WhereScape RED: Agile Software for Agile Data Warehouse Developers

By Vince Donovan
WhereScape USA

August 2010
Contents
1. Introduction ................................................................. 3
2. Working with Stories ....................................................... 3
   Operations vs. End Users .............................................. 4
   Data vs. Display .......................................................... 4
   Validation and Acceptance Criteria ............................... 5
3. Automated Database Development ................................. 6
   Table Naming ............................................................. 7
   Column Definition ....................................................... 7
   System-Maintained Columns ......................................... 8
   Automated Indexing and Constraints ............................. 8
4. Automated ETL/ELT Development .................................. 9
5. Database Refactoring ................................................... 14
   Table Validation ........................................................ 15
   Change Propagation ..................................................... 15
6. Automated Documentation and Source Tracking .............. 16
7. Version Control ............................................................ 19
   OBJECT VERSIONING .................................................. 20
   PROJECT VERSIONING ................................................ 21
   DATA WAREHOUSE VERSIONING ..................................... 21
8. Testing the WhereScape RED Data Warehouse ............... 22
   INCREMENTAL TESTING .............................................. 22
   VALIDATION TESTING ................................................ 23
9. Production Framework .................................................. 24
   SCHEDULING ............................................................ 24
   DEPENDENCIES ........................................................ 24
   ERRORS, NOTIFICATION, AND RECOVERY ........................ 25
10. From Zero to Hero: Agile Development with WhereScape RED 27
1. Introduction

“Working Software”. It’s at the core of the Agile manifesto. As a BI developer you know these aren’t just words; they’re a serious commitment.

And it’s a commitment that WhereScape RED helps you honor. WhereScape RED rapidly delivers production-quality code fast. It automates the routine aspects of BI development so you can spend more time working with users and refining the high-value project deliverables.

Important agile features of WhereScape RED include:

- Automated generation of standard code: error handling, status updates, documentation, parameters, naming conventions, indexing, etc.
- Automated development of standard BI logic: incremental data loads, slowly changing dimensions, surrogate keys, de-normalization.
- Pre-defined database objects specific to BI: normalized tables, dimension tables, fact tables, aggregates, cubes, operational data stores.
- Support for data warehouse management including partitioning and automatic indexing.
- Complete operations framework, with job scheduler, status and error reporting, dependency management, email notification.
- Complete version management for both stored procedures and database objects.
- Support for development, testing, and production data warehouses.

This document details how WhereScape RED enables an agile data warehouse development environment, so that you, the developer, can deliver on your agile commitment.

2. Working with Stories

WhereScape RED delivers working code quickly when user stories are the basis of your data warehousing or data marting project.

The document “What’s Your Story? Turn User Stories Into Working Data Warehouses with WhereScape RED” gives more detail about how to
User Stories into development tasks. This section is a summary of that document. While WhereScape RED supports a variety of data warehouse design approaches, this document focuses on using agile methods with a dimensional design.

**Deconstructing the Story**

We will break down each story into standard components that map directly into WhereScape RED development tasks.

**Operations vs. End Users**

Determine if the story refers to an end-user deliverable (“I want to see sales by territory”) or an operational requirement (“We need the data warehouse to load every night”). Since WhereScape RED includes a production-quality operational framework, most of the operational requirements of a data warehouse are already implemented within WhereScape RED’s methodological framework, or are easily configured by the design team.

With WhereScape RED you can focus on the end user deliverables, knowing that the operational side will quickly fall into place.

**Data vs. Display**

When analyzing a user story, determine first which aspects of the story refer to data, and which aspects refer to how the data will be displayed or analyzed. This defines two sets of tasks, some for the front-end developer, some for the WhereScape RED developer.

WhereScape RED developers now turn the story into tasks by focusing on the details of the story and then zooming out. For example, for the story “I want to see sales by territory”:

- **What’s the Measure?** Look for the numeric quantity in the user story. This will be the measure in our fact table. In this case it is SALES. We may require further definition as to how SALES is calculated, but this is a good starting point.

- **What attributes might the user want to use for slicing and dicing, reporting and analysis?** These attributes (grouped into natural categories) are the
dimensions. Here it is TERRITORY and, presumably, all of the various attributes that describe a territory.

