



## **Integrating Legacy Data**

Dealing with Legacy Data and Legacy Data Schemas

/// WHITE PAPER

**Bert Hooyman**  
MphasiS Europe B. V.

With Contributions from:

**Rajesh Srinivasagam**  
**Saravanavel Padmanabhan**

August 2009

## Table of Contents

- 1. Introduction ..... 2
- 2. O/R Mapping Strategies ..... 2
- 3. Replicate to Another Schema ..... 2
- 4. Apply a View to Transform the Schema ..... 2
- 5. Federate and Transform within the ORM Framework ..... 3
- 6. Mapping Object Models to a Database Schema ..... 3
  - Baseline Functionality ..... 3
  - Greenfield Databases ..... 3
- 7. Categories of Complexity ..... 3
- 8. Object Property Still Maps to a Single Column ..... 3
- 9. Properties No Longer Map to a Single Column, But Object Still Maps to a Single Table ..... 5
- 10. Object Maps to Multiple Tables ..... 5
- 11. Business Object Model Maps to Multiple Databases ..... 6
- 12. Conclusions ..... 7
- 13. Case Study ..... 8
- 14. Index of Issues ..... 9

## 1. Introduction

In this paper, we seek to identify potential problems that arise when, in the context of a new business software application, a business object model of the problem domain must be mapped against an existing (legacy) database schema. This is in contrast to the “greenfield” scenario, where a database schema can be defined that aligns with the business object model.

The purpose of this exercise is to understand what the requirements will be for any object-relational mapping framework that is considered for the data access. Such a list of requirements can be used to select one of the many existing ORM frameworks or alternatively it can be made part of the functional specification of a newly designed ORM framework.

## 2. O/R Mapping Strategies

Object/relational mapping is the industry standard approach to integrate a relational database in an otherwise object-oriented software design. The mapping framework is typically responsible for hiding the technical details of having to deal with a query language, database connections, referential integrity etcetera. Examples of popular O/R mapping frameworks include Oracle TopLink, Hibernate, the Java Persistence API in EJB 3.0, NHibernate and Microsoft’s Linq.

When having to deal with an existing database schema, rather than a newly defined schema, there are many issues that one would normally wish to avoid. These issues come in many different flavors of varying levels of complexity, and we will discuss these in more detail further down in this report.

Before we address these issues though, we will describe the broad solution directions that are normally considered when having to map against a legacy schema. There are three such directions:

1. Replicate Legacy Data to a Database with a Different Schema
2. Introduce a View on the Legacy Database
3. Federate and Transform using O/R Mapper

These three strategies are detailed in the following sections.

## 3. Replicate to Another Schema

The most drastic approach, from a data management perspective, is to replicate the database content to a shadow database that has a different schema, one that matches the target domain model more closely. During the replication, data is transformed to meet the data model needs of the new business software application.

Read-mostly scenarios, and especially write-once scenarios are good candidates for this strategy. There is a problem as soon as there are data updates to be made in the new business software application, as such updates will be constrained not only by the replicated schema but also by the ultimate source schema. The updated data (and any newly inserted data) must be posted back to the original source to be processed.

Other problems with this approach are issues with data timeliness and issues with data volumes.

## 4. Apply a View to Transform the Schema

This approach requires collaboration with the owner of the original database, and is further limited when there is no single current database that covers the business requirements of the new business software application in full. Defining a database view across multiple databases is not a recommended practice, and would anyway be limited to a single-vendor deployment (e.g. all Oracle databases, or all DB2).

The benefit of a database view is that it is always up to date and requires no data copying. Materialized views are of course an exception to this rule. When the views must be updateable, then there are limitations to the schema. For example, most databases do not support updateable views that join data from more than one base table, which limits the usefulness of view-based transformations.

## 5. Federate and Transform within the ORM Framework

Using this approach, the complexity of the object-relational mapping framework is increased so that the object model can map against a very complex (combination of) underlying database schema(s). Every transaction against the data source (data retrieval as well as data modification) is processed by the mapping layer. It is this approach that we consider in the remainder of this report.

## 6. Mapping Object Models to a Database Schema

### 6.1 Baseline Functionality

In this exercise, we will make some assumptions about the baseline behavior of an O/RM framework. The features and functions identified here are considered to be the “easy part” of the mapping problem. Later, we will describe the scenarios where one or more of these assumptions are not met.

