A Viewpoint-Based Framework for Discussing the Use of Multiple Modelling Representations

Nigel Stanger

University of Otago, Dunedin, New Zealand
nstanger@infoscience.otago.ac.nz

Abstract. When modelling a real-world phenomenon, it can often be useful to have multiple descriptions of the phenomenon, each expressed using a different modelling approach or *representation*. Different representations such as entity-relationship modelling, data flow modelling and use case modelling allow analysts to describe different aspects of real-world phenomena, thus providing a more thorough understanding than if a single representation were used. Researchers working with multiple representations have approached the problem from many different fields, resulting in a diverse and potentially confusing set of terminologies. In this paper is described a viewpoint-based framework for discussing the use of multiple modelling representations to describe real-world phenomena. This framework provides a consistent and integrated terminology for researchers working with multiple representations. An abstract notation is also defined for expressing concepts within the framework.

1 Introduction

In this paper is described a framework for discussing the use of multiple modelling approaches or *representations* to describe a real-world phenomenon. This framework is derived from work on *viewpoint-oriented design*, and provides a consistent and integrated terminology for researchers working with multiple modelling representations. An abstract notation for expressing various concepts within the framework is also defined.

Why is it useful to describe the same phenomenon using multiple modelling representations? There are several reasons, such the ability to provide a more complete description of the phenomenon in question, and because some representations are better suited to particular problems than others. The use of multiple representations and associated issues are discussed in Sect. 2.

The framework described in this paper is primarily derived from earlier work in the area of viewpoint-oriented design methods. The basic concepts of viewpoint-oriented design methods are introduced in Sect. 3, and a lack of clarity is identified with respect to the definitions of some

fundamental concepts (in particular, the meaning of the term 'representation').

The author's framework arose out of the need to clarify the definitions of various terms and also to integrate and simplify the potentially confusing range of terminologies developed by other authors. The framework is discussed in Sect. 4. In particular, the terms 'representation', 'technique' and 'scheme' are clarified, and the new terms 'description', 'construct' and 'element' are defined.

The author has also developed an abstract notation for expressing various framework concepts in a concise manner. This notation is defined in Sect. 5.

The paper is concluded in Sect. 6.

2 Using multiple representations to describe a phenomenon

There are many different types of information to be considered when designing an information system, and a wide variety of modelling approaches and notations have been developed to capture these different types of information: entity-relationship diagrams (ERDs), data flow diagrams (DFDs), use case diagrams, the relational model, formal methods and so on. Problems can arise when useful information is omitted from a design. Consider an information system whose data structures are designed using entity-relationship diagrams and are implemented in a relational database. Data entry forms derived from these models are built using a rapid application development tool. Good design practices are followed throughout, yet the finished application is difficult to use. Some of the commonly used data entry forms have multiple states, but the transitions between these states are unclear to users because state information was not included in the system design.

While this is a purely theoretical example, it serves to illustrate an important point. Information systems are typically built to handle the data processing requirements of some real-world phenomenon. Such real-world phenomena may often be too complex to describe using a single modelling approach, or *representation*. This is supported by the plethora of different representations that currently exist [26, 42], including those that model the structure of data (such as entity-relationship modelling), and those that model how data move around a system (such as data flow diagrams). This implies that in order to completely model a phenomenon,

multiple descriptions of the phenomenon are required, expressed using different representations.

Using multiple representations to describe a phenomenon is also important in other ways:

- If multiple developers are working on a project, each may prefer or be required to use a different representation to describe their particular part of the project [2]. Meyers [30] also notes that it can be very useful to simultaneously view a system in several different ways when multiple people are working on the system.
- Particular subproblems may be better described using some representations than others [15].
- Multiple representations are important when integrating heterogeneous data sources to form a federated or distributed system [2], as each data source may potentially use a different logical data model.

The idea of using multiple representations to model a phenomenon is not new. Grundy et al. have been examining the issues associated with building multi-view editing systems and integrated software development environments for many years [20, 21, 23], with emphasis on the issue of maintaining consistency between different views or descriptions of the same phenomenon [22, 24]. Grundy's work was derived from earlier work on multi-view editing systems for software engineering, such as FIELD [35] and Zeus [5].

Atzeni and Torlone [3] suggested the idea of translating between different representations as a means of facilitating the use of multiple representations. They proposed a formal model based on lattice theory [1] that allowed them to express many different data modelling approaches using primitive constructs of a single underlying representation.

Su et al. [40, 41] approached the use of multiple representations from the point of view of integrating heterogeneous data sources in order to build federated and distributed databases. Their approach is similar in many respects to that taken by Atzeni and Torlone, except that the underlying representation is object-oriented rather than mathematically based.

All three groups developed their work independently of each other over a similar time period (1991–94), and each approached the use of multiple modelling representations from a different starting point. This has resulted in a diverse and potentially confusing set of terminologies (see Table 1). A single integrated terminology would reduce this potential for confusion.