- **What’s the transaction?** Now think of the story in terms of what transaction the user is referring to. What is the transaction that contains the measure that is asked for? Are they asking about sales orders? Inventory movements? General ledger postings? In this example it is probably booked orders.

- **What’s the source?** Which is the source system that will provide these transactions? The ERP system? HR? The General Ledger? In our example, we may find the orders in the ERP system, if there is one, or potentially in a separate order management system. If there is more than one source required, we may want to modify this story to start with a single source first.

**Validation and Acceptance Criteria**
Each story should include validation and acceptance criteria. How will we know that we are handling the data correctly? Is there an existing report or data set that the data warehouse data should tie to? Defining the validation and acceptance criteria ahead of time will help avoid story creep and ensure that you are working with the right data from the beginning. We will also use these criteria to create the test module for the data warehouse objects as they are developed.

**The Task List**

Working backward from the analysis above, we can create a task list for the WhereScape RED developer:

1. Create the validation (test) procedure. In many agile methodologies, this is always the first step.
2. Create a connection to the source.
3. Identify the master tables in the source that contain the dimensional attributes needed for slicing and dicing.
4. Use these master tables to create load tables.
5. Create dimension tables from the load tables.
6. Identify the transaction tables in the source.
7. Use these transaction tables to create load tables.
8. Create a staging table to perform any required denormalization and to add dimension table keys.
9. Create a fact table from the staging table.
10. Create some reports, queries, or cubes from the fact table to verify that it fulfils the user story.

The following sections describe in more detail how these tasks are performed in WhereScape RED, and illustrate the agile features available at each step as you develop the data warehouse.

3. Automated Database Development

WhereScape RED speeds database development by automating many of the tasks for defining, creating, and maintaining database objects and code, including:

- Tables
- Views
- Indexes
- Sequences
- Stored procedures (see next section)
- User defined parameters
- Business logic
- Column transformations

Automated development is possible because WhereScape RED identifies the type of Data Warehouse object being built and performs the appropriate tasks. Each Data Warehouse object has different properties, so the tasks performed are specific to that object. Automated data warehouse objects include:

- Load Tables
- Staging Tables
- Dimension Tables
- Dimension Views
- Fact Tables
- Data Store (ODS) Tables
- Normalized (Inmon or Data Vault) Tables
- Aggregate Tables
- OLAP Cubes

**Example: Building a Dimension**

Let’s see in more detail how WhereScape automates the development of data warehouse objects, using an ERP system’s product master table as the template for a Product Dimension.
Here’s the source table, as seen from WhereScape RED:

```
<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>product_code</td>
<td>numeric(6)</td>
</tr>
<tr>
<td>description</td>
<td>varchar(64)</td>
</tr>
<tr>
<td>prod_line</td>
<td>varchar(24)</td>
</tr>
<tr>
<td>prod_group</td>
<td>varchar(24)</td>
</tr>
<tr>
<td>subgroup</td>
<td>varchar(24)</td>
</tr>
</tbody>
</table>
```

**Table Naming**

Dragging this table into WhereScape RED’s dimension panel triggers the process of creating a production-quality dimension object. The first step is to apply the previously defined data warehouse naming convention for this new object:

![Add a new meta data Object](image)

**Column Definition**

Secondly, the columns for the new dimension table are automatically defined. The initial definition of the columns is based automatically on the metadata from the source table. They can be further modified as required:
System-Maintained Columns

Note that not only have the table column names and data types been defined for the designer, saving time and reducing the risk of mapping errors, but new columns that are required for the dimensional tables have been defined automatically:

- **Dim_product_key**: this is the surrogate key for this dimension. In this example it is an integer identity type, so the value will self-increment when a new row is added from any source.
- **Dss_start_date, dss_end_date**: these are effective date start-and-end date stamps for each row. Required for type-2 slowly changing dimensions where there may be multiple copies of a dimension row.
- **Dss_current_flag**: set to ‘Y’ for the current version of each row.
- **Dss_version**: shows the sequence of versions for each dimension row.
- **Dss_update_time**: date and time of last update for this row.

With a quick drag and drop operation, WhereScape RED has created a production-ready dimension table, implementing industry best practices for this type of object.

