Database Change Management Best Practices

Achieving an Automated Approach and Version Control

INNOVARTIS

White Paper

Inside

- The need for database change management
- The disadvantages of current practices
- Understanding an automated approach
- Characteristics and benefits of an automated approach
- Substantial cost savings achievable for development projects

About Innovartis

Innovartis helps organisations transform and radically improve their database development processes with its DB Ghost™ change management product family. With database and development specialists, Innovartis offers a range of consulting services and solutions including assessment of the database system, design and implementation of change management processes, database administration and optimisation and tuning.

Information about Innovartis and DB Ghost™ can be reached at:

- Web: http://www.innovartis.co.uk
- Email: enquiries@innovartis.co.uk

© Copyright Innovartis Ltd 2004. All rights reserved

Contents

About Innovartis **2**

Executive Summary **4**

The Need For Database Change Management **5**

Problems With Current Approaches **6**
Manual Scripting 6
Create Brand New Database and Migrate all Data 7
Team Development using Owner Qualified Objects 7
Ascertain Changes with SQL Queries 8

An Automated Database Change Management Methodology **9**
What is an Automated Methodology? 9
Elements of an Automated Approach 9

Implementing An Automated Approach **12**

Benefits Of An Automated Approach **14**
Improved Quality and Faster Software Delivery 14
Auditable and Version Control 15
Sarbanes Oxley Compliance 15
Controlled Rollback 15
Benefits for ISVs 15
Parallel Code Development 16
Improved Productivity 16
Accurate Project Estimation & Planning 16

Implementation Considerations **17**
Return on Investment Analysis 18

Conclusion **21**

© Copyright Innovartis Ltd 2004. All rights reserved

Executive Summary

"Your paradigm is so intrinsic to your mental process that you are hardly aware of its existence, until you try to communicate with someone with a different paradigm."

Donella Meadows, *The Global Citizen*

Most approaches to database change management involve tedious, manual and costly processes that are error prone. In many cases there is no version control of the objects that reside within the database.

The result? Database chaos.

Typically, a database development project will suffer problems such as overwritten changes, code that won't compile, lost data and confusion over what has been applied to which database. Release into production becomes a chaotic and costly phase, with everybody "fire-fighting" and in "panic mode". Can you afford to have one of your most expensive departments working with inefficient processes?

It is a fact that database systems are becoming more complex with ever-increasing demands. Futurists believe that the world will change more over the next 20 years than it has over the past 100 years. Requirements for new functionality and services, timely delivery of information, 24x7 availability, 100% integrity, audit and corporate governance (e.g. Sarbanes Oxley), place enormous demands on database development projects and the teams who administer and support the environments.

Businesses must continually innovate and accept challenges. Can IT departments afford to complacently use processes simply because it's "always been done this way" or is the prevalent paradigm?

This paper will outline the drawbacks of current approaches to database change management, examine an automated approach and describe the benefits that can be realised. Some of the benefits include:

- Remove the need for time consuming manual scripting
- Elimination of errors
- Reduce the length of the development life cycle
- Minimise the cost of development projects
- Empower employees and increase their productivity
- Integration to a source code version control system, which provides the ability to version your database and achieve auditable, reversible (controlled rollback for contingency) and repeatable database changes
- Increase in accuracy of project estimation and planning

The efficiencies gained by adopting an automated methodology cannot be ignored, as the improved performance of the IT department quite often equates to improved performance of the organisation. This approach is an accelerator to the application development lifecycle – the time to market of your application is reduced, and quality of the end product increased. Ultimately, the implementation of an automated database change management methodology will have a positive impact upon a businesses bottom line.

© Copyright Innovartis Ltd 2004. All rights reserved

The Need For Database Change Management

"Change should be a friend. It should happen by plan, not by accident."
Philip Crosby, *Reflections on Quality*

One of the certainties in life is that things will change. This is especially true of database systems that must evolve and change in response to new user requirements and the continuous drive for competitive advantage. Changes to a database may be required for bug-fixing, resolution of defects and performance tuning e.g. de-normalisation. The complexity of some systems can be overwhelming when you consider:

- System integration and inter-dependencies
- The interaction of various technologies and protocols e.g. COM+, MDAC, Web Services, XML, HTML, ASP, SOAP
- Heterogeneous environments e.g. multiple database management systems
- Multiple applications e.g. billing, e-commerce, CRM
- Various environments through which a new release must pass e.g. test, functional test, user acceptance, QA and production

Question: How are changes to database structures and static data managed in such complex and often mission-critical environments?