In addition to the three groups outlined above, viewpoint researchers first suggested using multiple representations to describe viewpoints over

Table 1. Comparison of viewpoint/representation terminologies

Term used by the author	Meaning	Example	Corresponding term used by: Finkelstein [17]	Easterbrook [15]	Darke & Shanks [11]	Grundy *et al.* [21]	Atzeni & Torlone [3]	Su *et al.* [41]
perspective	A description of a real-world phenomenon that has internal consistency and an identifiable focus.	–	–	perspective	perspective	–	–	–
viewpoint	The formatted description of a perspective.	–	ViewPoint	viewpoint	viewpoint	–	–	–
technique	A collection of abstract constructs that form a modelling 'method'.	relational model	style	style	technique	–	model	data model
scheme	A collection of concrete constructs that form a modelling 'notation'.	SQL/92			scheme	–		
representation	The combination of a particular technique and scheme.	relational model + SQL/92			representation	representation		
description	An instantiation of a representation.	SQL/92 schema	specification	description	–	view	scheme	schema
construct	The basic unit of a representation.	a relation	–	–	–	–	construct	class[a]
element	An instantiation of a construct within a particular description.	Staff table	–	–	–	component	varies[b]	object

Notes on Table 1:
[a] Also 'construct'.
[b] Terms used include 'component', 'element' and 'concept'.
'–' indicates that a term is not used by that author.

a decade ago [17]. A viewpoint is effectively a formalisation of the perceptions of stakeholders with respect to some real-world phenomenon. Since we are dealing with the use of multiple representations to describe real-world phenomena, viewpoints should provide a useful framework within which to discuss such use [39]. The author's framework will be described in Sect. 4, but first the basic concepts of viewpoint-oriented methods must be defined.

3 Viewpoint concepts

A *viewpoint* can be thought of as a formalisation of the perceptions of a stakeholder group with respect to some real-world phenomenon that is being modelled. The first viewpoint-oriented approach (Mullery's CORE method) was introduced in 1979 [32], but the concept of a viewpoint was not formalised until ten years later [17].

Viewpoint-oriented methods were originally developed to assist with requirements definition in a software engineering environment [32], and subsequent research has followed a similar direction [15, 16, 28, 33]. The focus of the author's research has been on how to facilitate the use of multiple representations to describe a single viewpoint [38, 39].

In Fig. 1 are shown the relationships between the concepts of viewpoint-oriented methods. This initial framework was derived by the author [39] from the work of Finkelstein et al. [17], Easterbrook [15] and Darke and Shanks [11]. Their terminologies are also summarised in Table 1.

3.1 Perspectives and viewpoints

Easterbrook [15] defines a *perspective* as "a description of an area of knowledge which has internal consistency and an identifiable focus of attention". During the requirements definition phase of systems analysis, developers may encounter many different perspectives on the problem being modelled. Perspectives may overlap and/or conflict with each other in various ways.

Finkelstein et al. [17] describe a *viewpoint* as comprising the following parts:

- "a *style*, the representation scheme in which the ViewPoint [*sic*] expresses what it can see (examples of styles are data flow analysis, entity-relationship-attribute modelling, Petri nets, equational logic, and so on);

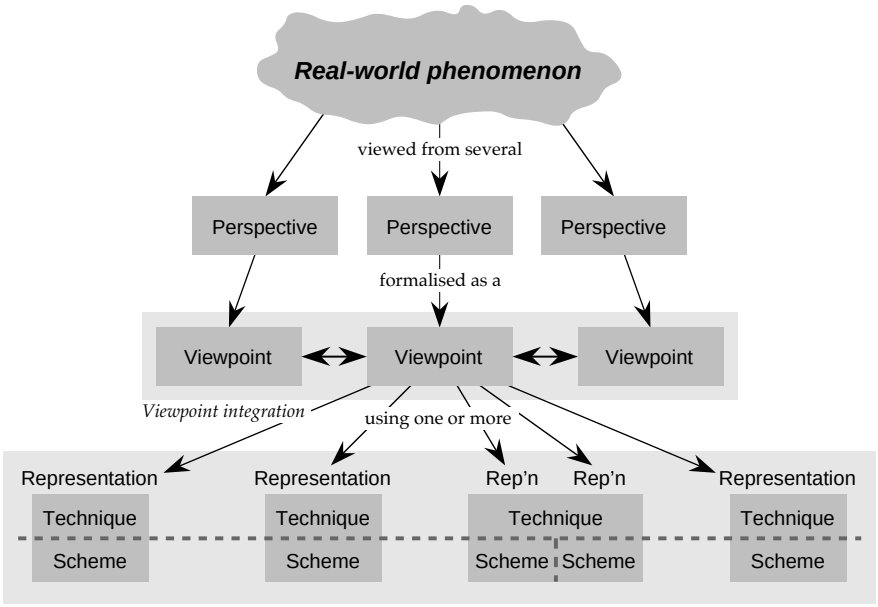


Fig. 1. Relationship between perspectives, viewpoints and representations

- a *domain* defines which part of the 'world' delineated in the style (given that the style defines a structured representation) can be seen by the ViewPoint (for example, a lift-control system would include domains such as user, lift and controller);
- a *specification*, the statements expressed in the ViewPoint's style describing particular domains;
- a *work plan*, how and in what circumstances the contents of the specification can be changed; [and]
- a *work record*, an account of the current state of the development."

