, code coverage and metrics, may not even persist beyond the initial generation of the report. It cannot be incumbent on individuals to keep and manage this data; data capture and persistence must be integrated into the development process itself.

Using data marts we can take data from the unstructured and structured sources, transform it and then insert it into something against which we can query. This is one area where BI provides the foundation for development intelligence. The other is in taking advantage of its data extraction and transformation tools, where data can be extracted from other repositories and then fed into a data mart.

Another big challenge for a DI-driven environment is data capture and selection. This requires a co-ordinated effort to get existing files into a set of managed locations (central or federated) and a (preferably automated) process to ensure that all new files are stored there as they are created. The data selection will be more problematic: what files hold useful data; file formats and internal structure of the documents (for example, random text or structured XML templates); how to extract the data in a way that allows working with multiple data formats at both the file and content level. Metadata will be needed to make it easier to index files.



Resolving the data representation problem is crucial, although it may be possible to automate some of this through the transformation process. For structured data, this is not too onerous. However, as a significant amount of the data from the development process is likely to be unstructured, the data representation and transformation processes are going to be hard.

How hard? This is still a significant pain point for experience BI practitioners, so you should not expect this to be a quick solution in the DI space. Those organisations that do use standard templates will at least have a common structure to their content inside the files which will help ease the problem. Loading the data into the data mart against which queries will be run will be dependent on the success of the transformation process.

The key to making DI work is to have data from multiple projects that can be queried, and any attempt to bypass or short-change these steps will risk failure.

### **Quick returns, but long-term commitment is required**

As already said, DI is not an instant fix. Data will have to be gathered, tools will need to be developed and methodologies capable of consuming the results in order to provide a better solution will need to be created.

Despite this, there will be some short-term gains and quick returns that will improve the delivery of IT. Applying intelligence to metrics and data gathered even in a basic format will enable more reliable deployment, especially if the process of generating metrics is automated so metrics can be captured more consistently. Even comparing the results of existing tests with the SCM and defect tracking databases will show how often code is revisited. This gives an immediate snapshot of the state and quality of the code in the project, as well as providing insight into developer activity and skill levels. BI tools allow this sort of analysis to take place quickly and the results can help identify, for example, particular sets of code that are error-prone, unreliable, or expensive to maintain.

More long-term commitment into implementing a more comprehensive DI framework would add to the move towards better software security. For example, software bugs are often related to particular software libraries or APIs. Being able to track what software calls a library or API that has been identified as a security risk and then patching it is a business-critical issue. Few companies are capable of doing this today.

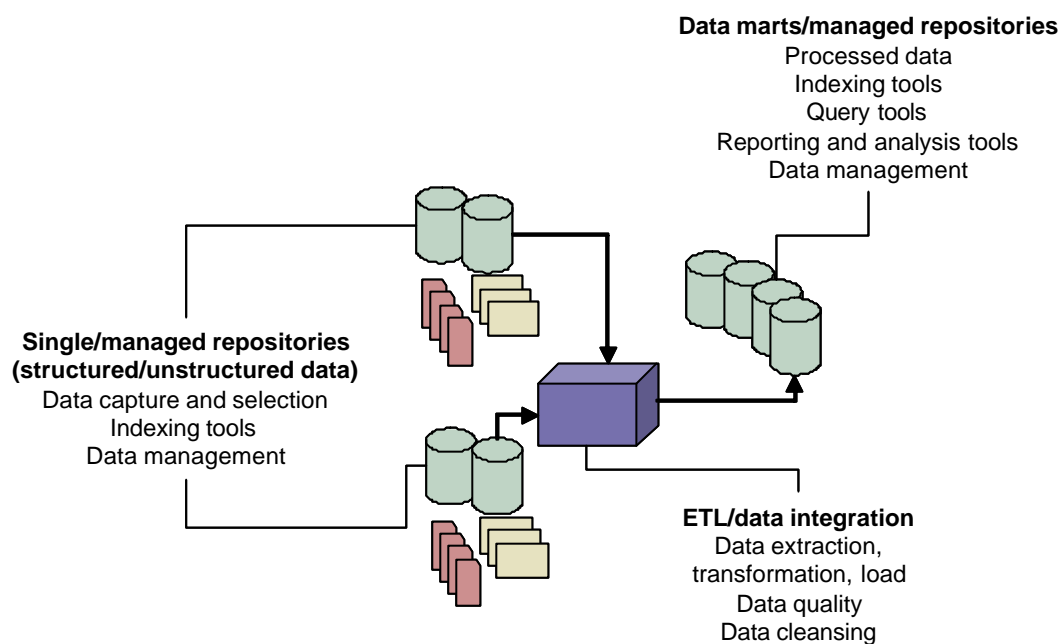
## **Development intelligence architecture**

DI architecture is based on weaving together a knowledge and query-based platform that elicits information from the functions and tasks carried out by the key roles responsible for the software delivery process. It also



needs to record the interactions, communications and relationships that occur between them. It is a quantitative approach that includes extracting data and mining information to assess the effectiveness of specific processes. As a result, the DI architecture is centred on a set of key functions depicted in *Figure 1*.