#### Every Object Maps to Exactly One Table

This assumption states that every business object is directly associated with exactly one database table<sup>1</sup>. The reverse need not be true; one table can be the storage destination for multiple object classes.

#### Every Property Maps to Exactly One Column

This assumption states that for every object property, there is a column defined in the associated table that is the storage destination for the property.

#### Every Business Object Model Maps to Exactly One Schema

This assumes that no federations are needed, i.e. all business domain data needed and produced by the business application is held in one single database.

#### Property Type Matches Column Data Type

In an ideal world, the database table stores data using a data type that is closely aligned with the data type defined for the object property. Dates map to dates, numbers to numbers, text to text etc. In reality, simple transformations are not a problem (currency mapping to text, date mapping to integer etc.)

#### Object Associations are Realized as Referential Integrity Constraints

Here, we are referring to 1:1, 1:N and N:1 associations; the more complex N:M association is typically realized through a merge table with two N:1 associations. The assumption is that all relationships are explicitly defined.

### 6.2 Greenfield Databases

The above baseline functionality represents the design blueprint for the scenario where you are free to set up a relational schema for a new development. This is the scenario that all the present O/RM frameworks cover comfortably. When faced with the task to design a database schema against a given object model, use the above assumptions as a design directive.

Some frameworks, notably Ruby on Rails, have gained popularity through an even more aggressive simplification based on a few more assumptions:

- Object Name Matches Table Name
- Property Name Matches Column Name

At this point, the problem is comfortably trivialized. Unfortunately, this does not match any realworld scenarios where a database is already deployed and a new business software application must be designed to work against that database.

## 7. Categories of Complexity

In the previous section, we presented some baseline assumptions around the complexity of object/relational mapping. Now, we will challenge those assumptions and use this as a way to classify legacy data mapping problems.

## 8. Object Property Still Maps to a Single Column

Even under this sunshine scenario, where a given object property maps comfortably to one single column of a database table, there is a significant series of mapping problems to be expected.

### Issue 1 – Column Abuse

A single column is being used for several purposes, depending on the user community. Data fields designed for dealing with a specific geography but carrying no meaning in another region are prone to abuse. The social security number is a prime example of this.

### Issue 2 – Column Interpretation

The purpose of a column is determined by the value of one or more other columns. In a “Party” table, the “party type” column can be used to distinguish employees from customers and vendors.

Problems may exist when the interpretation is ambiguous, for example when a bitmask is used for the party type column, one party record can identify both

<sup>1</sup> We consider only relational databases here, not hierarchical, networked, semi-structured, associative or EAV models.

a customer and an employee. In that case, the party number is either the employee ID or the customer number (but most likely not both).

### **Issue 3 - Incorrect Derived Data**

A Party table holding both a birth date and an age group indicator is vulnerable to incorrect (stale) derived data. As long as the derived value (i.e. the age group) is defined as a calculated field, the correctness will be ensured by the RDBMS. It is in the absence of calculated fields that the errors will be introduced.

### **Issue 4 - Missing Data**

The business object model specifies a property as required but the database schema defines it as optional. One will have to revisit the business object model and consider why the property had been identified as a required property.

### **Issue 5 - Data Lifecycle Mismatch**

The business object model specifies a property as updatable but the database schema defines it as immutable. Or even worse, the readonly nature of the data is implied.

Sales transactions are typically immutable: when an error is made a compensating transaction is created. An object model that allows changes to sales transactions is a violation of this implied constraint.

### **Issue 6 - Property is Part of a Column**

Important entities, attributes, and relationships are hidden and floating in text fields. Complex parsing may be required to extract the desired information. Smart keys (primary keys that contain one or more subparts which provide meaning) are one example of this problem. The key must be parsed to extract the meaning.

### **Issue 7 - Value Default Conflicts**

The business object model defines a property default value that differs from the default used in the database. Assuming that the chosen default is an allowed value for the database (which is not always the case), there is still a problem because both the legacy applications and the new business applications may make assumptions based on the presence of a default value ("if the sales date is 99-99-9999 then this is a test sale" versus "if the sales date is 00-00-0000 then this is a test sale").

### **Issue 8 - Wrong Data Type**

These are both the obvious problems solved either by type-casting or string parsing, as well as the more complex problems related to insufficient storage allocation.

A legacy database may define an ISBN number as a 10-digit number (and store it as string of length 10), whereas the business object model defines the ISBN number as a 13-digit number.