**Automated Indexing and Constraints**

Since the primary and foreign keys were identified during the table definition, WhereScape RED automatically generates the appropriate indexes and key constraints for these columns.
While these are probably not all of the indexes that will be required on this table -- additional tuning will probably be required once the warehouse is in production -- it's a useful head start:

```
-- dim_product_id After table dim_product Add Constraint dim_product_id PRIMARY Key Clustered (dim_product_id) WITH (SORT_IN_TEMPDB = OFF)
-- dim_product_id dim_product_id_Dropped
-- dim_product_id Create Unique NonClustered Index dim_product_id on dim_product (product_code,product_current_flag,product_version) WITH (SORT_IN_TEMPDB = OFF)
-- dim_product_id dim_product_id_Areated
-- dim_product_id Create Unique NonClustered Index dim_product_id_SC on dim_product (product_code,product_group,product_current_flag,product_version) WITH (SORT_IN_TEMPDB = OFF)
-- dim_product_id dim_product_id_SC_Created
-- dim_product_id dim_product_id_SC_Created
-- dim_product_id table comment added
-- dim_product_id table comment added
```

### 4. Automated ETL/ELT Development

Data coming into the data warehouse needs to be processed: loaded, de-normalized, transformed, de-duped, key mapped, change managed, quality assured, and so forth. Whether your development team prefers ETL or ELT (WhereScape RED offers both), you need database procedural code to handle these processing tasks, and usually lots of it.

This code must be of good quality. Because data warehouse systems have no control over what data is loaded into them, there is ample opportunity for error. Update code must have good error handling.

Another important part of good update coding is status reporting. Since data warehouse load structures always have many dependencies, it is important that each ETL process report its operational status, success or failure, so that load dependencies can be managed in case of failure.

On top of it all, the code must be supportable: well structured, documented, using accepted patterns.

That’s a lot of coding, and it needs to be done quickly and well if the development team is going to deliver on time.

Fortunately, WhereScape RED will do most of the development work automatically, by leveraging the meta data collected during the design process. By automating development, WhereScape RED assures high-quality, consistent, high performing code.

**Example: Loading the Dimension**
In the previous section we quickly designed and created a dimension table using the source product master as a template. That table is no good to anyone, of course, unless it has data. In the following sections we’ll continue the development process by using WhereScape RED to generate a T*SQL stored procedure to load the table. The stored procedure will handle the special features of a type 2 slowly changing dimension, and manage real-world error and status messages.

**DATA TRANSFORMATIONS**

Transformations are easily defined for any data column at any point in the load process. WhereScape RED supports all native SQL and procedural functions. New columns can be defined to support derivative business logic. Transformations are inserted into the update logic and are also part of the documentation produced, so the changes occurring to data as it flows through the data warehouse are obvious and well documented.
BUSINESS KEYS

The first step is to identify the business key: that is, the natural key of the source system. For our Product dimension, this is the product code. As we’ll see, WhereScape RED will automatically create the logic to map this to the data warehouse surrogate keys:

Define dimension business key(s)

Select the business keys that uniquely identify each record in the Dimension table.
Move them over to the business key list

Business key list

product_code

SLOWLY CHANGING DIMENSIONS

Next we identify any columns in the dimension that we would like managed as slowly changing attributes. This means that when changes to this item are detected, a new row is written rather than overwriting the old data. WhereScape RED supports all of the common slowly changing dimension types.
Define slowly changing dimension columns

Select any columns to be managed as slowly changing dimension columns.

Move any columns over to the slowly changing column list, or leave it empty.

Slowly changing column

| proc_line
| prod_group
| subgroup |

De-Normalizing

It’s common to join multiple tables when loading a data warehouse table. De-normalization is an important function of data warehousing. WhereScape RED detects if multiple tables are required from the source and facilitates creation of the join logic:

Define the joins [credit the where clause]
To define a Join select two tables and press the ‘Join’ button. Then select the join columns from the drop down lists provided.

Initial Load, Incremental Load, and More

We’ve got our basic load logic in place, but there is more to delivering the data than that. Simple logic is sufficient for an initial load of the tables, but real production requires incremental logic to detect changes and correctly update the records in the dimension tables.

Not only that, our practice of introducing a surrogate key into the data warehouse requires that we make available a stored procedure so that when transaction records are loaded into the fact tables, it is fast and efficient to look up the new keys.

