Conceptual Modelling for Web Information Systems: What Semantics Can Be Shared?

Simon McGinnes

School of Computer Science and Statistics, Trinity College Dublin, Ireland
Simon.McGinnes@tcd.ie

Abstract. There is an increasing need to allow software applications to exchange data, which usually requires the negotiation of meanings between incompatible conceptual models. Theoretically, the concepts in one application can be mapped to those in another, but this can be challenging in practice. The problem is more fundamental than “information plumbing”; it requires reconciliation between alternative and possibly conflicting ways of viewing the world. Ontologies and the Semantic Web have been proposed as potential solutions to the information exchange problem. This research investigates from first principles what kinds of data exchange are possible, with the aim of analyzing the issue in a useful way for the developers and users of web-based information systems. The analysis suggests particular means of facilitating data exchange involving the use of a simple set of shared basic-level categories.

Keywords: web information systems, ontologies, data models, conceptual modelling, Semantic Web.

1 Introduction

Now is an important moment in human development. Thanks to the internet, for the first time in history we have the capacity to share data on a massive scale, to move information in digital form more or less anywhere we want at the press of a button. But with this new ability comes the need to think in new ways about information exchange. Our conventional view of sharing information developed in the low bandwidth world of conversation, storytelling, books and newspapers. Trying to apply the same ideas to the internet may risk creating the potential for confusion. Recently, efforts have been made to automate the sharing of information using technologies such as the Semantic Web, microformats and web services. As we begin to use these technologies it is important to be clear about what we mean when we talk about sharing information and what kinds of information can feasibly be shared.

This position paper aims to clarify these questions in a useful way for the developers and users of web-based information systems. The discussion is necessarily rather philosophical, but it is made as practical as possible because sharing data is inherently a practical issue. In fact, the sole reason for sharing data is to facilitate action. Viewing shared data as passive information is to miss the point; the significance of sharing data is in the potential it creates for action.

1.1 The Need for Data Sharing

There is a greater need for data sharing within and particularly between organizations than ever before. Historically, most organizations have used a portfolio of information systems for different purposes, so corporate data has been locked into a number of separate, mutually-incompatible data structures. Organizations need their systems to work together, but sharing data between heterogeneous applications is rarely straightforward. In an attempt to achieve integration, many organizations use enterprise software applications, which address requirements across a range of business processes. However, the adoption of enterprise software products tends to lock organizations into a single supplier and prevents the selection of best-of-breed solutions. Both approaches—the "application portfolio" approach, and the adoption of enterprise software—can be expensive, and neither is ideal.

The situation is exacerbated when it comes to sharing data between organizations, or between organizations and individuals. There is at present no universal inter-organizational equivalent to the enterprise software solution. Therefore organizations have little choice but to make their mutually-incompatible applications work together. Historically, two factors have presented a barrier to interoperation. One is physical incompatibility: if two systems are physically disconnected then they cannot communicate. The Internet and related technologies have largely solved that problem. However, as soon as applications can physically exchange data then the need for semantic compatibility becomes paramount. This problem is less easily solved.

For example, consider two information systems which need to interoperate: (a) a sales order processing system containing data about product types, suppliers, customers, orders and employees, and (b) an accounting system with data on transactions, debtors, creditors and accounts. Although the two sets of concepts describe the same real-world phenomena (people, organizations and business transactions), the systems conceptualize the data very differently. The systems are conceptually incompatible, even though they store data about the same things. The incompatibility presents a barrier to data exchange between the applications. Unless a programmer crafts a suitable custom interface between the two applications, which translates between the two ways of conceptualizing the underlying business entities, it will be difficult to share data between them; the applications are built around concepts that do not map to one another in any straightforward way.

Of course, the problem is not confined to this trivial example of sales and accounting systems. It is rife, because most application software is structured around idiosyncratic, domain-specific concepts. This seemingly-benign design practice, which has long been the norm in information systems engineering, guarantees that different programs will tend to be semantically incompatible. But, regardless of the pros and cons of using ad hoc concepts in application design, we need to find ways of integrating conceptually-incompatible systems. This is where technologies such as ontologies and the Semantic Web offer some hope.

1.2 Structure and Terminology

The paper is structured as follows. Section 2 explores from first principles the ways in which data exchange between semantically-incompatible systems can and cannot be achieved. Section 3 discusses ontologies, the Semantic Web and related technologies in the light of this analysis. Section 4 outlines some possible changes to design practice suggested by this analysis, which might facilitate semantic interoperability. Section 5 concludes with a summary of findings and limitations, and suggests directions for further work.

Two commonly-used terms are avoided in this paper, because their ambiguity could contribute to confusion. The first is *information*; this is a term from everyday language which has been co-opted in IS/IT with a variety of meanings. It can refer to essentially the same thing as data, or to any digital signal, or to text, or to facts that have particular significance. The second term is *semantics;* this term has been adopted by particular academic communities, with the result that its meaning is rather blurred. Because the meanings of both terms are central to the arguments in this paper, we avoid using them altogether and will use other, less ambiguous terms as appropriate.