Figure 1 **Development intelligence architecture**



Source: Ovum

Some of the key components of a DI solution include:

- **repository:** with the ability to handle mixed content such as spreadsheets, word processing documents, configuration management, software deployment scripts, data centre configuration, use of external APIs and performance metrics. There are also indexing tools, query engine tools and possible data cleansing facilities, along with some of data management
- **metrics generation and data capture:** automated process to generate and capture data and results from a variety of development-related activities (for example, SCM systems, build and release management systems, test systems)
- **ETL/data integration:** extract, transform, load. Getting data from the repositories and into data marts will take some time. The right data will need to be identified, the transformation will need to be designed and the data mart capable of storing the data will need to be designed and



tested. BI teams took several years to understand the complexity of ETL, and DI practitioners must look closely at the lessons that BI teams had to learn. Also taking place is data quality and data cleansing

- **data mart:** while the repositories will hold the data, the analysis of trends and experiences will be done through data marts as they are better suited to data transformation and realtime query – a lesson learned by the BI community. Data marts are databases that contain snapshots of cleaned data against which intelligence queries are executed. They are focused sets of data drawn from various repositories providing common data structures so that results can be accurately compared and validated. In addition they have search, indexing and query engine tools to interrogate to find patterns and behaviours in order to enforce positive implementation practices and reduce detrimental actions. Data management will also be applied at this level.
- **reporting:** metrics, dashboard, historical trend data, predictive analysis, resource usage, skill analysis
- **culture:** the right attitude and commitment level to enforce usage and implementation of the right process along with the necessary incentives and penalties driven by management, such as policy enforcement processes and frameworks. You need buy-in support from all the key roles in the delivery process – something that is not always present in many IT teams today
- **methodology:** in the early stages of DI there is no clear methodology for implementation and use. This is not unusual with any technology introduction but DI needs something to provide a framework, not just for implementation but to ensure that data is captured and exposed as part of the software delivery cycle. To achieve this, whatever methodology is adopted, needs to be based around existing methodologies and best practices in this area.

Many methodologies are linear in focus and look at data on a project-by-project basis. Their 'after the event' analysis prevents improvements to the project once it has started. A new DI-based methodology may appear, or existing methodologies will adopt DI principles extending their reach and abilities.

Of the main methodologies and best practice approaches in the market, DI can both draw from what they do and offer an extension to improve them. ITIL/ITSM is a series of best practices that offer frameworks for managing across the entire IT service discipline. What they lack is the iterative analysis that DI offers. The same is true for COBIT, although its scope is more metrics focused than ITIL/ITSM. The metrics focus of COBIT offers a good model for how to build a metrics-based methodology, and DI could use that as a solid basis for the type of metrics it needs to be gathering. There is much to link 'agile practices' and DI, especially with



regard to the use of iterative and up-front analysis and exposing lessons learned at the start of any new project. As Agile is already embedded in a lot of organisations there should be a good basis for cementing DI practices.

A comprehensive DI framework should be able to act on data aggregated across the five main process areas – analysis, architecture, development, QA and deployment. It also needs to be flexible enough to recognise new areas that have an impact on the software delivery process. The notion of a connected application lifecycle framework that fosters seamless interaction and allows the aggregation of data and information from different phases is important to the architecture of a DI platform.

### **Acquiring meaningful metrics that add value**

All the best practices in the world are nothing more than words on paper if you cannot prove that they are having an impact. To prove that DI is worth the investment, it needs more than architecture, best practice and analysis. Metrics that mean something, add value or answer questions that matter or make a difference are an absolute necessity. The relationship between metrics and DI is symbiotic. Metrics convey knowledge not intelligence, while intelligence is the ability to apply knowledge. Both work to deliver and apply knowledge that is of value.

The proper collection of the right metrics provides a valuable predictor of problems. The problem is that a) metrics are almost never collected properly and b) the proper metrics are rarely collected.