Easterbrook [15] simplifies this description by defining a viewpoint as "the formatted representation of a perspective", and notes that a perspective is a "more abstract version of a viewpoint". In effect, a viewpoint is the formalisation of a particular perspective, so there is a one-to-one correspondence between a viewpoint and the perspective it formalises, as illustrated in Fig. 1.

The term 'viewpoint' is very similar to the term 'view' as used in multi-view editing systems [5, 20, 27, 30]. These terms refer to different concepts, however: a 'view' is more akin to the concept of a *description*, which will be introduced in Sect. 4. The similarity of the two terms has

led to some confusion: the terms 'viewpoint' and 'view' have been used interchangeably in the past [27].

Darke and Shanks [11] define two main types of viewpoint:

1. *user viewpoints* that capture "the perceptions and domain knowledge of a particular user group, reflecting the particular portion of the application domain relevant to that group"; and
2. *developer viewpoints* that capture "the perceptions, domain knowledge and modelling perspective relevant to a systems analyst or other developer responsible for producing some component of the requirements specification".

Since a viewpoint is the formalisation of a perspective, some form of model is required to provide the formalised structure. The concept of a *representation* provides this.

3.2 Representations

Darke and Shanks [10] note that viewpoints may be described using different representation *techniques*, within each of which there may be available a number of representation *schemes*. Neither Darke and Shanks nor Finkelstein et al. [17] clearly define the terms 'representation', 'technique' or 'scheme'; rather, they introduce each term by means of examples. This has led to some confusion in the use of this terminology. Darke and Shanks use the terms 'representation' and 'representation technique' interchangeably, while Finkelstein et al., as can be seen in their definition of a 'style', use the term 'representation scheme' in a similar way.

The intent appears to be that a *representation* should be thought of as a structured modelling approach that can be used to describe the content of a viewpoint. In order to clarify the confusion in terminology, the author has refined this informal definition and defined a representation as the combination of a particular technique and scheme to describe a viewpoint. This will be discussed further in Sect. 4.

Darke and Shanks [11] group representations into three general categories:

1. *informal* representations that form unstructured descriptions, often expressed using natural language or simple diagrams;
2. *semi-formal* representations that form structured descriptions, such as entity-relationship modelling or data flow diagrams; and
3. *formal* representations that form structured descriptions and include a set of operators for processing these descriptions, such as the relational model or logic-based models.

Unlike informal representations, which are often ill-defined, inconsistent and ambiguous, semi-formal and formal representations are well-defined, consistent and generally unambiguous. A key feature of formal representations that is lacking in semi-formal representations is the inclusion of operators that allow assertions to be made about the viewpoints being described; Greenspan et al. [19] describe this as the ability to 'reason' about representations. User viewpoints are typically defined using informal representations, whereas developer viewpoints are typically defined using semi-formal or formal representations.

Finkelstein et al. [17] first mooted the idea of using multiple representations to describe a viewpoint in 1989, but there has been surprisingly little work in this area since then. Darke and Shanks [12] found in a review of twelve different viewpoint development approaches that only two supported multiple representations to describe a single viewpoint: the Soft Systems methodology [7] and Scenario Analysis [25], both of which are user viewpoint approaches rather than developer viewpoint approaches.

The author's own research [38, 39] has followed the approach of using multiple representations to describe a single *developer* viewpoint, and uses an integrated terminology framework derived from viewpoint-oriented methods. This framework will now be described.