Question: How can you know with certainty what version a database is in a particular environment?

Question: How can you deploy new versions with accuracy and reliability?

Question: Can you reliably determine who made changes to your database objects?

Question: Do you have an audit trail recording what changes have occurred and at what point in time?

Question: If severe errors occur, can you back-out the corrupt version and rollback to a known state of integrity?

This is the essence of database change management. Every database development project must have a process or methodology in place to propagate and deploy upgrades to database schema and data. After all, the database is an extremely valuable asset, containing transactional, financial and customer information that are the lifeblood of an organisation.

But how efficient and cost-effective are your current database change management processes? Are you able to account for every change made to your database systems in a controlled manner? Do you think there's room for improvement within one of your most costly and business critical departments?

© Copyright Innovartis Ltd 2004. All rights reserved

Problems With Current Approaches

This section describes some of the current methods of database change management, outlining their flaws, and the degree of risk that they contribute to the application development lifecycle. Consequently, this section enables you to understand why an automated approach to database change management adds value to your IT effort by eradicating the issues outlined over the following pages.

Manual Scripting

Probably the most common method of managing database change is via a manual process using script files. This involves hand-crafted upgrade scripts for changes to database code, schema and static data. There may also be the need for a corresponding downgrade script to be used as a contingency to rollback a failed upgrade.

A typical scenario is that a DBA will receive an upgrade script from a developer that is different to the copy in the version control system (VCS), which is again different to the production environment. The DBA(s), while performing a vital quality assurance role, becomes a bottleneck as they must manually resolve such issues plus review every script for correct syntax, functionality (often requiring an understanding of the business rules) and performance. There is also a need to understand how the changed objects will affect their dependent objects (e.g. due to referential integrity) and the order in which changes should be propagated.

After the scripts have been executed against the target environment, if further issues are found, then another upgrade script is required to be coded and so on and so on. You very often end up with lots of incremental scripts just for one new piece of functionality. When you multiply that out over many developers and many new pieces of functionality, the number of scripts could run into the hundreds.

How much does it cost you to have a database failure in the production system? With manual processes, there is always the risk that the development and testing environments do not match the production database in some subtle way. This can lead to production outages that could have been avoided if there was a guarantee that what was tested, matches exactly what is placed on the production system.

Reliance on hand-crafted scripts to manage code baselines and propagate change within your database development project, cannot guarantee complete control. Many IT departments are still using cumbersome manual scripting, which ties-up highly skilled resources in tedious, time-consuming and error-prone tasks.

Summary of drawbacks

- Manual process, therefore time consuming (expensive) and error prone
- Requires upgrade and possibly downgrade scripts
- May require additional upgrade scripts if errors are found (may run into hundreds)
- Increasing pressure upon DBA(s), resulting in bottlenecks
- Quality assurance may be impeded if DBA(s) come under pressure
- Production and test environments may still not match
- Real risk of production outages
- Complete database management control is not a reality

© Copyright Innovartis Ltd 2004. All rights reserved

Create Brand New Database and Migrate All Data

This approach involves the development of scripts to create the database schema and code such as stored procedures. These scripts encapsulate the changes required for the next version of the database. A brand new database is created, upon which the scripts are executed, giving an empty schema. A migration process is executed against the new database involving routines to extract data from the original database and load it into the new upgraded database. This approach has the advantage of being less of a change process, with all its inherent complexities, to more of a creation and initial load process.

Summary of drawbacks

- A much longer time to upgrade due to the migration of potentially huge amounts of data (with an increased risk of data loss and corruption)
- Additional requirements for disk space
- The creation, testing and maintenance of extraction and loading routines
- Inefficient manual process to generate scripts to build the new database with the upgraded schema and code
- Complete database management control is not a reality

Team Development using Owner Qualified Objects

This method provides each developer with their own objects or schema within a common development database. Developers are given exclusive security access using a login, which also prefixes each object to identify ownership. They are then free to code and unit test in complete isolation. Only the DBA has security access to the baseline database objects e.g. in Microsoft SQL Server, this would be the dbo (database owner) prefixed objects.