So what are the proper metrics for DI? Many come to mind, some are straight forward – for example revision velocity, build failure rate, test coverage completeness, or the frequency of certain types of bugs, when they were introduced and how quickly they were fixed (and by which developer). Others are more complex – metrics that tell you how effective a particular methodology or process implementation (Agile and ITIL) was in delivering a solution that met all or most of the business requirements and gave the greatest customer satisfaction. The list of appropriate metrics is long and in many cases subjective. There is no central database where organisations can share metrics that would offer collective value, a failing of the industry and a key pain point for many enterprises.

What will be important for DI is understanding what questions you need to ask of the data and what data is needed to answer these questions. This is likely to reveal any significant gaps in the data, where it either doesn't exist or it is inaccessible. The key to knowing which metrics make sense is to find those that answer the questions that matter to the delivery process, whether they are directly related to delivering the business outcome or achieving some quality- or policy-based goal.



## Development intelligence platforms for sale?

The concepts behind DI are well established – old even. The value proposition and benefits are recognisably desirable, but DI solutions/platforms are still in their infancy. Few vendors have made the leap required to aggregate the necessary features and functions highlighted above into a coherent semblance of a platform, let alone align it with the rest of the key technology initiatives bombarding IT organisations and businesses today.

However, this shouldn't stop interested and visionary organisations from identifying the key features needed by a DI solution.

- Monitoring facilities: automatic collection and tracking of metrics, for example, key performance indicators (KPIs) covering test results and code coverage; alerts to show violations of boundary conditions; ability to track software components, especially third party libraries.
- Archival processes to ensure that the 'history' of everything related to development is captured; for example, a common/ shared repository architecture with the ability to query all sets of data.
- Tools to correlate positive and negative experiences.
- Management reporting system; project health and risk indicator trends; resource identification based on skill sets and experience; model tracking system to identify reuse opportunities; 'what if?' analysis and simulation.
- Tools to correlate positive and negative experiences.
- Iterative processes to improve relationships between the various stakeholders in software development.
- Integration with existing systems such as helpdesk systems, compliance and security.

A number of the features mentioned are already part of other tools – such as portfolio management tools, and performance dashboard interfaces. However, users should expect to see an out-of-the-box, DI-specific targeted offering rather than one that they would need to cobble together from other products in place.

## Support from the IT delivery team

There is a general feeling among developers and others in the software delivery team that they are often blamed for all the woes of the IT department. Their working practices have changed markedly over the last decade. The introduction of developer-driven testing, increasing use of source code management solutions, a new methodology every couple of years, new languages and new architectures – have all affected the software delivery team heavily.



The team does not want more cheap shots or invasive changes to working practices, but something that they can use. It needs to engage them in the 'analysis' of quality process without requiring them to dedicate hours trying to make it work. This is where DI will have its biggest initial impact.

The software delivery team, and developers in particular are not averse to this sort of approach, after all, there is nothing more frustrating than spending large amounts of time rewriting code or chasing bugs. Most, when questioned, welcome the idea of gaining better insight into the quality of their processes and actions. If DI can show that the information is timely, can pick up previous issues to stop them reoccurring, and help developers improve their skills and be more efficient with their coding practices, then they will almost certainly be the standard bearers for DI within the software group.

DI's ability to provide feedback across the delivery process of past projects will be a welcome feature. Areas that are not under the control of developers, such as requirements analysis, software design, testing and deployment, all need better quality information that DI can both leverage and feed into.

By demonstrating that collaboration and intelligence tools can work, vendors will get serious traction from developers.

## Development intelligence in practice

### Real-world struggles

#### Case study: a European public sector IT organisation

The group's IT delivery management team recognises the importance of obtaining better intelligence and insight information in order to help them address issues at the planning stages and overall effectiveness of their processes.

*'Gathering metrics may be nice but you should then do something about them (or with them).'*

They have a number of key areas where they believe DI would help them directly.

- They struggle with estimating development budgets and want to understand how well their methodologies and processes are stacking up in delivering a specific piece of work.
- On a yearly basis, they are evaluated on their performance, which is checked against a peer group. However, to begin with, they are required to provide some up-front figures of how well (or badly) projects were delivered, and the performance of the development



process. At the moment, they have to rely on the gut feel of project managers. Fundamentally, they don't want to rely on gut feelings, but on the reality of the execution.

- They see issues around focus. Their designers spend too much time focusing on functionality when equal time should be devoted to technical aspects, such as network architecture and optimisation. Such technical issues could have as big an impact on the overall effectiveness of an application to the end user. Making subsequent changes to pass user acceptance tests could end up influencing the underlying architecture of the application. A degree of insight into the effects of the downstream impacts could allow the design and development teams to address technical issues up-front.
- The company is pushing the handling of multiple views, such as security and deployment to get an insight into the effect each has on the development, deployment and operation of the project.