Fortunately, WhereScape RED does all this for us automatically. All of the logic created is incremental, so this code is ready to go into production. A special stored procedure is generated automatically to support surrogate key lookups.
STANDARD, OPEN, QUALITY CODE

The output of WhereScape RED’s development process is not some hidden proprietary file, but open, standard code in the native procedural language of the database technology we are working with. The code can be further modified by hand if required, though if your project is cycling quickly, staying with the code wizard as long as possible will give you the quickest and best results.

Here’s a section of the update code written for us after just a few minutes of working with WhereScape RED:

```sql
.
WHERE product_code = @v_dim_product_code
AND das_current_flag = 'Y'
END

-- Add the new record
-------------------------------
SET @v_step = 150
IF @v_insert_ind = 1
BEGIN
SELECT @v_dim_das_end_date = NULL
-- Increment the version number. Normally will just set to 1
SET @v_dim_das_version = @v_das_version + 1

INSERT INTO dix_product
    (product_code, description, prod_line, prod_group, subgroup, das_current_flag, das_version, das_start_date, das_end_date, das_update_time)
VALUES
    (@v_dix_product_code, @v_dix_description, @v_dix_prod_line, @v_dix_prod_group, @v_dix_subgroup, @v_dix_current_flag, @v_dix_version, @v_dix_start_date, @v_dix_end_date, @v_dix_update_time)

SELECT @v_return_status = @ERROR
    @v_row_count = @ROWCOUNT

IF @v_return_status <> 0
BEGIN
ROLLBACK
SET @p_return_msg = 'Failed insert into table dix_product'
SET @p_dub_code = CONVERT(varchar, @v_return_status)
SELECT @v_db_msg = description FROM master.dbo.schemas
```

QUICK CHANGES
Did we get the business logic slightly (or very) wrong? Did we forget to include some columns as slowly changing dimensions? Is there a new source of data to be included? Or did the user just think up some new requirements?

Agile methods don’t ignore these situations, but rather embrace them as the quickest route to satisfied customers. WhereScape RED embraces change as well, implementing new requirements quickly, with high quality and low risk.

For additions or changes to the user’s business logic, simply update the transformation in WhereScape RED for the appropriate column and rerun the code wizard. WhereScape RED remembers all previous values, so re-generation is quick and solid. Logic changes may also require database changes, which are discussed in the Database Refactoring section.

**5. Database Refactoring**

New user stories or revised business rules may require database changes as well as code changes. WhereScape RED makes it easy to incrementally develop database objects.

**NEW, EDIT, COPY, ADD**

New columns are quickly defined in WhereScape RED. A useful strategy for calculated columns is to copy an existing column that will be a source of the calculation.
**TRANSITIONAL OBJECTS: LOAD TABLES, STAGE TABLES, WORK TABLES**

Most of the tables in a data warehouse are used to stage data as it goes through the update process. If refactoring is required for these tables, WhereScape RED makes sure that dependent stored procedures are updated as required:

![Re-create an existing table](image)

**PERSISTENT OBJECTS: DATA STORE, NORMALIZED, FACTS, DIMENSIONS**

Once in production, certain tables should never be dropped. These are the key reporting tables that are accumulating the organization’s data over time. Normalized, Fact, Dimension, and Data Store tables all fall into this category.

WhereScape RED facilitates incremental database development and database refactoring even of production tables. Changes to the tables can be propagated easily.

**Table Validation**

Any metadata changes are first validated against the existing table:

![Table validation list](image)

**Change Propagation**

The alter table wizard then generates the script to propagate the change to the live table.
A new DDL script for the entire table is also automatically created in case the table ever does need to be dropped and rebuilt.

6. Automated Documentation and Source Tracking

“Working software over comprehensive documentation” is one of the foundations of Agile methodologies. Working software is paramount, but documentation is still important. What if you could have both without compromising the agile spirit?

As you develop your data warehouse, WhereScape RED automatically tracks object properties and dependencies, as well as join conditions, table relationships and other information that will be useful for the ongoing support and administration of the data warehouse.

With only a few clicks, WhereScape RED uses this metadata to create a valuable resource for report programmers and developers that would otherwise require many hours. The following sections illustrate the important features of WhereScape RED’s data warehouse documentation.
Join relationships between fact and dimension tables are automatically detected and tracked as the tables are built.