It may also be the case that the database holds user-defined data types as part of the legacy schema. For these, the O/R mapping software must provide either a direct mapping to the user-defined type, or alternatively a parsing/interpretation of the underlying raw data item(s).

### **Issue 9 - Use and Interpretation of Special Characters**

The use of hyphens, carets, angle brackets, asterisks, question marks, periods, braces, forward and backward slashes etcetera may lead to interpretation problems in the business object model.

As an example, consider "-200", "12-07-2009" and "1995-1998". In the first case, the minus sign means "negative number", in the second case, it acts as a separator between day, month and year, whereas in the third case it is used as a range indicator.

### **Issue 10 - Required Data Formatting**

Related to the interpretation of special characters, the business requirements may impose specific formatting constraints on data that is stored as a free-format text string. This type of requirement would come from scenarios where the business software application is automated, i.e. there is no human interaction. As an example, EDI type data integration requires that connection details are provided in a known fixed format. When the database holds the details in a free-format text field, the O/R mapping software may be held responsible for formatting the data before exposing it to client code.

### **Issue 11 - Data Decryption & Encryption**

For reasons of privacy or security, data may be stored in the database in an encoded or encrypted form. The responsibility for decryption may lie with either the O/R mapping software or with the business logic of the business software application. Likewise, business data may require encryption on its way towards the database. Note that this is very similar to the issue of "required data formatting" discussed above.

### **Issue 12 - Wrong Encoding**

Encodings are frequently used as a means to reduce storage. Weekdays may get sequence numbers, data stage or state is encoded using an acronym or a number, etcetera. Mapping problems arise from the use of different encoding schemes. The problem may be aggravated when combined with issue 2: "column interpretation" (see above).

### **Issue 13 - Undocumented Column Encodings**

This is a data quality issue as it pertains to data values in an encoded column that can no longer be reconstructed to any business meaning. Still, such code points must be represented in the business object model.

**Issue 14 - Data Lifecycle Mismatch**

The business object model specifies that object instances can be deleted, but the database schema defines that data cannot be removed. Or even worse, the permanent nature of the data is implied.

In a library administration, individual loan records (transactions) are captured in a database table. These records form a history trail of a book's usage and this might be required for determining when a replacement copy is needed. The business object model may wrongly assume that old loan data is irrelevant and define it as data that can be removed.

**Issue 15 - Data Timeliness Mismatch**

Table content may be synchronized with some external source using a batch data movement scheme. In between synchronizations, the data is stale. The business object model may not be aware of this.

This becomes complicated when the data movement is set up at the individual column level, mixing content from multiple sources into one table.

## 9. Properties No Longer Map to a Single Column, But Object Still Maps to a Single Table

This class of scenarios deals with a very common business situation, where the business entities are mapped onto the database correctly, but the entity details (object properties) are either elaborated or changed since the database schema was first defined.

**Issue 16 - Property Derived from Multiple Columns**

A single object property value is derived from combining multiple data column values. For example, the database schema holds street name, house number and house suffix as three columns whereas the object model uses a simple address property that combines the three fields. May be seen in combination with issue 6: "property is part of a column".

**Issue 17 - Row Subset Mapping**

Not all rows of a table represent valid instances of an object. The filter can be based on a single column value, a combination of column values or some external, possibly dynamic, evaluation.

**Issue 18 - Unmapped Required Columns**

Required (not null) columns in the database are left unmapped. There is no matching object property. In this case, the O/R mapping framework must provide valid default values in case the database schema has no default set up. In general, though, this is a signal that the object/relational mapping is incomplete and further analysis of the legacy schema is recommended.

**Issue 19 - Unmapped Properties**

This is the situation where the business object model is more detailed in its model compared to what the database schema can support.

For hard business requirements, when the unmapped properties are absolutely indispensable, the options are to either extend the legacy schema or to create an extension database where, for each object with unmapped properties, a table is defined that holds those properties. We then have a situation where object properties are spread over multiple databases (i.e. database federation), which is in itself an issue of another level of complexity, as discussed later in this report.

**Issue 20 - Implicit Subtyping**

Object-oriented models are comfortable with the concept of a base class with subclasses. The same concept exists in entity-relationship modeling and mapping strategies exist. In some cases, a more ad-hoc form of subtyping is used where conditional data interpretation has been added late in the lifecycle of a relational database (see also issue 2: "column interpretation" above).