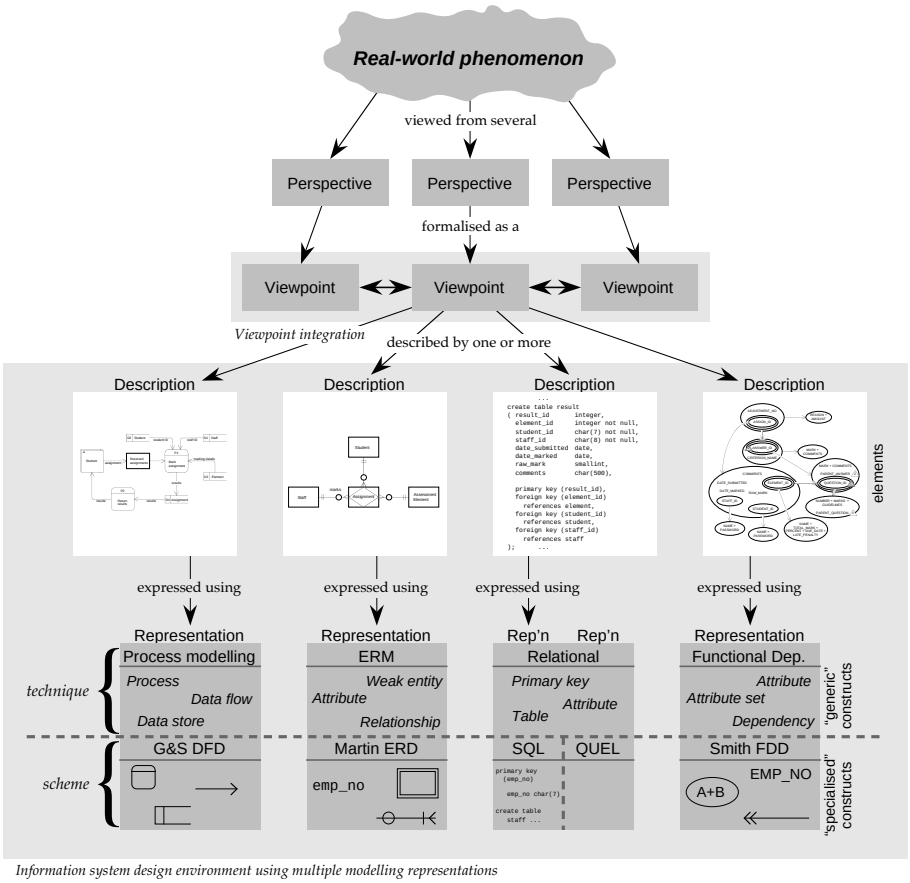
4 The viewpoint framework for discussing the use of multiple modelling representations

Looking at Table 1, there is an obvious dichotomy between the three viewpoint terminologies on the left and the three multiple-view terminologies on the right. The viewpoint terminologies deal primarily with 'high level' concepts and ignore how representations are internally structured; conversely, the multiple-view terminologies deal primarily with constructs within representations and ignore higher-level structure. The two sets of terminology are clearly related, yet the only real overlap between them is at the representation level.

There are also many synonyms in the terminologies presented in Table 1, for example, the terms 'style', 'representation', 'model' and 'data model' are all used to refer to similar concepts. Conversely, the term 'scheme' is used to refer to two completely different concepts. Such variation can lead to confusion, so there is a definite need to develop a consistent and integrated terminology framework.

The initial framework shown in Fig. 1 provides a good basis for extension, but the author has identified some confusion over the exact def-

initions of the terms ‘representation’, ‘technique’ and ‘scheme’. The author has addressed this confusion by clearly defining these terms, and has extended the original framework with the concepts of *descriptions*, *constructs* of representations and *elements* of descriptions (see Fig. 2).



Information system design environment using multiple modelling representations

Fig. 2. The extended terminology framework

4.1 Representations

Informally, a data modelling representation can be thought of as comprising two main parts:

1. a *generic* part that specifies the generic constructs that may be used to describe a viewpoint, such as entities, relations, and so on; which then determines
2. a *specialised* part that specifies the constructs peculiar to the representation, along with their visual appearance or notation, such as boxes for entities, SQL `CREATE TABLE` statements for relations, and so on.

Finkelstein et al.'s [17] use of the term 'style' does not clearly distinguish between these two parts; conversely, the 'techniques' and 'schemes' of Darke and Shanks [10] match these two parts well. It is therefore proposed to use the term *technique* to refer to the generic part of a representation, and the term *scheme* to refer to the specialised part of a representation. In practical terms, a technique can be thought of as a modelling 'approach', such as the entity-relationship approach or the relational model, and a scheme can be thought of as a particular 'notation' within that approach, such as a particular entity-relationship notation or relational calculus. Another way to think of this is that a scheme is an 'instantiation' of a particular technique.

A *representation* can thus be defined as the combination of a particular technique with a particular scheme, as illustrated in Fig. 2. In general, a technique may have one or more associated schemes, but each combination of a technique and a scheme forms a distinct representation. For example, the relational model is a technique, with *SQL* and *QUEL* as two possible schemes, but the combinations $(\mathit{Relational}, \mathit{SQL})$[1] and $(\mathit{Relational}, \mathit{QUEL})$ form two distinct representations, as shown in Fig. 3. Similarly, the entity-relationship approach (*E-R*) is a technique, with $\mathit{ERD}_{\mathit{Martin}}$ and $\mathit{ERD}_{\mathit{Chen}}$ as two possible schemes. The combinations $(\textit{E-R}, \mathit{ERD}_{\mathit{Martin}})$ and $(\textit{E-R}, \mathit{ERD}_{\mathit{Chen}})$ again form two distinct representations.

It is expected that a technique will not attempt to specify all possible concepts for all possible schemes within that technique. Rather, a technique defines the 'base' model, which is then specialised and extended by schemes to form a representation. This implies that a scheme may provide new constructs to a representation that have no direct analogue in the technique. For example, the relational technique [9] does not include general constraints, but they are an important feature of the relational scheme SQL/92 [14]. Similarly, type hierarchies are not part of the base E-R technique [8], but they do appear in some E-R schemes.