Because these diagrams are output by the development process, they are self maintaining. Diagrams are never out of date, never require manual modification.

**SOURCE TRACKING**

As the data warehouse gains functionality, it also gains complexity. Track-back diagrams are crucial to the supportability of the data warehouse system.

Again, WhereScape RED generates these diagrams automatically, as part of the development process:
Like the database schemas, these diagrams are self-maintaining. As the data warehouse develops, the documentation develops right along with it.

**Web Based User Documentation**

In an Agile project, great emphasis is placed on verbal communication between developers. But front-end developers, who are rolling out the critical reports, dashboards, and data visualizations, also need detail-level information about the data warehouse objects. Details about the dozens of columns in a data warehouse are often most efficiently delivered in a document or over the web.

WhereScape RED generates a user-level document as a basic web page that report developers can use as a reference. It includes the latest schema diagrams, table and column definitions, a glossary, and catalog of naming conventions.
7. Version Control

Agile development means being quick and bold. Agile means trying new things, and then trying more new things if the first ones don’t work out.

This approach requires easy, flexible version control. And not just of code files; database objects and their dependencies must be managed as well.

WhereScape RED allows versioning at several levels. It also automatically versions most objects at key points during the development.

<table>
<thead>
<tr>
<th>Object</th>
<th>Versioning Details</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped_dm_product_key</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-15 12:24:07 3/73</td>
</tr>
<tr>
<td>ped_dm_product_key</td>
<td>Auto version on delete by Vince</td>
<td>2010-07-15 11:11:27 793</td>
</tr>
<tr>
<td>update_dm_product</td>
<td>Auto version on delete by Vince</td>
<td>2010-07-15 11:11:36 201</td>
</tr>
<tr>
<td>update_last_orders_analysis</td>
<td>Auto version on regeneration by Vince</td>
<td>2010-07-14 14:45:01 547</td>
</tr>
<tr>
<td>update_oep_orders_analysis</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-14 14:27:36 547</td>
</tr>
<tr>
<td>update_last_orders_analysis</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-14 14:28:373</td>
</tr>
<tr>
<td>update_last_forecast</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-14 14:06:51 903</td>
</tr>
<tr>
<td>update_last_budget</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-14 14:05:37 450</td>
</tr>
<tr>
<td>update_stage_forecast</td>
<td>Auto version on auto complete by Vince</td>
<td>2010-07-14 13:07:967</td>
</tr>
</tbody>
</table>
WhereScape RED allows versioning for:

- Tables
- Procedures
- Database objects
- Projects
- Whole data warehouses

**Procedures**

Procedures are automatically versioned when they are re-generated using the wizard, when they are manually edited, or when they are re-compiled.

**Object Versioning**

Choosing an object to version gives you the option to version a linked set of database tables and dependant procedures:
PROJECT VERSIONING

WhereScape RED allows you to divide the development effort into projects. As each developer works down the task list, an entire project can be versioned:

To restrict the versioning to specific projects, select the required projects and click the 'Version Projects' button.

Select the projects to be versioned. Double clicking on a group will limit the project list to only those projects in the group.

DATA WAREHOUSE VERSIONING

Ready for a release? Version the whole environment before starting the next development cycle:
8. Testing the WhereScape RED Data Warehouse

Testing is a part of every development effort. WhereScape RED supports test-led development with a variety of features:

- Configurable web link enables quick communication between WhereScape RED and DBFit or other test management facility.
- Validation testing is now automated and a standard part of WhereScape RED’s operational framework.
- Tests can be applied to database tables, transformations, stored procedures, cubes, and aggregates.

Generally two types of testing are required for all data warehouses: incremental testing, as the new data warehouse object is being developed, and validation testing, which is used during operation to monitor the quality of the incoming data. WhereScape RED facilitates both.

**Incremental Testing**

Each database object can be tied to an external stored procedure or script that will be called after the standard load procedure has completed. In the example below, the table “stage_budget”, has a custom procedure defined called “unit_test_stage_budget”, which can be called individually, or as part of a periodic load routine.
This test procedure, usually written before development begins, can output to the WhereScape RED scheduler log so that test error conditions can be quickly assessed.

Test procedures can also increment or update database parameters. This is a good way to develop statistical data on incremental testing performance when many database objects are involved.

