Exposing the Myth: Object-Relational Impedance Mismatch is a Wicked Problem

Christopher Ireland, David Bowers
Department of Maths and Computing
The Open University
United Kingdom
e-mail: cjireland@btinternet.com, D.S.Bowers@open.ac.uk

Abstract—**Addressing a problem of software integration is a fact of life for those involved in software development. The popularity of both object and relational technologies means that they will inevitably be used together. However, the combination of these two technologies introduces problems. These problems are referred to collectively as the object-relational impedance mismatch.**

A mismatch is addressed using one or more mapping strategies, typically embodied in a pattern. A strategy is concerned with correspondence between the schema of a relational database and an object-oriented program. Such strategies are employed in mapping tools such as Hibernate and TopLink, and reinforce the received wisdom that the problem of object-relational impedance mismatch has been solved.

In this paper, we observe that it is not clear whether each strategy, as one possible solution, addresses the cause or a symptom of a mismatch. We argue that the problem is not tame and easily resolved; rather it is complex and wicked. We introduce a catalogue of problem themes that demonstrate the complex nature of the problem and provide a way both to talk about the problem and to understand its complexity.

In the future, as software systems become more complex and more connected, it will be important to learn from past endeavours. Our catalogue of problem themes represents a shift, in thinking about the problem of object-relational impedance mismatch, from issues of implementation towards an analysis of cause and effect. Such a shift has implications for those involved in the design of current and future software architectures. Because we have questioned the received wisdom, we are now in a position to work toward an appropriate solution to the problem of object-relational impedance mismatch.

Keywords—object-relational; impedance mismatch; wicked problem; problem theme

I. INTRODUCTION

Addressing a problem of software system integration is a fact of life for those involved in software development [26], p46. Typically, an organization will employ a number of software systems, possibly written using different programming languages, each to a separate design, and running on different operating systems on different hardware platforms. Each software system will support different facets of the organisation's business activities.

An object-relational application is a software system that combines technologies based on the concepts of both " object" and "relation". Object-relational impedance mismatch is the term we use to refer to a difference between the schema of an object-oriented program and the schema of a relational database. Despite the received wisdom that the problem of object-relational mismatch has been "solved", reinforced by technologies such as Hibernate [2], TopLink [3] and LINQ [4], the resolution of a mismatch typically involves some form of object-relational mapping, and costs significant time and effort to address.

In this paper, we explore the nature of object-relational impedance mismatch. We demonstrate that, contrary to the received wisdom, the problem of object-relational impedance mismatch is not tame and easily resolved but, rather, it is wicked and complex. We provide a new way both to talk about the problem and to understand its complexity. Such understanding provides a sound foundation for work toward an appropriate solution to the problem.

This paper is structured as follows: in Section II. we illuminate the received wisdom. In Section III we expose the wicked nature of impedance mismatch. In Section IV we introduce a catalogue of problem themes as a lens through which we can understand the problem. In sections V and VI we use problem themes to demonstrate the complex nature of impedance mismatch and expose relationships between themes. Finally, in Section VII, we set out the limitations of a perspective on impedance mismatch based on problem themes, before presenting our conclusions and proposing future work in Section VIII.

II. OBJECT-RELATIONAL IMPEDANCE MISMATCH

An object-relational mismatch can occur only when an object-oriented program uses a relational database for persistence. A mismatch between an object-oriented program and a relational database does not materialise until a particular mapping strategy is selected. An object-relational mapping strategy (mapping strategy) sets out the correspondence between classes in the schema of an object-oriented program and the schema of a relational database. An object-relational application comprises many such strategies. However it is not always clear what each strategy addresses: the cause of or a symptom of a mismatch.

The problem of object-relational impedance mismatch is important in practice because addressing a mismatch costs both time and effort. Contrary to the suggestion of [1], the decoupling of a program and a database does not resolve a mismatch. Problems still occur at the point where objects and relations are combined, and they do not go away simply because a persistence layer [2][3] or a hybrid language [4] is used. A persistence layer embodies a number of mapping strategies. Such a layer will only address the cause of a mismatch if the mapping strategy employed is an *appropriate* solution.