When development of an object is complete, the developer checks it into the VCS and informs the DBA it is ready for promotion. The DBA replaces the baseline object in the database with the new version and updates an overall upgrade script.

Summary of drawbacks

- Tendency to only work for objects such as stored procedures and functions i.e. Data Manipulation Language (DML). Meaning Data Definition Language (DDL) must be maintained by the DBA(s)
- Results in many manual steps which are time consuming and error prone such as promotion of objects and modifying the final upgrade script
- The DBA(s) must ensure integrity of the process, but in so doing, becomes a bottleneck
- Far too much reliance on diligent communication between individuals
- Complete database management control is not a reality

© Copyright Innovartis Ltd 2004. All rights reserved

Ascertain Changes with SQL Queries

In order for the developer or DBA(s) to apply upgrades, they must ascertain what changes need to be propagated and in what order. One way to achieve this, is to develop SQL queries and stored procedures to compare schemas and extract object properties e.g. in SQL Server, interrogation of information schema views with a FULL JOIN and WHERE clause finding NULL values.

Summary of drawbacks

- This could entail a large amount of visual comparison of schemas, DDL files and system catalogues with the aid of a file comparison tool
- Although such queries and stored procedures can tell you what is different, they rarely have the ability to actually deploy changes to upgrade or synchronise
- An ad hoc and manual approach that is tedious and error prone
- Requires an excellent understanding of the database schema and system catalogues
- Need in-depth knowledge of object dependencies and business rules
- Complete database management control is not a reality

© Copyright Innovartis Ltd 2004. All rights reserved

An Automated Database Change Management Methodology

The following sections discuss the characteristics and theory behind an automated database change management methodology.

What is an Automated Methodology?

If you are involved in a project that involves database development, a crucial element of a project's success, is the accurate propagation and deployment of the database code and schema changes. Often, an unnecessarily large proportion of highly-skilled resource is required for this aspect of the project. Once the analysis and design has been completed, the deployment of the changes, while critical, is quite laborious and repetitive. Therefore, it is highly recommended to systematise or automate these tasks.

An automated methodology is a complete end-to-end solution for database change management and version control, which will minimise human intervention for tasks that are repetitive, time-consuming and error-prone. This process is enabled by an application called the "change control tool".

Elements of an Automated Approach

The following outlines the basic elements required to achieve an automated approach.

Local Development Database

There are two choices here:

- To have a single development database that all developers use, or;
- Each developer has their own copy of the development database

The latter is recommended as it provides each developer with an isolated workspace that cannot be changed by another developer. Typically, each developer would have a cut-down extract of the production system on their desktop. How do the developers keep their local databases up to date? An automated methodology provides the ability for each developer to incorporate all other developer's changes on their local database and then unit test their changes.

Integration with Version Control System

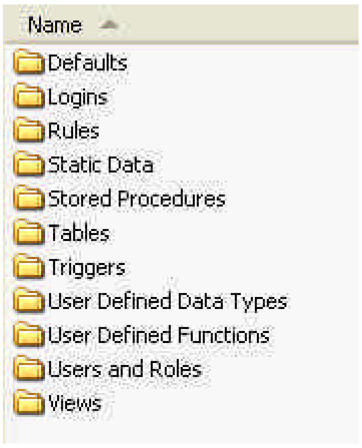
Most IT departments utilise the critical functionality of a VCS to maintain and track changes to source code. It is quite a common practice for developers to store new and amended code and components for languages such as VB, C++, C#, ASP, HTML, Java and XML. But what about the SQL source code and objects which are resident in the database system? It is recommended to store all SQL objects within a VCS, thus enabling automated versioning of the database.

© Copyright Innovartis Ltd 2004. All rights reserved

Building the Source Structure

To implement an automated methodology, all SQL code and schema definitions must first be reverse-engineered and lodged into a VCS (a one-off task per system) usually with a separate folder for each object. An automated approach facilitates the extraction of DDL and SQL code and the creation of the appropriate source structure.