The most complex case is when the subtyping is implied, in other words there is no single subtype identifier column. In this scenario, data interpretation becomes complicated ("if country code equals DE and transaction type equals DIR or transaction year less than 1995 and amount greater or equal than 100000 then net sales amount is sales amount minus discount amount times (one plus vat)").

## 10. Object Maps to Multiple Tables

We now discuss the issues associated with the situation where object properties are spread over multiple tables. This is usually a result of poor data modeling in the legacy database, where business entities have not been properly identified. It could, alternatively, also be a case where your new business object model is in error and the original legacy data model is in fact a better representation of the truth.

**Issue 21 - Referential Integrity Not Enforced**

The most important issue is maintaining referential integrity; in other words ensuring that the two (or more!) tables can be cross-referenced using some primary key/foreign key mechanism. When there is no referential integrity, it will be impossible to retrieve the correct data across multiple tables, and likewise it will be impossible to modify data across all tables when a business domain object changes state.

Cross-table key mapping tables may be introduced to solve this problem, but this leads to database federation.

#### **Issue 22 - Redundant Data**

An important issue is one of data redundancy - when data about a business object is stored in multiple tables, chances are that data is replicated across those tables. During data modification, the new data must be distributed to all tables. During data retrieval, the O/R mapping software must have a data selection mechanism, choosing the "golden" source either as a hardwired choice or based on actual data values e.g. date of last update.

#### **Issue 23 - Conflicting Data Values as Part of Redundancy**

Whenever there is redundant data, there is a risk that the data is not identical in all places. This is a pure data quality issue, but one that is exposed in the O/R mapping algorithm: which data sources is "best" and can be considered the golden source?

Also, whenever such an error is detected, shall the O/R mapping software propagate the "winner" to all outliers automatically or simply flag the errors to a journal, or ignore them?

#### **Issue 24 - Conflicting Data Types as Part of Redundancy**

Whenever there is redundant data, there is a risk that the same data attribute is modeled through different data types, i.e. as a number in one table and as currency in another table. This may lead to situations where the two data types cannot represent the same data domain.

#### **Issue 25 - Conflicting Data Encodings as Part of Redundancy**

This issue is very similar to issue 12: "wrong encoding" discussed earlier. Here, the problem is between multiple code copies across database tables. This is a data modeling error in the original schema.

#### **Issue 26 - Missing Data**

With referential integrity in place, it may still be the case that for some object instances there is no data to be found in one of the underlying tables.

Between two tables, one (the dependent table) would normally contain a foreign key identifying the associated record in the other table (the master table). The FK can be defined as unique and non-null, i.e. the relationship must exist and be of a 1:1 nature. But the same is not true in the other direction - normally the presence of a record in the master table cannot enforce presence of a record in the dependent table (making the relationship 1:0..1). In the object model, the implication is that properties that depend on columns of the dependent table cannot be required object properties.

#### **Issue 27 - Object Properties Spread Over Multiple Tables**

This is an extension of the case where issue 16: "property is derived from multiple columns" which is discussed earlier. Here, the problem is aggravated because of the possibility of issue 26: "missing data" (discussed above, leading to incomplete property values) and issue 22: "redundant data" (including conflicting values, conflicting data types and conflicting value encodings, as per discussions above).

#### **Issue 28 - Use of Encapsulation**

Stored procedures may be used in the legacy database to protect the consistency and completeness of the underlying data. This complicates the work for the O/R mapping framework especially when redundant data must be synchronized across multiple tables, or when dependent data must be joined to master data based on non-indexed data attributes.

#### **Issue 29 - Undocumented Data Models**

Essentially a problem of the legacy database, the lack of schema documentation is a serious setback for the object modelers and the O/R Mapping framework as there is always more risk of misinterpretation.

## **11. Business Object Model Maps to Multiple Databases**

A business object model may well be an aggregation of information held in multiple legacy databases, or in a combination of legacy database(s) plus a new database. A good example is a Customer Master database, and Customer Data Integration is one of the key IT activities for companies wishing to establish a single customer view.

Whenever a business object model is derived from more than just one single database, we use the term database federation. It introduces a new class of problems with a complexity higher than any of the other issues discussed so far. All of those of course still occur in this scenario, and some of them are aggravated such as data redundancy issues and the total lack of referential integrity across databases.