In contrast we define an *acceptable* solution as one that gives the illusion that a mismatch is solved even if it addresses only a symptom of the mismatch. Such a solution might reinforce a belief that the mismatch has been avoided even though its cause has not been addressed. However, it should not be concluded that a mismatch is inevitable and that there is no alternative but to deal just with the symptoms. In the next section we explore the misconception that underpins this received wisdom by demonstrating the true nature of the problem of object-relational impedance mismatch.

III. A WICKED PROBLEM

Rittel and Webber [5] observe that some problems cannot be resolved in a linear way. They label such problems as wicked. A tame problem is particularly suited to a linear resolution because it is well defined; it has a clear and well-defined stopping point, when a solution is found from a list of possible solutions; and it is possible to choose a solution because there is a set of pre-defined criteria for making such a choice.

A wicked problem is less straightforward. Rittel and Webber describe ten characteristics of a wicked problem. In essence, a wicked problem is a problem that resists resolution because its definition is incomplete, the requirements of multiple stakeholders change, and there is no single definitive and optimal solution, so a choice of solution typically involves a compromise.

The concepts of tame and wicked problems represent the two extremes of a continuum along which a given problem may be positioned, depending on its particular characteristics. The characteristics of a wicked problem were derived from work on planning policy in the 1960s. Conklin [6], p21 subsequently refined them so they could be applied to areas other than planning policy. In Table I, we use each of Conklin’s characteristics to explore the extent to which the problem of object-relational impedance mismatch may be considered wicked.

Object-relational impedance mismatch is an exemplar of a wicked problem. There is no single problem or solution. Each problem involves a number of stakeholders both within and outside an organisation, such as programmers, designers, analysts, software vendors and language designers. Furthermore, a problem is not addressed in isolation. The solution to a problem is a mapping strategy but each strategy involves a compromise because data about an object will not fit neatly into the schema of a relational database. Consequently, a choice of a particular strategy may cause another problem.

The search for a solution involves accepting compromises (or satisficing [6], p14). The result is a mapping strategy that produces the best fit rather than the optimal fit, and a solution that is somehow acceptable rather than appropriate. Furthermore, any choice of solution has implications for the design of an object-relational application. Consequently, we can think of an object-relational application as a complex collection of interrelated problems; what Ackoff [11] termed a *mess*.

Thinking about object-relational impedance mismatch as a wicked problem raises new questions about how we understand and address a mismatch. Such questions (Table I) expose issues with the received wisdom that the problem of object-relational impedance mismatch has been solved. We present next a new vocabulary to describe the problem of impedance mismatch. This vocabulary provides a way both to structure the mess and to understand the complex nature of the problem.

TABLE I. OBJECT-RELATIONAL IMPEDANCE MISMATCH FRAMED AS A WICKED PROBLEM

Characteristic of a Wicked Problem [6]	The problem of Object-Relational Impedance Mismatch
You don’t understand the problem until you have developed a solution. Every solution exposes new aspects of the problem. There is no single definition of the problem instead an interlocking set of issues and constraints from different stakeholders.	There is no mismatch between an object-oriented program and a relational database until a decision is made to use a particular mapping strategy. There are many mismatches and there are many mapping strategies each of which may be a potential solution. Each solution involves a compromise [7]. There are issues such as those of a consistent identity and the preservation of semantics [8]. How do we understand the nature and consequence of a compromise?
Wicked problems have no stopping rule. There is no single definition of the problem and so there is no definitive solution.	A problem does not exist in isolation and a solution to one problem may cause another problem. There are a number of object, relational, and mapping technologies. A solution is chosen based on some criteria [9] but how do we know that these criteria are appropriate for making such a choice?
Solutions are not right or wrong simply better/worse/good enough. Stakeholders each interpret the solution based on their objectives.	Each solution involves a compromise either in the design of a program or in the design of a database. For example, Ambler [7], Chapter 14 lists a number of pros and cons for each mapping strategy. How then do we make an informed choice of an appropriate solution?
Each problem is essentially unique and novel because there are so many factors and conditions.	There are a number of mapping strategies but each must be interpreted in the context of a particular object-relational application. On the surface, defining a mapping strategy appears to be a straightforward activity. For example, a class corresponds to a table and an attribute corresponds to a column, but as [10] observes, a quagmire of issues rapidly develops. How then do we understand and avoid this quagmire?
Every solution is a one-shot operation. It has consequences and changes the context.	A choice of solution impacts the design of a program and the design of a database. Once a particular mapping strategy is implemented in an object-relational application how easy it is to adopt a different strategy?
There are no given alternative solutions. It is a matter of creativity to devise new solutions and a matter of judgement to decide which are valid and worth pursuing.	A one-solution-fits-all approach may not be appropriate but to what extent do we accept the available mapping strategies as a given? Are there other possibilities for a solution outside the code of an object-relational application?