Figure 1: Example Directory Structure for SQL Objects in VCS


Name
Defaults
Logins
Rules
Static Data
Stored Procedures
Tables
Triggers
User Defined Data Types
User Defined Functions
Users and Roles
Views

Static Data

The commands to insert static data (reference tables) are also lodged into a VCS. These commands can be generated by a stored procedure or the change control tool to create an insert statement for each row within your static data tables. Automation simplifies the generation of static data insert commands.

Build Environment

A “build” environment is configured containing a database which is a replica of the production database. An hourly (or daily) compile and build of all SQL objects is run as a scheduled task. This facilitates a proactive approach to verification.

Custom Scripts

As with any complex database system and the applications it supports, not all situations can be covered and will need intervention. On rare occasions, a scenario arises which is ambiguous and the change control process has to guess as to what is required. For example, if a new table schema requires a rename of a column, the decision might be made to drop the column and all existing data – something you may not want to happen.

There are other situations in which the process would simply not know what is required e.g. data is required for a newly created column. In these cases, the automated approach would provide the ability to execute ad hoc custom scripts. For example, for new column data, you would create a script to insert the new data. Then have the automated process call this script after changes have been made to your database.

© Copyright Innovartis Ltd 2004. All rights reserved

Change Control Tool

At the centre of an automated methodology is the change control tool. This is the software that automates database propagation and deployment by performing such tasks as generation of the SQL commands necessary to move a database to the desired version. Key features of such a tool would include:

- Ability to run in user mode (wizard-based) or auto mode (command-line execution) for automated scheduling
- Integration with a version control system e.g. Visual Source Safe
- Schema and data comparison and synchronisation
- Build a new database from a set of object creation scripts hence verifying code
- Compare a database to a set of object creation scripts and perform an upgrade of the database
- Object dependency resolution – algorithms to ensure objects are built in the correct order and relationships are maintained e.g. defaults bound to columns, DRI and circular relationships
- Identify and report all differences and errors
- Automatic generation of SQL change scripts for schema and data
- Capability to reverse-engineer existing databases
- Session-save facility
- Ad hoc custom script execution
- Ability to compare all database objects and properties

© Copyright Innovartis Ltd 2004. All rights reserved

Implementing An Automated Approach

There are probably numerous ways to implement the specific tasks of an automated approach. This is due to the complexities and uniqueness of each environment. But a typical implementation of this methodology would involve the following components (see Figure 2):

1) As a one time process, script out your entire production database's schema and static data into individual scripts and lodge these into your VCS. Instruct the development team to use them in the same manner that they would VB, ASP, C++ or Java code.

2) Decide whether each developer will use a local development database, which is a subset of production, or use a central, shared development repository.

3) Provide each developer with the ability to perform an ad hoc or scheduled update of their local database with the change control tool. Or setup a similar scheduled update if using a shared environment.

4) From that point on, in order to make a change, a developer would:

   - Check out the relevant scripts
   - Make the required amendments
   - Execute it against their local (or shared) development database
   - Perform unit testing and bug fixing
   - Use the change control tool to extract the latest version of scripts from the VCS (which incorporates all other developers changes)
   - Run the change control tool to build a database locally, using the scripts from the VCS and the developer's own changes (to verify that nothing else has been broken)
   - Then check the relevant scripts back into the VCS

5) Use the change control tool to execute a scheduled (hourly or daily) build of the entire schema and code to a "production-sized" build database to verify successful compilation. Label (or snapshot) the source code at these regular intervals to have a named, point-in-time view of your entire database.

6) To propagate and deploy changes to your various environments, you connect the change control tool to the labelled scripts within the VCS and to your target database. The change control tool will then verify that all the scripts compile, with no dependency errors, and make all necessary changes to the target database to ensure that it matches the scripts.