**Parameter listing**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>load_budget_0</td>
<td>18</td>
<td>Rowcount from a native load</td>
</tr>
<tr>
<td>load_customer_0</td>
<td>6</td>
<td>Rowcount from a native load</td>
</tr>
<tr>
<td>load_product_0</td>
<td>9</td>
<td>Rowcount from a native load</td>
</tr>
<tr>
<td>incomp_project_test_failures</td>
<td>8</td>
<td>Number of test script failures in the last build.</td>
</tr>
<tr>
<td>incomp_project_test_successes</td>
<td>34</td>
<td>Number of test script success in the last build.</td>
</tr>
</tbody>
</table>

**Validation Testing**

Even after development is complete, testing must continue. Periodic production loads into the data warehouse must support validation test to make sure that the incoming data is of good quality.

WhereScape RED’s production framework supports validation scripts that can be tied to any and all objects as they are loaded.

<table>
<thead>
<tr>
<th>Fact</th>
<th>Process</th>
<th>23.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>fact_budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact_budget_validation</td>
<td>Execute</td>
<td>24.9</td>
</tr>
<tr>
<td>fact_orders</td>
<td></td>
<td>25.8</td>
</tr>
<tr>
<td>fact_order_validation</td>
<td>Execute</td>
<td>25.9</td>
</tr>
</tbody>
</table>
9. Production Framework

Multiple source systems. Dozens, even hundreds of tables. Thousands of lines of code. Complex dependencies. How can we make sure that everything will load at the right time, in the right order? How can we reduce the impact of the inevitable errors in the incoming data, or problems with source system availability, network outages, etc?

There are several possible solutions. Most operating systems have native schedulers that can trigger procedures or builds. But only WhereScape RED’s scheduler is integrated with the data warehouse objects themselves. Not only can it manage dependencies, but it also reports status and makes problem detection recovery quick and painless.

Scheduling

WhereScape RED’s scheduler offers a wide range of options for job execution, including daily, weekly, monthly, yearly, or completely customized.

Dependencies

Load dependencies are critical to data warehousing. WhereScape’s job builder provides default dependencies, based on object type, with full configurability. The “Order” column below shows the initial assignment of dependencies for a typical load job. Load tasks can be grouped to run in parallel, if required.
Load errors are inevitable in most data warehouse systems where we have no control over the source data or the source systems. For this reason, our data warehouse production framework must be designed from the ground up to gracefully detect, handle, and recover from production errors.

WhereScape RED’s scheduler is a complete operational solution for managing the hundreds of objects and events that may be part of any periodic load. Notification is easy to understand:
Ample details are provided for problem investigation and resolution:

And recovery is a snap:
10. From Zero to Hero: Agile Development with WhereScape RED

This document has characterized the features of WhereScape RED that help the agile data warehouse developer meet the commitments of the project: quick delivery of working code that meets the customer’s needs.

To recap:

- WhereScape RED facilitates the direct conversion of typical user stories for a data warehouse project into tasks for the WhereScape RED developer.
- WhereScape RED gets real source data in front of business users early and often during the design process so that their feedback becomes part of the design and source data quality issues are exposed early.
- WhereScape RED’s extensive automation of both logic and database development enables the developer to
focus on the high-value deliverables while WhereScape RED handles the routine tasks.

- WhereScape RED has features to support both test-led and feature led development.
- WhereScape RED’s output is ready for production from the very start, with incremental logic, slowly changing dimensions, error handling, and process communication, so that a “hardening” project cycle is not required.
- WhereScape RED’s automated source tracking and user and technical documentation means that documentation need not suffer in the push for working code.

WhereScape RED’s operational framework for data warehouses means that no developer resources need be spent on setting up or maintaining the day-to-day operations of the data warehouse.

11. From Zero to Hero: Agile Development with WhereScape RED

Everyone understands the competitive advantage a business intelligence environment can bring – the problem is they take too much time to build before they deliver. That’s where WhereScape RED comes in. It provides a complete methodology as well as a development and management framework – all while leveraging industry standard database technology.

WhereScape RED enables the building of fully functional data warehouses in days or weeks, saving weeks or months of consulting time and money. This means you can get user feedback faster, and improve the quality of the warehouse implementation, as well as save time and money.

For more information on WhereScape RED please visit www.wherescape.com.