IV. PROBLEM THEMES

Neward [12] refers to the problem of an object-relational impedance mismatch as “a quagmire of issues”. In this section we set out to understand the problem of object-relational impedance mismatch. The objective is to make sense of the problem and the different interpretations of object-relational impedance mismatch.

Copeland & Maier [13], Neward [12] and Ambler [7], p105-113 each characterise object-relational impedance mismatch in a different way. Copeland & Maier are concerned with issues of concept and data structure, Neward focuses on problems of implementation and Ambler is concerned with technical and cultural difficulties.

It is not clear how each characterisation relates to the others, whether a particular characterisation refers to the cause of a mismatch or a symptom, whether the list of characterisations is complete, or why each characterisation was chosen. Ambler and Neward consider issues beyond those of technology but it is not clear whether Copeland & Maier, Ambler and Neward describe the cause or a symptom of a mismatch.

Each characterisation draws attention to a collection of mismatches that together represent a particular problem. In this section, we consolidate the characterisations described by Copeland & Maier, Neward and Ambler, along with contributions from others, as a catalogue of problem themes. An early version of this work can be found in [14][15].

A problem theme is defined as a collection of mismatches. A problem theme reflects a particular characterisation, such as the "object-to-table mapping problem" and the "schema ownership problem" described by Neward, or the "cultural impedance mismatch" described by Ambler, and helps to make sense of a collection of mismatches. A mapping strategy is one solution to a mismatch. It follows that a mapping strategy is also (part of) one solution to a problem theme. The relationships between a theme, a mismatch and a mapping strategy are summarised in Figure 1.

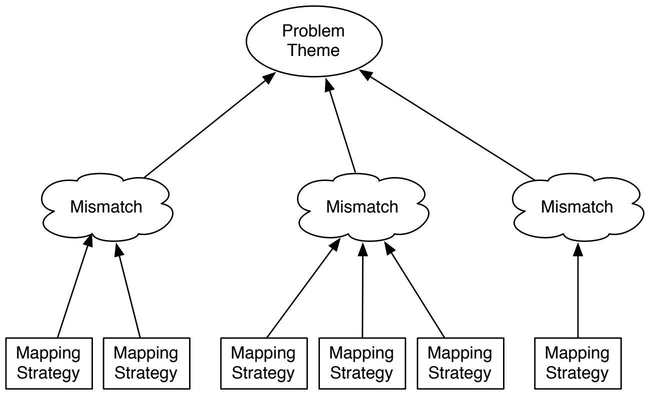

Figure 1. A Problem Theme, Mismatches and Mapping Strategies

A problem theme is important for two reasons. A problem theme provides a way to understand one aspect of object-relational impedance mismatch. It makes it possible to talk about and focus on a specific problem rather than use the general term object-relational impedance mismatch. Table II summarizes the concern of each problem theme.

In summary, a problem of object-relational impedance mismatch displays the characteristics of a number of problem themes. Problem themes provide a way to understand impedance mismatch. Each theme is concerned with a collection of mismatches. In the context of a problem theme it is possible to talk about the problem of a specific subtype of object-relational impedance mismatch. In the next section, we use the catalogue of problem themes to move toward an understanding of the complex nature of impedance mismatch.