© Copyright Innovartis Ltd 2004. All rights reserved

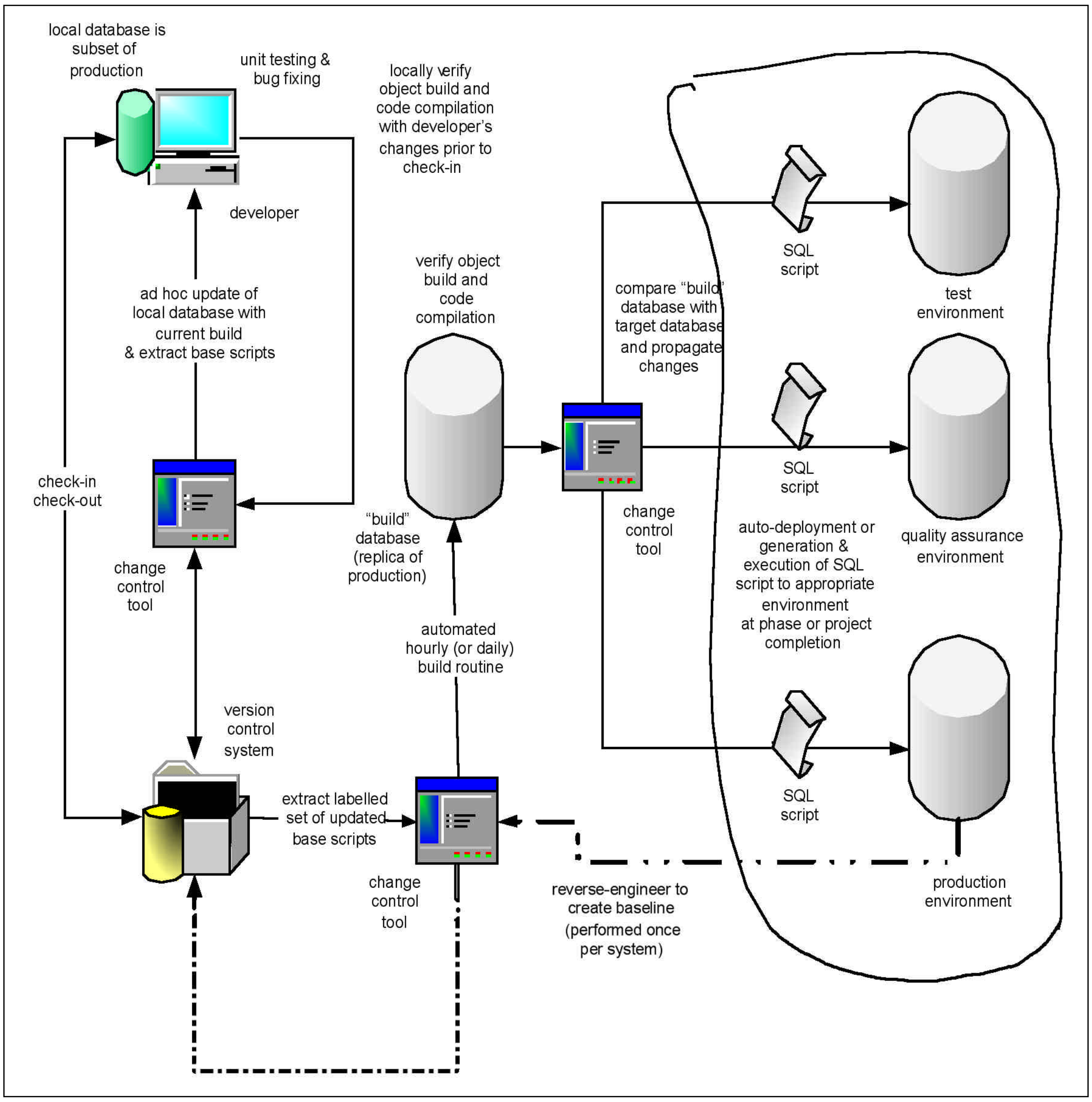

local database is subset of production
unit testing & bug fixing
locally verify object build and code compilation with developer's changes prior to check-in
developer
ad hoc update of local database with current build & extract base scripts
check-in check-out
change control tool
verify object build and code compilation
"build" database (replica of production)
compare "build" database with target database and propagate changes
change control tool
SQL script
test environment
SQL script
quality assurance environment
auto-deployment or generation & execution of SQL script to appropriate environment at phase or project completion
automated hourly (or daily) build routine
version control system
extract labelled set of updated base scripts
change control tool
SQL script
production environment
reverse-engineer to create baseline (performed once per system)

Figure 2: An Automated Database Change Management Solution - Logical Diagram

© Copyright Innovartis Ltd 2004. All rights reserved

Benefits Of An Automated Approach

This section outlines the benefits of an automated approach. This will enable you to appreciate the value that such a solution adds to your IT efforts, when compared to the drawbacks associated with the current methods of database change management.