2 Shared Meanings

When talking about data and information systems it is important to distinguish clearly between real-world things ("non-information resources" in linked data terminology), the signs that represent them, and mental states which corresponding to signs and real-world things. For example, the term "IBM" is a sign, which corresponds to a particular organization known as IBM (a real-world thing). An observer may have an idea of the organization known as IBM, and this is a mental state. This three-way relationship is sometimes encapsulated in the "semiotic triangle" [1].

Information systems store signs (bits, bytes, images, text, etc.) which represent real-world things and mental states. When people communicate, they do so using signs such as words and gestures. In all of these cases, signs are manipulated in the hope of evoking mental states. This much is uncontroversial; in the following discussion, we consider how mental states can be evoked through the use of signs by people and machines. We are interested in knowing how software applications can exchange data despite conceptual incompatibility. In order to understand that, we need to understand how meaning is transmitted. Since humans often try to transmit meaning to one another, it is helpful first to consider how this works.

2.1 How Do People Share Meanings?

Figure 1 illustrates two views of the communication of meaning between individuals. In view (a) meanings are shared between individuals, and the purpose of language is merely to evoke these shared meanings. This view is reminiscent of Jung's collective unconscious or the concept of the platonic ideal. In view (b) meaning is transmitted directly by language and so there is no particular need for meanings to be shared.

We argue that, at best, both of these views are unhelpful. There is no evidence for view (a); despite centuries of philosophical theorizing, no scientific research has uncovered any mechanism by which meanings might exist "out there" and be shared between individuals. As for view (b), meaning is not something which can flow; it is an experience. When we talk of conveying meaning, this is a figure of speech and not

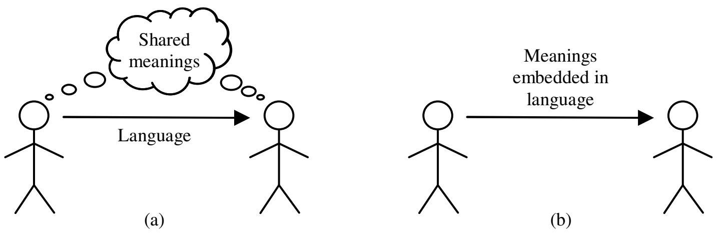

Fig. 1. Models for the communication of meaning between people

to be taken literally. Language cannot literally carry meaning since it consists only of signs: sounds and gestures. The medium in this case is *not* the message. Meaning arises only as an experience in each observer when he or she hears the sounds and observes the gestures. The mental states that are evoked by these perceptions give rise to mental states which we experience as meaning (Figure 2). According to this view, no meanings can be shared; experience and memory are private mental states of each individual. Language, which consists of signs, flows between individuals and evokes the experience of meaning separately in each individual.

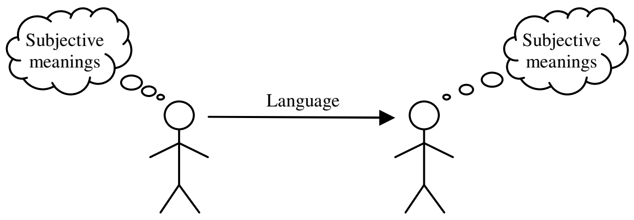

Fig. 2. Improved model for the communication of meaning between people

However, this raises the question of how humans communicate at all, if there are no shared meanings. The answer is that all meaning is subjective, yet we can assume that much of our experience is similar. Humans communicate imperfectly and misunderstanding is common. But we can proceed for much of the time as if we share common meanings, because we are physiologically similar to one another and have similar formative experiences. To give a trivial example, it is a reasonable working assumption that we all mean the same thing by “red”. This assumption is in fact incorrect, because evidence tells us that many people are colorblind and cannot experience red in the same way as non-colorblind individuals. It is also likely that individual variations in the perception of colors exist quite apart from colorblindness. Nevertheless, it is more helpful to assume common meanings than to assume we have no shared experience. The same argument can be extended to many concepts; while there may be disagreement between individuals on the interpretation of particular signals, it is more beneficial to attempt communication than not to.

Unlike computers, people can rely on having a common set of concepts, because we all share the experience of living in the same world (roughly), are members of the same species, and share elements of brain function. Hence our most fundamental concepts (whether innate or learned) tend to be similar; most people would agree on what a person is, for example. These ideas provide the context for thought and tell us generally how to behave in relation to any given thing or situation that we encounter.

2.2 How Is Meaning Experienced?

A related question is how each individual's subjective meanings are stored and what form they take when consciously experienced. This is an area of academic debate, but one broad trend in our understanding of meaning can be identified. Early theories of cognition postulated that meaning stems from conceptual structures in the mind. In psychology, the spreading activation model is an example of such a theory; nodes representing distinct concepts are connected explicitly, reflecting associative links between ideas [2]. Similar thinking in computer science gave rise to conceptual graphs and the ideas of schemata and frames [3]. There have even been suggestions that the unconscious mind performs computations using symbolic logic [4].

However, despite a great deal of looking, neuroscience has not found evidence for conceptual structures in the mind. It is becoming apparent that meaning arises in a rather different way. Rather than activating concepts, perceptions appear instead to elicit the recall of prior experience in a holistic manner. When we observe a situation or hear language we recall a complex of memories with associated emotional states. To recall is to re-experience, and it is this conscious re-experiencing of prior experiences which we know as meaning.