TABLE II. PROBLEM THEMES

Problem Theme	Concern
Structure	The structure problem theme is concerned with any difference of data structure between the schema of an object-oriented program and the schema of a relational database, and so adopts a broad interpretation of the notion of structure. The essence of a structure problem is the extent to which an object-oriented data structure can be, and should be, described by a relational data structure. Problems of the structure theme are important because they are concerned with a description of the data processed by an object-relational application.
Instance	The essence of an instance problem theme is, where is the canonical copy of state located? Problems of the instance theme are important because they are concerned with the ownership of and the responsibility for data.
Encapsulation	The principle of encapsulation requires that the state of an object can be determined only by its behaviour, so in an object-oriented program the value of an attribute of an object is accessed via a method. Problems of encapsulation are important because, in a database, the value of a column in a row has no such protection. Consequently, once stored in a database, data may be changed without the protection of the semantics encoded in a method.
Identity	The essence of an identity problem is how to identify uniquely a collection of data values between both object-oriented program and a relational database. Such problems of identity are important to ensure the integrity of data between an object-oriented program and a relational database.
Processing Model	The essence of a processing model problem is how to represent in, maintain and retrieve from a database a sufficient set of objects for processing. Such problems are important because they concern issues of software performance [16].
Schema Ownership	The essence of the schema ownership problem is that the team who design and implement an object-oriented program can be different from the team who design and implement a relational database. Such problems are important because they concern the choices made by those responsible, respectively, for the object-oriented program and the relational database.

V. A COMPLEX MIX OF PROBLEMS

Problem themes classify mismatches that must be addressed during the development of an object-relational application. However, such concerns are not independent. We explore in this section relationships between problem themes. Each relationship is causal; collectively they describe the complex nature of object-relational impedance mismatch.

A structure problem can be the consequence of an ownership problem. A conceptual mismatch and a structural mismatch, as described by Copeland & Maier, are not independent. A conceptual framework determines the semantics of a language, so a language such as Java is referred to as an object-oriented language. Those who implement an object-relational application make a choice of abstraction but can use only the artefacts of a particular language to describe that abstraction.

Neward [12] refers to “the schema ownership conflict” and “the dual schema problem”. Each demonstrates a relationship between a schema ownership problem and a structure problem. The schema ownership conflict describes a mismatch of agenda. For example, a performance issue might mean that those responsible for a database have to change a data structure [17]. The dual schema problem occurs when a database must be changed in order to accommodate another application. In this case a structure problem is caused by a solution to a schema ownership problem.

A choice of abstraction made in the design of one application can produce a structure problem in another. Keller [16] helps to reinforce a link between the schema ownership problem and the structure problem. Whilst Neward is concerned with accommodating a new application, for Keller the problem is incorporating an existing data structure, from another application, into the schema of an object-oriented program.

A schema ownership problem can cause an instance problem. Differences in perception between stakeholders of the role of a program and a database in an object-relational application bring into question the location of a canonical copy of state. A solution to a schema ownership problem must reconcile these different perceptions.

Problems of structure and identity are related. A choice of language will decide the data structure to which an identity refers. For example, in Java, an object has an identity whilst in SQL the value of a primary key represents the identity of a row. In order to address an identity problem it is important to be clear about the structure to which an identity refers.

However, a solution to an identity problem can then cause a structure problem. Keller [16] describes a solution to a correspondence of identity between the schema of an object-oriented program and the schema of a relational database. He addresses an identity problem by introducing a surrogate identifier, resulting in the need for a change to the structure of a database schema.

An instance problem can cause an encapsulation problem. Once data has been stored in a database, that data may be modified independently of the logic employed in a program. Such a change can occur if the instance problem is caused by a schema ownership problem whereby those responsible believe that a database maintains the canonical copy of state.

A structure problem can lead to an encapsulation problem. Lodhi [18] observes that the way an association is represented in a relational database can be different from an association between two objects. Consequently the representation of an association between objects as a foreign key in a relational database does not necessarily preserve the encapsulation of an object.

A structure problem can also lead to a processing model problem. The process of normalisation can cause data about an entity to be split across a number of tables. Consequently, in the context of a reference between two objects, in order to retrieve the data for a referenced object it may be necessary to join a number of tables.

An instance problem can be caused by a processing model problem. It may not be necessary to retrieve or store all the data about an object in order to satisfy a request. However it may still be necessary to retrieve all the data for an object in order to create that object. As a result those responsible for an object-oriented program might believe that a program maintains the canonical copy of state.

An encapsulation problem can lead to a processing model problem. In order to reference an object, a program must first create an object. It may not be desirable or practical to load data about all objects in a network from a relational database so a decision must be made at which point to stop. That decision is difficult because the network of references between objects is encapsulated within the objects themselves.