Improved Quality and Faster Software Delivery

Local Configuration

The use of a local development database means that no developer can be prevented from delivering their changes just because someone else has broken a shared database halfway through unit testing. A regular, automated update of this database from the base scripts in the VCS enables a thorough and swift examination of the developer's changes. This configuration encourages the developer to deliver an improved quality of code and ensures that what ends up in the VCS is of a high standard.

Scheduled Build Verification

The continual testing and verification of deployment via a scheduled build of the code base helps detect bugs, such as syntax, regression and dependency errors, much earlier and while fresh in the developer's mind. The end result is faster software delivery, enhanced quality and integrity of code along with the avoidance of costly and stressful "fire-fighting" at release time.

Elimination of Manual Tasks

One of the main aims of an automated methodology is to systematise change propagation and deployment. Why perform mundane tasks such as table modifications that lend themselves so well to automation? Why, in effect, re-invent the wheel every time a deployment is due? This point is highlighted even further, if you need to deploy to several environments, with possibly hundreds of databases, on the way to production release.

An automated methodology can deploy in a totally controlled, repeatable and reliable manner. Ultimately, elimination of manual tasks will:

- Minimise the cost of development projects
- Eradicate errors
- Reduce the length of the development life cycle
- Reduce anxiety within the project team

© Copyright Innovartis Ltd 2004. All rights reserved

Auditable and Version Control

It is crucial to have the ability to label (snapshot) database code and thereby provide an audit trail and versioning of your database. This provides reliable information on how, when and by whom the database code has changed. It also paves the way for the painless maintenance of static data, which can also break a database when data is lost, overwritten or out of synch. An automated approach with version control enables you to leverage the benefits of your VCS for your database repositories.

Sarbanes Oxley Compliance

In the wake of the high-profile Enron and WorldCom financial scandals, companies today have become more accountable for the integrity of their financial records and practices. Recent introduction of new legislation, such as the Sarbanes Oxley Act (SOX) in the United States, affects the IT departments whose job it is to store a corporation's electronic records. To satisfy the requirements of SOX, an IT department needs the ability to create and maintain a corporate record archive in a cost-effective manner. An automated methodology for database change will help make your systems auditable and assist efforts to meet corporate governance requirements.

Controlled Rollback

A critical attribute of an automated process is the ability to back-out an upgrade in the event of failure. Rolling back deployments manually can become extremely complex. In effect, you have to craft a change script for the failed change script. Things can get nasty if you need to re-instate data that was deleted by the upgrade – either validly or invalidly. If a critical error occurs in your deployment, automation will ensure a straight-forward and controlled rollback.

Benefits for ISVs

If you are an ISV (Independent Software Vendor), then you possibly distribute a product to your customers that relies on an underlying database. To release a new version of your software may also require an update to this database. How do you confidently distribute these changes? How do you know with certainty, what version of the database each customer has and in what condition? What if the customer has made unknown changes or customisations? Creating an install package for your latest release can be daunting.

Another approach would be to include with your release package, the required version of base scripts and an application to interrogate the customer's current database and how it differs from the new version. Changes to databases can then be automatically deployed "on-the-fly", without having to manually track the version and state of each customer's database.

© Copyright Innovartis Ltd 2004. All rights reserved

Parallel Code Development

With the pressures of tight deadlines for development projects, one strategy is to “split” the code base and assign teams to work in parallel on separate pieces of functionality. If you have two or more code bases that developers are making amendments to, then re-integrating them can be tricky at the best of times. With an automated approach and integration to a VCS, you label your code hourly (or daily) and can test against that baseline. With baselines in place, it is easier to analyse the code and perform a merge, knowing that all code on either side is syntactically correct beforehand.