Ideally a DI platform would provide them with:

- better insight into how their projects are evolving
- accurate project costs and what is gained when you have smarter requirements, or when you have made a particular methodology choice.
- an understanding of the effect that a specific methodology had on the performance of the project
- helping them to answer questions like, 'are the new methodologies really helping them to focus on delivering more of what the client wants?' 'Are things being delivered within the project budgets and where are the holes?' Ultimately, they want to be allowed to compare different methodologies and approaches and draw conclusions from them. Hence they want to know if they are effectively implementing their ITIL and agile processes and methodologies
- better insights into, and meaning from, the metrics that they are collecting. They specifically would like to obtain better qualitative information about estimation and trend figures from the past that will support the processes in play
- the ability to record communications that happen during the delivery process. They are working on a central build engine. They want to be able to do a new build on code that is checked in and then have an email automatically sent to the developers to highlight errors that have been found and record them as a project asset for future comparisons and improvements.

DI for them is a stepped approach. In the past months they have looked strongly at their suppliers, some of whom have started delivering metrics frameworks with their application development platforms. However, these



suppliers have not yet made the leap and promoted this to the type of intelligence framework that should ultimately underpin DI.

On an intermediate basis, the suppliers are gathering metrics in project-specific portals, and in many cases these portals are being made visible to the clients through web links. The portals help both the client and the suppliers to gain an insight into things like defect-tracking and bug-fixing efficiencies, as well as enabling them to evaluate what is happening in the design, develop and test phases, and identify trends and patterns that can be improved upon.

The peer group believes this is a good halfway house into seeing what the project delivers before taking the final plunge into employing a data mart and a more dedicated intelligence platform that will provide more sophisticated managerial intelligence information.

### **Migration strategy: path to DI nirvana**

Getting from where we are now to the ultimate development intelligence solution will take time and commitment. The path will not always be smooth. The most critical part of achieving this will be a change to the whole software delivery culture. On top of this will be the tools and architecture. This is where we can borrow heavily from the BI paradigm.

The migration steps follow four stages that take us from the environment typical of many implementations today to one where DI is delivering an advanced and sophisticated level of analysis that is better able to contribute to successful software delivery.

#### **Step 0**

Today we see a very fragmented set of structured and unstructured data across the entire software delivery process, for example, word processing documents, models, diagrams, reports and existing database-driven solutions. There are many discrete silos of structured and unstructured data that exist within the three key phases of the IT lifecycle – IT business management, application delivery and support services. The data that each gathers is spread across multiple machines, and where the data is kept it tends to be in multiple silos, either project focused or on an individual's computer. Very little is indexed, made searchable or reused. Consistency is variable, although the introduction of methodologies with standard templates has improved this. Making use of the data for historical analysis of a project is extremely difficult as not all data is gathered or stored.

As mentioned in the case study above some suppliers have begun to implement solutions that don't yet fully embrace intelligence-based tooling and feature sets but do offer some intermediate steps towards DI.



**Action:** the first instance in this step would be to ensure metric-based data and information from processes like QA and developer unit testing is actually recorded and stored, preferably in a structured manner. The next is to look for ways to bring together metrics and associated data/information for a specific project implementation, for example project-specific performance/quality-based dashboards.

### Step 1

Take advantage of the BI tools that are appearing on the desktop. These allow unstructured data such as word processor documents, project-planning diagrams and spreadsheets to be indexed. The ideal would be a single index covering this unstructured data. Both time and careful checking are required to remove duplicates and clean up the data being indexed, a lesson learned from the move to BI. Adding metadata will make query results more accurate.

The tools and knowledge required to do this already exist within most companies but not inside the software development teams. Training will have to be factored in and resources allocated to do the cleaning and assessment of data to minimise disruption to existing software projects.

Gathering all the unstructured data into a common area to improve searches and queries will be a challenge. This will be compounded by the need to begin formalising the retention of data from areas such as unit tests. Test data is often discarded over time rather than being kept for historical analysis. Early DI queries will focus on improving the quality of individual processes.

**Action:** implement intelligence-based tools for each phase to allow coherent analysis and query. This will start the process of bringing a comprehensive set of data together. Silos will continue to exist, but indexing and metadata will allow DI queries to begin to target historical lessons.