[1] In practice, the many dialects of SQL will form many different representations. This has been ignored here in the interests of clarity.

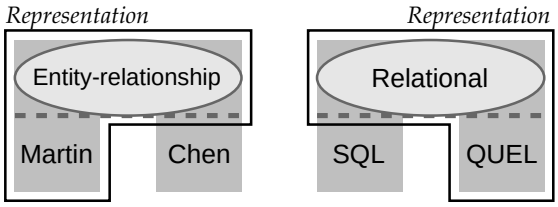


Fig. 3. Multiple schemes within a technique

4.2 Descriptions

Representations are an abstract concept, so they must be instantiated in some way in order to describe the content of a viewpoint. One way to view the instantiation of a representation is as a set of 'statements' that describe a viewpoint or some subset thereof. Finkelstein et al. [17] refer to this as a 'specification' or 'description'; Easterbrook [15] also refers to this concept as a description. The author has adopted the term 'description' as it emphasises the idea that they are used to *describe* a viewpoint.

A viewpoint is thus specified by a set of *descriptions*, each expressed using some representation, as shown in Fig. 2. Each description may describe either the whole viewpoint or some subset of the viewpoint; this is analogous to the concept of a 'view' as used in multi-view editing systems [5, 20, 27, 30]. For example, a developer viewpoint might be specified by union of the following four descriptions:

1. an object class description expressed using Unified Modelling Language (UML) class diagram notation [31];
2. a functional dependency description expressed using Smith functional dependency diagram notation [36, 37];
3. a relational description expressed using SQL/92; and
4. a data flow description expressed using Gane & Sarson data flow diagram notation [18].

Similarly, a user viewpoint might be specified by the union of a natural language description and a collection of diagrammatic descriptions. Descriptions may be distinct from each other, or they may overlap in a manner similar to viewpoints. Such redundancy can be useful in exposing conflicts both between descriptions and between viewpoints [15].

4.3 Constructs and elements

Every representation comprises a collection of *constructs*. These may be divided into generic constructs associated with the technique (*technique-*

level constructs) and specialised constructs associated with the scheme (*scheme-level constructs*), as shown in Fig. 2. The nature of a construct is defined by its *properties*, which include both its *relationships* with other constructs, and its *attributes*, such as name, domain or cardinality[2]. For instance, as illustrated in Fig. 4, a data store in a data flow diagram might have the attributes *name* (the name of the data store), *label* and *fields* (a list of data fields in the data store). The *flows* relationship specifies an association between the data store construct and a list of data flow constructs.

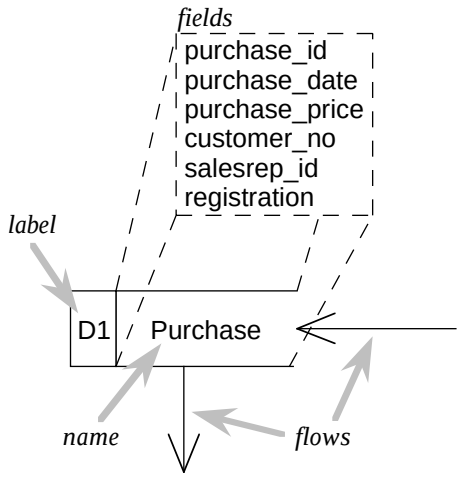


Fig. 4. Properties of a construct

In the same way that a description is an instantiation of a representation, an *element* is an instantiation of a construct; elements are combined to build descriptions. Examples of constructs include object classes, processes and attributes; elements corresponding to these constructs could be Order, Generate invoice and address.

5 A notation to express representations, descriptions, constructs and elements

It can be cumbersome to discuss aspects of representations and descriptions using natural language, for example, 'the Staff regular entity element

[2] The terms 'property', 'attribute' and 'relationship' come from the Object Data Management Group's object model [6].

of the description D_1 (expressed using Martin entity-relationship notation) of the managers' viewpoint'. The author has therefore developed a concise abstract notation for expressing representations, descriptions, constructs of representations and elements of descriptions. This notation is modelled in part on the data transfer notation of Pascoe and Penny [34]. Using the notation, the statement above could be expressed as:

$$D_1(V_{mgrs}, E\text{-}R, ERD_{Martin})\,[\mathit{staff} : \textsc{RegularEntity}]\,.$$

The notation is summarised in Table 2. The author has defined additional notations for expressing translations of descriptions from one representation to another [38]. For example, the expression:

$$D_1(V, FuncDep, FDD_{Smith}) \rightarrow D_2(V, E\text{-}R, ERD_{Martin})$$

denotes the translation within viewpoint V of a functional dependency description D_1 into an entity-relationship description D_2. Such additions are beyond the scope of this paper, however; only the base notation is discussed here.

5.1 Description and representation notation

The notation $D(V, T, S)$ denotes that description D of viewpoint V is expressed using constructs of technique T and scheme S (this may be abbreviated to D when V, T and S are clear). Thus, $D_1(V_p, E\text{-}R, ERD_{Martin})$ denotes a description D_1 of the viewpoint V_p that is expressed using constructs of the entity-relationship technique (E-R) and the Martin ERD scheme (ERD_{Martin}) [29].