In summary, object-relational impedance mismatch is a complex mix of interrelated problems. Using problem themes we have explored this complexity and demonstrated that solving problems of any particular class can generate problems of another. Consequently each such problem cannot be addressed in isolation. In the next section we use these relationships to understand the mess and to explore the consequences for our understanding of impedance mismatch.

VI. Relationships Between Problem Themes

The previous section demonstrated that problem themes are related. These relationships are summarised in Figure 2.

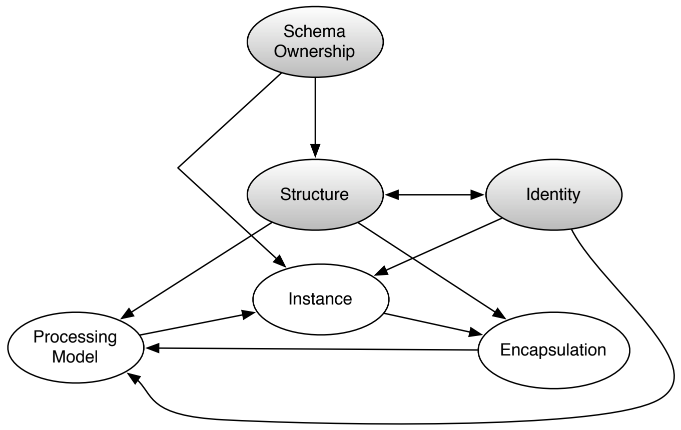

Figure 2. Problem Themes and Relationships

Two themes are related if a solution of one theme leads to a problem of another theme. The arrows on each line in Figure 2 indicate the direction of influence between two problem themes. It is possible to talk about a specific problem such as that of schema ownership, structure or identity and see that such problems are related. For example the line from the problem theme of structure to the problem theme of identity indicates that a solution to a structure problem can have a consequence for an identity problem.

Because it is possible to make a connection between themes it is also possible to explore the complex nature of object-relational impedance mismatch. For example, in order to address an identity problem it might be necessary to first address a structure problem, but a structure problem might be caused by a schema ownership problem. By exposing such relationships between themes it is possible to begin to understand the problem of object-relational impedance mismatch in a systematic way.

Because the solution to one problem can cause another problem, it follows that a problem should not be considered in isolation. The relationships between problem themes help us to understand problems of object-relational impedance mismatch.

Figure 2 shows that a structure problem can be caused by a schema ownership problem. It is important that a solution to a structure problem involves both those responsible for an object-oriented program and those responsible for a relational database. Because a solution to one problem can lead to another, the relationships between themes provide a way for those responsible for a solution to understand with whom to consult when assessing its consequences.

Figure 2 also shows that a schema ownership problem is related to a structure problem but a structure problem has a consequence for a number of other problem themes. Similarly an identity problem has a consequence for a number of problem themes. The identity problem and the structure problem are also related. An understanding of the structure problem, the schema ownership problem and the identity problem is therefore important to understanding object-relational impedance mismatch.

The importance of a structure problem is reflected in the many mapping strategies between the schema of an object-oriented program and the schema of a relational database. Many authors describe a correspondence of structure between the schema of an object-oriented program and the schema of a relational database. A mapping strategy is based on a perceived correspondence, such as that between a class and a table (for example [18][19][20]); a class hierarchy and a table or a collection of tables (for example [7][21][22][23]); a relationship and a foreign key or a table (for example [7][19][24]); and an aggregation and a table or a column (for example [16][25]). In order to understand such a choice of mapping strategy and whether it results in an appropriate or an acceptable solution first the cause of a mismatch of structure must be understood.

There is a cycle in Figure 2. A solution to a structure problem can cause an encapsulation problem. A solution to an encapsulation problem can cause a processing model problem. A solution to a processing model problem can cause an instance problem. A solution to an instance problem can cause another encapsulation problem. Because there are different solutions to a problem a choice of solution must be made.

A choice of a mapping strategy can break a cycle if that solution does not cause another problem. For example Shadow Information [7], p228 introduces a change in the structure of an object-oriented program that addresses an instance problem. The implication is that it is important to understand the consequences of a mapping strategy as well as the artefacts involved in a correspondence.