Improved Productivity

The individuals responsible for the management of database change control; whether it is the DBA(s), developer(s) or change controller(s), have a crucial role as gatekeepers and custodians of a valuable asset. Throughout the database development project, the DBA(s) must liaise between development and operations to ensure the smooth deployment of upgrades. This is a delicate balance between swift deployments of functionality (to keep a project on schedule and minimise hold-ups); while at the same time guaranteeing the integrity of the database (by verifying all proposed changes).

Introducing efficiencies to the process of database change will empower these people with greater flexibility and improved productivity. The DBA(s) would then have the ability to propagate a change far more quickly and with a greater standard of quality compared to using manual methods. Consequently, they would have more time to focus on proactive planning, integrity and performance issues. In fact, the effect of an improved change control approach can be felt throughout the organisation and even extend to the customer.

Accurate Project Estimation & Planning

It is unfortunately far too common for IT projects to run severely over budget with continual postponement of completion deadlines. Attempts to hit deadlines by allocating additional resources often exacerbate the problem of escalating project costs. One of the main reasons for this is the myriad of complex technical issues and unforeseen critical errors that can occur. A degree of contingency can be allocated to combat these uncertainties, but methods of reducing development errors is a topic that is high on the project manager’s agenda.

So, another key strength of an automated approach is that it eradicates many unforeseen technical issues that manifest throughout the build and deploy phases of the application development lifecycle. The actual task of estimating project length and appropriate resource allocation becomes a lot easier and much more accurate as many of the “unknowns” are removed. This is because the tasks of generating, building, verifying and deploying changes are automated, thereby minimising errors and avoiding unexpected consequences of changing database objects.

© Copyright Innovartis Ltd 2004. All rights reserved

Implementation Considerations

For the vast majority of organisations, the implementation of an automated database change management approach would require very little effort. This is because most of the pre-requisite components for an automated approach are already available within the organisation itself. It is merely a case of using what is there in a different manner. In addition, such an approach enjoys the following attributes resulting in a smooth implementation:

- Low overhead to install and configure
- Quick to learn and productive within a few minutes
- Requires little or no staff training
- No additional hardware requirements

However, depending on the organisations culture towards change acceptance, the following will need to occur in varying degrees of depth (dependant upon the requirements of the organisation). This will facilitate internal recognition that an investment in time and money should be made for the adoption of a more efficient and effective process:

- Undertake a detailed technical evaluation of the enabling software via a pilot study/trial run in order to equip the evaluator with sufficient evidence to eradicate misgivings concerning acceptance of the new methodology
- Demonstrate that the current processes are outdated and inefficient in comparison
- Produce evidence that a speedy and notable return on the initial investment will be achieved
- Establish an effective communication strategy targeted towards those that are likely to benefit from and utilise the "new way of doing things" so as to ease implementation and gain acceptance

As in any business, in order to justify the implementation of change, an evaluation must include the analysis of return on investment alongside the cost of continuing to practice an obsolete methodology. Only then can the "opportunity cost" be calculated, which can in itself, be the deciding factor in moving forward with the implementation of a new process. If a decision is made to "leave it for now", such a decision incurs an "opportunity cost". What is this cost? Is it more than a monetary value?

By undertaking these exercises, evidence can be communicated and confidence generated amongst the users (and investors), paving the way for a seamless implementation of a new approach to database change management.

© Copyright Innovartis Ltd 2004. All rights reserved

Return on Investment Analysis

The following tables outline an example of the costs of a manual process and the potential savings that can be realised by implementing an automated methodology:

Table 1: Assumptions for Monthly Return on Investment Calculation

Fig.	Assumptions	
-	Developer hourly rate	**$50.00**
-	DBA hourly rate	**$60.00**
-	Change Controller hourly rate	**$30.00**
A	Average hourly rate for systems development staff	**$46.67**
B	Number of individual changes made to scripts per month	**250**
-	Percentage of scripts that break existing code (dependency issues)	**5%**
C	Total database dependency breaks	**13**
-	Percentage of scripts deployed incorrectly or in wrong order	**2%**
D	Total database deployment breaks	**5**
-	Number of target environments e.g. dev, test, QA, production etc.	**20**
-	Number of local development databases	**10**
E	Total target databases to be managed	**30**
F	Average time spent, in hours, per script, fixing defects This should encompass the time from creation of a defect report, triaging it, assigning it to a developer, unit testing, and visual verification by a DBA to ensure the problem has been solved, creating a patch, deploying the patch and re-testing in a QA environment	**3**
G	Number of hours spent by developers completing “release” documentation This includes effort by configuration management staff, administering script changes and verbal communication (per script that has changed)	**0.25**
-	Number of developers	**40**
-	Number of DBAs	**7**
-	Number of “build” machines	**6**