The notation $\mathfrak{R}(T, S)$ denotes a representation $\mathfrak{R}$ that comprises a collection of constructs defined by the combination of technique T and scheme S (this may be abbreviated to $\mathfrak{R}$ when T and S are clear). Thus, $\mathfrak{R}_e(E\text{-}R, ERD_{Martin})$ denotes the representation $\mathfrak{R}_e$ formed by combining the constructs of the entity-relationship technique (E-R) with the Martin ERD scheme (ERD_{Martin}). This notation is similar to that used by Finkelstein et al. to describe viewpoint styles [17], but focuses on the technique and scheme used rather than individual constructs within a representation.

The combination of technique T and scheme S forms the representation $\mathfrak{R}(T, S)$, so it is also possible to denote the description $D(V, T, S)$ by $D(V, \mathfrak{R}(T, S))$, or simply $D(V, \mathfrak{R})$. Thus, the notations $D_1(V_p, E\text{-}R, ERD_{Martin})$, $D_1(V_p, \mathfrak{R}_e(E\text{-}R, ERD_{Martin}))$ and $D_1(V_p, \mathfrak{R}_e)$ are equivalent.

Table 2. Summary of the abstract notation

Notation	Associated term	Definition
V	Viewpoint	A formatted expression of a perspective on a real-world phenomenon.
T	Technique	A collection of generic constructs that form a modelling 'method', for example, the relational model or object modelling.
S	Scheme	A collection of specialised constructs that form a modelling 'notation', for example, SQL/92 or UML class diagram notation.
$\mathfrak{R}(T,S)$ or $\mathfrak{R}$	Representation	Representation $\mathfrak{R}$ comprises constructs defined by the combination of technique T and scheme S.
$D(V,T,S)$ or D	Description	Description D of viewpoint V is expressed using constructs of technique T & scheme S.
$\mathfrak{R}(T,S)$ [CON], $\mathfrak{R}$ [CON], or CON	Construct of a representation	CON specifies a construct of representation $\mathfrak{R}(T,S)$.
$D(V,T,S)$ [e : CON], D [e : CON], or D [e]	Element of a description	e specifies an element (instantiated from construct CON) of description $D(V,T,S)$.

The first form is preferred by the author as it clearly distinguishes between the technique and scheme.

Representations may differ in both the technique and scheme used, or they may share the same technique and differ only in the scheme. Thus, two descriptions D_1 and D_2 of viewpoint V that are expressed using representations having different schemes S_i and S_j are denoted by $D_1(V,T,S_i)$ and $D_2(V,T,S_j)$ respectively. Similarly, two descriptions D_3 and D_4 of viewpoint V that are expressed using representations having different techniques (T_k, T_l) and schemes (S_m, S_n) are denoted by $D_3(V,T_k,S_m)$ and $D_4(V,T_l,S_n)$ respectively.

Consider a viewpoint V_q that has three descriptions D_1, D_2 and D_3. D_1 is expressed using the entity-relationship technique and the Martin ERD scheme, and is denoted by $D_1(V_q, E\text{-}R, ERD_{Martin})$. D_2 is expressed using the functional dependency technique and the Smith functional dependency diagram (FDD) scheme, and is denoted by $D_2(V_q, FuncDep, FDD_{Smith})$. D_1 and D_2 differ in both the technique and the scheme used.

D_3 is expressed using the entity-relationship technique and the Chen scheme, and is denoted by $D_3(V_q, E\text{-}R, ERD_{Chen})$. D_3 differs from D_1 only in the scheme used.

If the viewpoint, technique or scheme are unspecified, they may be omitted from the notation. Thus, the notation $\mathfrak{R}_r(\mathit{Relational},)$ denotes any relational representation, and $D_1(, \mathit{Object}, CD_{UML})$ denotes a UML class diagram in an unspecified viewpoint.

5.2 Construct and element notation

Constructs are the fundamental components of a representation, whereas elements are the fundamental components of a description. Given a representation $\mathfrak{R}(T, S)$, a construct CON of $\mathfrak{R}$ is denoted by $\mathfrak{R}(T, S)\,[\text{CON}]$, or, if T and S are clear, simply $\mathfrak{R}\,[\text{CON}]$. Often $\mathfrak{R}$ may also be clear from the context, allowing the $\mathfrak{R}\,[]$ notation to also be omitted, leaving just CON. The name of the construct itself is denoted by SMALL CAPS.

The construct CON can be thought of as analogous to the concept of a relational domain in that it specifies a pool of possible 'values' from which an element e may be drawn. The notation $e : \text{CON}$ is used here to denote that e is a member of the set of all possible elements corresponding to the construct CON. This use of the ':' notation is similar to both domain calculus [13] and Z [4], where it is interpreted as meaning 'e is a member of the set CON'.