In order to address one mismatch it may be necessary to address another problem first. A Synthetic Object Identity [16],p21 is a surrogate identifier used in a number of mapping strategies. In order to address an identity problem a Synthetic Object Identity introduces a change of structure. The question remains whether this change of structure addresses the real cause of a mismatch of identity, and so is an appropriate solution, or whether the change of structure deals with the symptoms and so is an acceptable solution. To answer that question first the cause of a mismatch of identity must be understood.

In summary, relationships between problem themes can be used to visualise the mess, or what Neward referred to as a quagmire. Exploring relationships between problem themes demonstrates that object-relational impedance mismatch is a complex problem, illuminates problems of particular importance, and highlights that there are consequences from a choice of solution. In the next section we highlight the limitations of a perspective based on problem themes.

VII. The Limitations of Problem Themes

The problem themes represent a consolidation of the work of others. However it is not clear whether they identified all possible problems and explored all possible relationships, or whether that was in fact their objective. It is also not clear from the literature whether their categorisations of the problem are simply observations based on experience or an exhaustive search of the problem.

Copeland & Maier talk in general terms of concept and structure, whereas Neward is concerned with specific problems such as retrieving data for an object from a database. Whilst the categorisation of Neward appears more comprehensive than that of Copeland & Maier, because it describes more problems, the level of abstraction can explain such a difference. Consequently the catalogue of problem themes, the relationships between themes, and the categorisations of Copeland & Maier, Neward and Ambler must be considered as partial but illustrative of the problem of object-relational impedance mismatch.

Relationships between problem themes cannot be used to locate the cause of a mismatch. In order to locate the cause of a mismatch it is necessary first to explore the reason for that mismatch between the schema of an object-oriented program and the schema of a relational database. The reason for a mismatch does not lie in a relationship between two problem themes. For example, the answer to an identity problem is not found by understanding that it may be caused by a mismatch of structure. Why there is a mismatch of structure must be understood first.

Choices of transformation in the design of an object-oriented program and a relational database provide the context for a mismatch. One mismatch is that of a data structure. Differences of language and abstraction lead to such mismatches, but a conceptual framework, respectively those of an object and a relation, underpins each language and each abstraction. Using problem themes it is clear that a problem of structure can have consequences for other problem themes, but it not clear whether a choice of abstraction, language or conceptual framework is the root cause of a mismatch of structure. In [15], we describe a framework, based on these choices, for exploring the cause of a mismatch.

VIII. Conclusions and Future Work

Contrary to the received wisdom, we do not know whether a solution addresses the cause or a symptom of an object-relational impedance mismatch. A mapping strategy is simply a pragmatic solution to a problem in the implementation of an object-relational application. Because we do not know the cause of a particular mismatch, we cannot be sure whether such a solution is appropriate or whether it is somehow acceptable.

Object-relational impedance mismatch is a wicked problem, and we introduce problem themes as a way of making sense of such mismatches. Our catalogue of problem themes provides a new vocabulary for describing the problem of object-relational impedance mismatch. Each problem theme focuses attention on a particular aspect of an object-relational impedance mismatch. Problem themes also provide a structure to the problem and demonstrate the complex nature of object-relational impedance mismatch.

Problem themes provide an insight into distinct, but interacting, aspects of object-relational impedance mismatch. Problem themes have implications for those developing an object-relational application. Relationships between themes expose the "quagmire of issues" referred to by Neward and demonstrate that those developing an object-relational application must think about issues of more than one theme in the design and implementation of a mapping strategy.

Our catalogue of problem themes suggests a shift in thinking about the problem of object-relational impedance mismatch from issues of implementation towards an analysis of cause and effect. Because we have questioned the received wisdom, we are in a position to work towards appropriate solutions to problems of object-relational impedance mismatch. Future work might explore also the extent to which such a shift in thinking provides a way to illuminate other issues of software integration.

The problem of object-relational impedance mismatch involves a number of stakeholders. Our own future work will concentrate on identifying a suitable mechanism to engage those responsible for the design and implementation of an object-relational application in an effective dialogue about a problem and its cause.

REFERENCES

[1] M. L. Fussell, "Foundations of Object Relational Mapping," 17th March 2015, 2007; http://www.database-books.us/databasesystems_0003.php.