### Step 2

Extend the use of the intelligence tools and create unified repositories of structured and unstructured data. Making the intelligence tools more effective depends on the quality and completeness of the underlying data. Data marts will be populated by extracting and transforming data from multiple repositories. As query use increases, a process to establish how effective the queries are and improve the underlying data transformation will be needed.

Unified repositories for each phase are preferred but not a requirement. Some areas such as application delivery may find the move to a single repository relatively easy. This is because a number of software vendors have begun to deploy unified repositories, bringing together functions such



as requirements analysis, architecture design, software engineering and testing. DI queries will begin to look at historical data across particular areas such as testing, code performance, and requirements analysis to see what lessons can be learned. Some early analysis of the entire project may be taking place.

**Action:** implement a managed repository architecture for each phase and extend the use of intelligence tools to allow multi-store queries. Each of the phases will continue to manage their own data, but development and project managers will begin to be able to query across entire projects and not just within each phase. Project lessons will become easier to extract and feed back to the start of the development process.

### Step 3

Ideally the repositories will be merged into one coherent structure. This will take a considerable time to achieve. However, the most important thing will be the way that the use of DI tools develops. Data marts and data transformation tools capable of extracting and cleaning data ready for querying are just as important in this phase.

The DI queries will be comparing multiple projects and looking at the detail of specific areas. All of this mirrors the way that we have seen BI being used within the general computing area today. Initially, it will be used to gain more detailed understanding of parts of the business process, then as a comparison against historical data, and finally for a complete understanding of the business, both historically and in realtime.

**Action:** implement a managed repository architecture and intelligence tools across the entire software delivery process. This is an ultimate goal and one that may never occur. More important is the development of the tools to allow 3D analysis of data using projects and historical data as the key axis.

## Development intelligence – making IT delivery teams more responsive

Ultimately DI is about using data to learn lessons, reduce costs and mitigate risks. Initially it will apply to a single project or the short-term gains associated with comparing the SCM database with the results of QA tests. However, it will eventually encompass everything that makes up a project. BI practitioners talk about data cubes where they take a detailed in-depth view of data. The goal for DI is to achieve that for the whole software delivery process.

### Repository consolidation – not a deal breaker

The combined repositories envisaged in steps 2 and 3 are not an absolute requirement for DI, although, if put in place, they will make the process



for it simpler. Indexing of unstructured data and the deployment of BI tools are the two key requirements. Sitting on top of these is data security and access so that when queries are run, they can return meaningful results.

There are a small number of software vendors who are already designing and shipping repositories capable of aggregating the various types of structured and unstructured data that exists in the software delivery lifecycle today. The downside of these products is that while they are great for new projects, they seem to lack the tools necessary to import the vast amount of existing data.

This is where data marts sit. Data is extracted through automated queries, transformed and pushed into the data marts for users to run their BI queries against. It doesn't matter, to some degree, how that data is stored, just that it can be extracted, validated, transformed and pushed into a data mart.

#### **A shift in culture**

We need a cultural shift in the entire software delivery process. 'Quick fix' software is an oxymoron. It is never quick and often has to be fixed several times. The whole mentality around this approach is about saving time, and not about doing a quality job.

ISVs often explain deficiencies in their software as being the result of a commercial need to get something to market. This same excuse is now becoming more common with in-house development. While ISVs have the funds to put a bug fix/new version process in place, it can literally be years before any in-house project is revisited. Just a little more time spent actually writing the software properly would be more than recouped from the time currently spent fire-fighting bugs.

Making the cultural shift will need more than just the will of customers. Vendors need to provide the tools. BI tools allow data to be mined and lessons drawn. Embedding these tools inside development suites is what is required.



## Client re-use disclaimer

- This is a verbatim reproduction of independent material that has previously been published by Ovum within the last 6 months
- Ovum operates under an Independence Charter. For full details please see [www.ovum.com/about/charter.asp](http://www.ovum.com/about/charter.asp)
- Ovum may have been paid by the client for the right to re-use the material
- Ovum may have a deal with the client to supply research or consultancy. However, no other relationship exists between the 2 companies (e.g. shareholdings, loans, non-executive directorships etc)
- Ovum does not endorse companies or their products
- While we take every care to ensure the accuracy of the information contained in this material, the facts estimates and opinions stated are based on information and sources which, while we believe them to be reliable, are not guaranteed. In particular, it should not be relied upon as the sole source of reference in relation to the subject matter. No liability can be accepted by Ovum Limited, its directors or employees for any loss occasioned to any person or entity acting or failing to act as a result of anything contained in or omitted from the content of this material, or our conclusions as stated
- This material is the copyright of Ovum Europe Ltd.