Now consider a description $D(V, T, S)$ (alternatively, $D(V, \mathfrak{R}(T, S))$). An element e of D (instantiated from construct $\mathfrak{R}\,[\text{CON}]$) is denoted by $D(V, T, S)\,[e : \text{CON}]$, or, if V, T and S are clear, simply $D\,[e : \text{CON}]$. The construct may also be omitted if it is clear from the context, that is, $D\,[e]$. The representation $\mathfrak{R}$ is omitted from the construct CON because $\mathfrak{R}$ is implied by T and S in the description and would therefore be redundant.

Some examples of construct and element expressions are given in Table 3. Both types of expression may specify a list, as illustrated by the last two examples.

6 Conclusion

In this paper has been described a framework for discussing the use of multiple modelling representations to describe a viewpoint. Earlier work on the use of multiple representations has produced a diverse and potentially confusing set of terminologies, none of which provides a complete set of terms covering all concepts in the area. Viewpoint concepts provide

Table 3. Examples of construct and element expressions

$\mathfrak{R}_e(E\text{-}R, ERD_{Martin})$ [EntityType] denotes the generic entity construct of the E-R/Martin representation $\mathfrak{R}_e$
$D_1(V, FuncDep, FDD_{Smith})$ [s : SingleValued] denotes a single-valued dependency element in the Smith notation functional dependency description D_1
$D_2(V, Relational, SQL/92)$ [$c_1, \ldots, c_n$: Column] denotes a collection of column elements in the SQL/92 description D_2
$\mathfrak{R}_d(DataFlow, DFD_{G\&S})$[DataStore, DataFlow] denotes the data store and data flow constructs of the data flow modelling/Gane & Sarson representation $\mathfrak{R}_d$

a useful framework within which to discuss the use of multiple representations, but there is a lack of clarity over the definitions of the terms 'representation', 'technique' and 'scheme'.

To remedy these issues, the author has clarified the definitions of 'representation', 'technique' and 'scheme', and extended the viewpoint framework with the concepts of *description*, *construct* of a representation and *element* of a description. Also described was an abstract notation for writing representation, description, construct and element expressions.

The framework described in this paper provides a consistent, integrated terminology and notation for researchers working on the use of multiple representations to describe a viewpoint.

References

[1] Paolo Atzeni and Riccardo Torlone. Schema translation between heterogeneous data models in a lattice framework. In Robert Meersman and Leo Mark, editors, *Database Applications Semantics, Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)*, pages 345–361, Stone Mountain, Atlanta, Georgia, USA, May 30–June 2 1995. IFIP, Chapman & Hall, London.

[2] Paolo Atzeni and Riccardo Torlone. Management of multiple models in an extensible database design tool. In P. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Fifth International Conference on Extending Database Technology (EDBT'96)*, volume 1057 of *Lecture Notes in Computer Science*, pages 79–95, Avignon, France, March 25–29 1996. Springer-Verlag.

[3] Paolo Atzeni and Riccardo Torlone. MDM: A multiple-data-model tool for the management of heterogeneous database schemes. In Joan M. Peckman, editor, *SIGMOD 1997 International Conference on the Management of Data*, pages 528–531, Tucson, Arizona, May 13–15 1997. ACM, ACM Press.

[4] S. M. Brien and J. E. Nicholls. Z base standard. Technical Monograph PRG-107, Oxford University Computing Laboratory, Oxford, UK, nov 1992.
[5] Marc H. Brown. Zeus: A system for algorithm animation and multi-view editing. Research Report 75, Digital Equipment Corporation, Systems Research Center, Palo Alto, California, 28 February 1992.
[6] R.G.G. Cattell, Douglas K. Barry, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez. *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann, San Francisco, California, 2000.
[7] P.B. Checkland. *Systems Thinking, Systems Practice*. John Wiley & Sons, Chichester, England, 1981.
[8] Peter Pin-Shan Chen. The entity-relationship model — Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1), 1976.
[9] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 1970.
[10] Peta Darke and Graeme Shanks. Viewpoint developments for requirements definition: An analysis of concepts, issues and approaches. Working Paper 21/94, Department of Information Systems, Monash University, Melbourne, Australia, December 1994.
[11] Peta Darke and Graeme Shanks. Viewpoint development for requirements definition: Towards a conceptual framework. In *Sixth Australasian Conference on Information Systems (ACIS'95)*, pages 277–288, Perth, Australia, September 26–29 1995.
[12] Peta Darke and Graeme Shanks. Stakeholder viewpoints in requirements definition: A framework for understanding viewpoint development approaches. *Requirements Engineering*, 1:88–105, 1996.
[13] C.J. Date. *An Introduction to Database Systems*. Addison-Wesley, Reading, Massachusetts, seventh edition, 2000.
[14] C.J. Date and Hugh Darwen. *A Guide to the SQL Standard*. Addison-Wesley, Reading, Massachusetts, fourth edition, 1997.
[15] Steve M. Easterbrook. *Elicitation of Requirements from Multiple Perspectives*. PhD thesis, Imperial College of Science Technology and Medicine, University of London, London, 1991.
[16] Steve M. Easterbrook and Bashar A. Nuseibeh. Using ViewPoints for inconsistency management. *Software Engineering Journal*, 11(1):31–43, 1996.
[17] A.C.W. Finkelstein, M. Goedicke, J. Kramer, and C. Niskier. ViewPoint oriented software development: Methods and viewpoints in requirements engineering. In J.A. Bergstra and L.M.G. Feijs, editors, *Second Meteor Workshop on Methods for Formal Specification*, volume 490 of *Lecture Notes in Computer Science*, pages 29–54, Mierlo, The Netherlands, September 1989. Springer-Verlag.
[18] C. Gane and T. Sarson. *Structured Systems Analysis: Tools and Techniques*. Prentice-Hall Software Series. Prentice-Hall, Englewood Cliffs, New Jersey, 1979.
[19] S. Greenspan, J. Mylopoulos, and A. Borgida. On formal requirements modeling languages: RML revisited. In Bruno Fadini, editor, *Sixteenth International Conference on Software Engineering*, pages 135–148, Sorrento, Italy, May 1994. IEEE Computer Society Press.
[20] John C. Grundy. *Multiple Textual and Graphical Views for Interactive Software Development Environments*. PhD thesis, Department of Computer Science, University of Auckland, Auckland, New Zealand, June 1993.
[21] John C. Grundy and John G. Hosking. Constructing integrated software development environments with MViews. *International Journal of Applied Software Technology*, 2(3/4):133–160, 1997.