#### **Issue 30 - Referential Integrity Not Enforced**

There is no referential integrity that is enforced between databases; at best there can be a very disciplined use of data identifiers (primary keys) for example an enterprise-wide system for issuing Account Numbers or Air Way Bill numbers. Such identifiers may be present across databases and can be considered good candidate keys for cross-database data references.

Very often, though, it will be necessary to introduce a cross-database key resolution table. Such a table would hold, for a given business object, one column of data holding the primary key value for every database where

the object is represented, extended with one column holding a neutral synthetic identifier. This last column is the only column that can be defined non-null, because of issue 26: “missing data” as described on page 6.

#### **Issue 31 - Object Model Redundantly Mapped to Multiple Databases**

Inevitably, there will be overlap in the information stored in the various databases. This introduces all the elements of redundancy, including conflicting values, conflicting types and conflicting encodings.

In some cases, apparent overlap is in reality not there, for example the product description field in a product database may hold the vendor description in the procurement system but the customer-visible product description in the sales system. These fields are both “product descriptions”, but they should be modeled to two distinct object properties.

#### **Issue 32 - Value Default Conflicts**

Between databases, different choices may have been made regarding default values, and care must be taken to ensure that upon an insert of an object instance that leaves a redundant attribute undefined, the correct default value is posted back to each database. Alternatively, a new default value strategy can be enforced.

#### **Issue 33 - Cross Database Queries**

When an object query is applied in order to select a subset of data from the underlying databases, chances are that the where-clause joins data from multiple databases. Such a query cannot be executed by any one of the underlying databases unless all databases are driven by the same RDBMS (single vendor scenario), and hence the query must be processed by the O/R mapping software, possibly applying complex subquery optimization strategies.

#### **Issue 34 - Cross Database Transactions**

As with cross-database queries, cross-database transactions cannot be executed by any single database. Therefore, the O/R mapping framework must support two-phase commit across all participating databases. The complex case is when an insert in one database generates an identifier that is subsequently needed as a reference in another database. For this to work, the transaction must quite likely be implemented with a compensation mechanism to undo early sub-transactions when later sub-transactions fail.

## **12. Conclusions**

During an application modernization initiative, it is always advisable to consider modernizing the underlying database as part of the project. However, very often new business software applications are added to an existing application portfolio, and these new applications shall co-exist with other applications that make use of the same data. This is in fact a well-known enterprise integration strategy, known as a “shared database”<sup>2</sup>. Of all the database sharing problems and issues discussed in this report, a good number is related to either poor data quality or a poor data model in the legacy databases. Unfortunately, issues in those categories cannot be resolved at the source data level - because both the data and the schema are shared with other applications. Poor data models can be improved using database views, but these are limited to a single database (vendor) and are of limited value when data that is accessed via views must also be updatable.

The two remaining strategies at this point are to either replicate all the data (and transform it on the fly to match a more appropriate database schema newly defined for the modernized application), or to continuously map the data on its way in to and out of the database (using an O/R mapper).

The replication scenario is certainly worth considering, especially when the volatility of the source databases is low. There are issues with data timeliness as the replication may not happen in real time. And, needless to say, replication introduces redundancy by definition, so there is a newly introduced risk of having conflicting data in multiple databases. Moreover, data updates occurring in the replica must be posted back to the original database, leading to complex data integration scenarios. This leaves us with the on-the-fly transformation strategy, which is of value in highly volatile data scenarios as well as scenarios where the new business application is a significant contributor to source data modifications. This report identifies and classifies an extensive range of problems that are likely to happen in the shared database scenario. Software application designers are encouraged to understand these issues and evaluate to what extent their intended O/R mapping strategy is capable of addressing the issues. Software designers with an aspiration to build their own O/R mapping software are encouraged to ensure that the issues identified in this report drive the features and capabilities of their solutions.

<sup>2</sup> See “Enterprise Integration Patterns”, Gregor Hohpe and Bobby Woolf, Addison-Wesley 2004

### 13. Case Study

An MphasiS customer in Europe is underway with a large scale process improvement exercise. Today, various business units work in isolation on copies of business data, and an elaborate EAI system is in place to ensure the required bits of data are properly exchanged between business units. Each BU has its own databases, the total number of databases that hold relevant business data easily exceeds 25. All of these databases contribute to the same business object model for the principal domain object. Millions of object instances reside in the company's databases.

The innovation roadmap calls for a data access gateway that federates across all the participating databases. There is relatively little redundancy in the data attributes held across various systems, but there are inconsistencies in the area of data hierarchies (1:N in one system maps to 1:1 in another system, etc). Data quality is considered to be in reasonably good shape.