© Copyright Innovartis Ltd 2004. All rights reserved

Table 2: An Example Monthly Return on Investment Calculation

		Manual Approach	Automated Approach	Saving
	Cost of Discovering and Fixing Broken Code (see notes)			
1	Figure C	13	13	
2	Figure F	3	0.25	
3	Total hours spent fixing broken code (1 x 2)	39	3.25	
4	Cost Per Month (3 x Figure A)	$1,820	$152	**$1,668**
	Cost of Deployment Errors (see notes)			
5	Figure D	5	0	
6	Figure F	3	0	
7	Total hours spent fixing broken code (5 x 6)	15	0	
8	Cost Per Month (7 x Figure A)	$700	$0	**$700**
	Cost Of Deployment (see notes)			
9	Figure G	0.25	0	
10	Number of hours spent administering script changes (9 x Figure B)	62.5	0	
11	Number of hours spent deploying the scripts, per script	0.02	0.006	
12	Figure E	30	30	
13	Number of hours spent deploying script changes to all databases (including developers local databases) (11 x 12 x Figure B)	150	45	
14	Total hours spent deploying changes (10 + 13)	212.5	45	
15	Cost Per Month (14 x Figure A)	$9,917	$2,100	**$7,817**
16	Total Cost Per Month (4 + 8 + 15)	$12,437	$2,252	**$10,185**
	Total Cost per Annum (16 x 12months)	$149,244	$27,024	**$122,220**

© Copyright Innovartis Ltd 2004. All rights reserved

Notes to Table 2:

- Cost of Discovering and Fixing Broken Code Using an hourly build verification, broken code is discovered and fixed almost immediately with no paperwork or undue communication with the deployment and test teams. These figures are for just one environment where the problem is discovered. If the script has been deployed to a number of different environments, each tester may raise their own defect causing a “defect storm” which has to be mitigated by the overall test manager (or developer) costing a great deal more time and money.

- Cost of Deployment Errors An automated approach guarantees the reliable deployment of all your scripts i.e. no failures.

- Cost of Deployment Using automated processes, no time is spent by developers completing “release” documentation or communicating with the configuration management team. What is in the Version Control System is what is labelled and deployed. The developers "check-in comments" are all that is needed for a full audit trail.

© Copyright Innovartis Ltd 2004. All rights reserved

Conclusion

The complexity of database systems is on the rise and IT departments face increasing pressures to deliver functionality and timely, accurate information to ensure an organisation's competitive advantage. Additional responsibilities arise from legislation such as the Sarbanes Oxley Act. Companies must assess their current approach to database development and be receptive to a better way of doing things. Failure to embrace improved methodologies can only waste valuable technical resources and prove extremely costly. Your IT department cannot continue to endure database chaos, as it can directly affect a corporation's bottom line. The ramifications of an inefficient methodology extend to lost business opportunities, higher operating costs, second-rate product/service quality, poor public relations, high staff turnover and legal liabilities.

This paper has outlined a new approach to database change management which is increasingly viewed as a best practice and there is no reason why it shouldn't become a de facto industry standard. It is an approach that organisations are strongly encouraged to adopt, as it provides an audit trail and is both reversible and repeatable. This is achieved through integration with a version control system, hourly compilation (hence verification) of database objects, automated generation and propagation of changes and deployment of upgrades. This has the ability to reduce database development project phases from weeks down to days, free-up expensive technical resources from mundane and time consuming tasks (so they can concentrate on deploying their knowledge and skills more effectively and efficiently) and virtually guarantee the integrity of your database code.

© Copyright Innovartis Ltd 2004. All rights reserved