[22] John C. Grundy, John G. Hosking, and Warwick B. Mugridge. Supporting flexible consistency management via discrete change description propagation. *Software — Practice and Experience*, 26(9):1053–1083, September 1996.

[23] John C. Grundy and John R. Venable. Providing integrated support for multiple development notations. In *Seventh Conference on Advanced Information Systems Engineering (CAiSE'95)*, volume 932 of *Lecture Notes in Computer Science*, pages 255–268, Finland, June 1995. Springer-Verlag.

[24] John G. Hosking, Warwick Mugridge, Robert Amor, and John Grundy. Keeping things consistent. *New Zealand Journal of Computing*, 6(1):353–362, August 1995.

[25] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, and C. Chen. Formal approach to scenario analysis. *IEEE Software*, 11(2):33–41, March 1994.

[26] Richard Hull and Roger King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[27] D.A. Jacobs and C.D. Marlin. Software process representation to support multiple views. *International Journal of Software Engineering and Knowledge Engineering*, 5(4), December 1995.

[28] Gerald Kotonya and Ian Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996.

[29] James Martin. *Information Engineering, Book II: Planning and Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, revised edition, 1990.

[30] S. Meyers. Difficulties in integrating multiview environments. *IEEE Software*, 8(1):49–57, January 1991.

[31] Pierre-Alain Muller. *Instant UML*. Wrox Press, Birmingham, 1997.

[32] G. Mullery. CORE — A method for controlled requirements specification. In *Fourth International Conference on Software Engineering*, pages 126–135, Munich, Germany, September 17–19 1979. IEEE Computer Society Press.

[33] B. Nuseibeh, J. Kramer, and A.C.W. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994.

[34] Richard T. Pascoe and John P. Penny. Constructing interfaces between (and within) geographical information systems. *International Journal of Geographical Information Systems*, 9(3):275–291, 1995.

[35] Steven P. Reiss. Connecting tools using message passing in the Field environment. *IEEE Software*, 7(7):57–66, July 1990.

[36] Henry C. Smith. Database design: Composing fully normalized tables from a rigorous dependency diagram. *Communications of the ACM*, 28(8):826–838, 1985.

[37] Nigel Stanger. Modifications to Smith's method for deriving normalised relations from a functional dependency diagram. Discussion paper 99/23, Department of Information Science, University of Otago, Dunedin, New Zealand, December 1999.

[38] Nigel Stanger. *Using Multiple Representations Within a Viewpoint*. PhD thesis, Department of Information Science, University of Otago, Dunedin, New Zealand, December 1999.

[39] Nigel Stanger and Richard Pascoe. Environments for viewpoint representations. In Robert Galliers, Sven Carlsson, Claudia Loebbecke, Ciaran Murphy, Hans Hansen, and Ramon O'Callaghan, editors, *Fifth European Conference on Information Systems (ECIS'97)*, volume I, pages 367–382, Cork, Ireland, June 19–21 1997. Cork Publishing.

[40] S.Y.W. Su and S.C. Fang. A neutral semantic representation for data model and schema translation. Technical report TR-93-023, University of Florida, Gainesville, Florida, July 1993.
[41] S.Y.W. Su, S.C. Fang, and H. Lam. An object-oriented rule-based approach to data model and schema translation. Technical report TR-92-015, University of Florida, Gainesville, Florida, 1992.
[42] D. Tsichritzis and F. Lochovsky. *Data Models.* Prentice-Hall, 1982.