The organization historically has opted for different instance identification strategies across the various business units. Some units apply smart identifiers, others use synthetic keys.

The quality of data models varies; some units have well-designed data models that are continuously updated and documented, other business units have immature data modeling skills that have resulted in poor data models.

A perhaps surprising issue in this organization is the difficulty of coming to agreement on the sheer definition of the primary business entity itself, its lifecycle scope and whether it needs to be all derived from one base class with some subclasses or instead as a set of independent business entity types. For every single data attribute, lengthy debates can arise around the attribute definition, its proper name, the allowed values, the desired encoding and the error handling mechanisms when incorrect values are present. To date, there is no consensus as to what the final detailed business object model is going to be.

## 14. Index of Issues

Issue 1 - Column Abuse .....	3
Issue 2 - Column Interpretation .....	3
Issue 3 - Incorrect Derived Data .....	4
Issue 4 - Missing Data .....	4
Issue 5 - Data Lifecycle Mismatch .....	4
Issue 6 - Property is Part of a Column .....	4
Issue 7 - Value Default Conflicts .....	4
Issue 8 - Wrong Data Type .....	4
Issue 9 - Use and Interpretation of Special Characters .....	4
Issue 10 - Required Data Formatting .....	4
Issue 11 - Data Decryption & Encryption .....	4
Issue 12 - Wrong Encoding .....	4
Issue 13 - Undocumented Column Encodings .....	4
Issue 14 - Data Lifecycle Mismatch .....	5
Issue 15 - Data Timeliness Mismatch .....	5
Issue 16 - Property Derived from Multiple Columns .....	5
Issue 17 - Row Subset Mapping .....	5
Issue 18 - Unmapped Required Columns .....	5
Issue 19 - Unmapped Properties .....	5
Issue 20 - Implicit Subtyping .....	5
Issue 21 - Referential Integrity Not Enforced .....	5
Issue 22 - Redundant Data .....	6
Issue 23 - Conflicting Data Values as Part of Redundancy .....	6
Issue 24 - Conflicting Data Types as Part of Redundancy .....	6
Issue 25 - Conflicting Data Encodings as Part of Redundancy .....	6
Issue 26 - Missing Data .....	6
Issue 27 - Object Properties Spread Over Multiple Tables .....	6
Issue 28 - Use of Encapsulation .....	6
Issue 29 - Undocumented Data Models .....	6
Issue 30 - Referential Integrity Not Enforced .....	6
Issue 31 - Object Model Redundantly Mapped to Multiple Databases .....	7
Issue 32 - Value Default Conflicts .....	7
Issue 33 - Cross Database Queries .....	7
Issue 34 - Cross Database Transactions .....	7

## Contact us

### USA

Mphasis  
460 Park Avenue South  
Suite # 1101, New York  
NY 10016, U.S.A.  
Tel: +1 212 686 6655  
Fax: +1 212 686 2422

### UK

Mphasis  
88 Wood Street  
London EC2V 7RS  
Tel: 0208 528 1066  
Fax: 0208 528 1001

Mphasis  
Edinburgh House  
43-51 Windsor Road  
Slough SL1 2EE, UK  
Tel: +44 0 1753 217 700  
Fax: +44 0 1753 217 701

### INDIA

Mphasis  
Bagmane Technology Park  
Byrasandra  
C.V. Raman Nagar  
Bangalore 560 093, India  
Tel: +91 80 4042 6000  
Fax: +91 80 2534 6760

## About Mphasis

Mphasis, an EDS company, delivers Applications Services, Infrastructure Services, BPO and KPO Services through a combination of technology know-how, domain and process expertise. We service clients in the Manufacturing, Financial Services & Insurance, Healthcare, Communications, Media & Entertainment, Transportation & Logistics, Consumer & Retail industries and to Governments around the world. We are certified with ISO 9001:2000, ISO/IEC 27001:2005 (formerly known as BS 7799), assessed at CMMI v 1.2 Level 5 and are undergoing SAS 70 certification. We also provide SEI CMMI, ISO and Six Sigma related services support.

Mphasis is a performance based company, dedicated to outstanding customer service. We offer capabilities to provide innovative solutions by sustainable cost savings and improved business performance through flexible engagement models. Customer centricity, transparency in operations, result-oriented activity and flexibility are the values on which we build long-term relationships with our clients.