[2] Hibernate. 17th March 2015; www.hibernate.org.

[3] TopLink. 17th March 2015; http://www.oracle.com/technetwork/middleware/toplink/overview/index.html.

[4] J. Schwartz and M. Desmond, "Looking to LINQ," 17th March 2015, 2007; http://adtmag.com/articles/2007/04/04/looking-to-linq.aspx.

[5] H. Rittel and M. Webber, "Dilemmas in a General Theory of Planning," Policy Sciences, vol. 4, pp. 155-169, 1973.

[6] J. Conklin, Dialogue Mapping - Building Shared Understanding of Wicked Problems, Chichester, England: Wiley, 2006.

[7] S. W. Ambler, Agile Database Techniques - Effective Strategies for the Agile Software Developer: Wiley, 2003.

[8] C. Ireland, "Object-Relationla Impedance Mismatch: A Framework Based Approach," Mathematics and Computing, Open University, Milton Keynes, 2011.

[9] F. Marguerie. "Choosing an object-relational mapping tool," 17th March 2015, 2007; http://madgeek.com/Articles/ORMapping/EN/mapping.htm.

[10] T. Neward, "Avoiding the Quagmire," 17th March 2015; http://www.odbms.org/wp-content/uploads/2007/05/031.02-Neward-Avoiding-the-Quagmire-May-2007.pdf.10

[11] R. Ackoff, "Systems, Messes, and Interactive Planning," Redesigning the Future, New York: Wiley, 1974.

[12] T. Neward, "The Vietnam of Computer Science," 17th March 2015; http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx.

[13] G. Copeland, and D. Maier, "Making Smalltalk a database system," ACM SIGMOD Record, vol. 14, no. 2, June 1984, pp. 316-325.

[14] C. Ireland, D. Bowers, M. Newton, Waugh, K., "A Classification of Object-Relational Impedance Mismatch," Proc. of The First International Conference on Advances in Databases, Knowledge and Data Applications. March 2009, pp. p36-43.

[15] C. Ireland, D. Bowers, M. Newton, Waugh, K., "Understanding Object-Relational Mapping: A Framework Based Approach," International Journal On Advances in Software, vol. 2, no. 2, 2009, pp. 202-216.

[16] W. Keller, "Mapping Objects to Tables: A Pattern Language," Proc. of European Conference on Pattern Languages of Programming Conference (EuroPLoP), 1997

[17] G. L. Sanders, and S. Seungkyoon, "Denormalisation effects in performance of RDBMS," in 34th Annual Hawaii International Conference on System Sciences, Hawaii, January 2001, pp. 9.

[18] F. Lodhi, and M. A. Ghazali, "Design of a simple and effectve object-to-relational mapping technique," Proc. of ACM Symposium on Applied Computing, March 2007, pp. 1445-1449.

[19] K. Brown, and B. G. Whitenack. "Crossing Chasms: A Pattern Language for Object-RDBMS Integration "The Static Patterns"," 30 December 2008; http://www.ksc.com/articles/staticpatterns.htm.

[20] S. Philippi, "Model driven generation and testing of object-relational mappings," Systems and Software, vol. 77, 2005, pp. 193-207.

[21] U. Hohenstein, "Bridging the Gap between C++ and Relational Databases," Proc. of European Conference on Object-Oriented Programming. 1996, pp. 398-420.

[22] L. Cabibbo, and A. Carosi, "Managing Inheritance Hierarchies in Object/Relational Mapping Tools," Lecture Notes in Computer Science, vol. 3520, 2005, pp. 135-150.

[23] M. Pizzo, "An Application-Oriented Model for Relational Data," The Architecture Journal, no. 12, July 2007, pp. 19-25.

[24] L. Cabibbo, and R. Porcelli, "M2ORM2: A Model for the Transparent Management of Relationally Persistent Objects," Proc. of Database Programming Languages: 9th International Workshop. 2003, pp. 166 - 178.

[25] C. Russell, "Bridging the Object-Relational Divide," ACM Queue, vol. 6, no. 3, 2008, pp. 18-28.

[26] K. Roebuck, Object-Relational Mapping: High-impact Strategies - What You Need to Know: Emereo Pty Limited, 2011.