Recall occurs on the basis of similarity using perceptual feature-matching processes, which operate on multiple levels. This allows us to recall memories because of literal as well as abstract (analogical) similarities—one situation may be quite unlike another in detail and yet we still are reminded, because of more abstract similarities [5]. This suggests that, as a general principle, thinking about a concept does not depend on definitions or on analytical thinking; it involves the retrieval of prior experience on the basis of perceptual similarity. The definition of a concept emerges and becomes crisp only when we try to define it consciously.

Although neuroscience has not found evidence for the existence of conceptual structures, the results of studies (some of which use brain imaging) suggest that the primate brain possesses "hardwired" semantic regions which process information about particular subjects such as people, animals, tools, places and activities [6]. There is debate about the interpretation of these results, but the implication is that the brain automatically segregates (on the basis of feature matching) cognitive processing into certain broad categories. Categorization occurs unconsciously and the categories appear to be innate, not learned. They correspond to concrete, everyday ideas rather than abstractions. We can hypothesize that other concepts—more specialized ideas like *customer* and *account*, for example—are learned, and become associated with the corresponding basic-level innate categories.

2.3 How Do Computers Share Meaning?

To reiterate, we are interested in knowing how software applications can exchange data despite conceptual incompatibility. Software applications cannot be said to experience, at least in the sense meant above, and so they cannot be considered capable of experiencing meaning in the way that a human does. The closest equivalent to meaning for a software application is when it has been programmed to act on specific types of data; we may then say (figuratively) that the application "understands" that type of data. This figurative understanding consists of two elements: (a) being programmed to accept data with a particular structure, and (b) being programmed to deal with that type of data appropriately. However, we must be clear that the ability to handle each specific type of data requires explicit programming.

So, if two applications "understand" different types of data (and so are conceptually incompatible) how can they exchange their own equivalent of meaning? As before, we consider different models of communication. In Figure 3, view (a) suggests that it is sufficient merely to exchange data. View (b) suggests that metadata should also be included with the data; its function is to explain the structure and purpose of the data, so that the receiving application can process it properly.

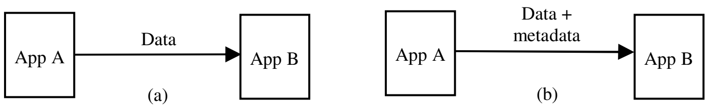

Fig. 3. Models for the communication of meaning between applications

View (a) is clearly insufficient since (to use our earlier example) the accounting system will not recognize data about customers, orders and so on, and will therefore be unable to do anything useful with them. Therefore some explanation of these concepts is needed. However, view (b) is also incomplete, because any metadata must be expressed in terms of concepts that the receiving application understands. The accounting system will understand what to do with data about customers and orders only if the metadata explains the concepts *customer* and *order* in terms of the concepts *account* and *transaction*, or in terms of other constructs which the accounting system understands. That would require the sending application to have knowledge (e.g. of accounts and transactions) which it does not have.

For two conceptually-incompatible applications to exchange meanings, they must both "understand" how to process particular types of data (Figure 4). This requires that both be programmed with the ability to handle those types of data. For that to be the case, both applications must share particular conceptual structures and must contain program code which can deal with the corresponding data. But that would mean that *the applications would no longer be conceptually incompatible*. Of course, this is a contradiction. It means that the only way for conceptually-incompatible applications to exchange data is by becoming conceptually compatible.

This analysis tells us that conceptually-incompatible applications can never exchange data unless they become conceptually compatible first. It makes no difference how complicated the data exchange method is or how much markup is included. When data is exchanged it can be processed meaningfully only if the sending and receiving applications are conceptually compatible and this situation can be achieved only in advance of the exchange, through programming. In the next section, we consider the implications of this finding for data exchange technologies.

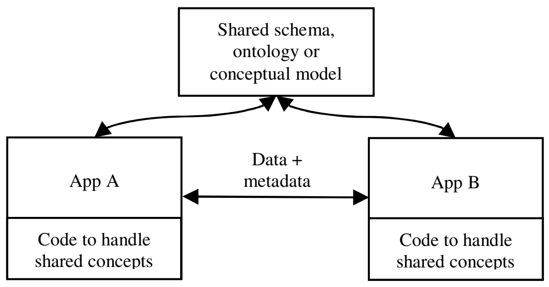
Shared schema, ontology or conceptual model
App A
Code to handle shared concepts
Data + metadata
App B
Code to handle shared concepts

Fig. 4. Improved model for communication of meaning between applications

3 Implications for Data Exchange Technologies

3.1 Ontologies and the Semantic Web

Ontologies have been proposed as a solution to data transfer problems. An ontology is a formal description of concepts and their relationships. Unlike conceptual models, ontologies are generally prepared independently of particular applications. Upper ontologies such as SUMO contain hundreds of thousands of interconnected concepts; they aim to encapsulate broad areas of knowledge in model form. They contain both highly-generic and highly-specific abstractions; for example, SUMO contains the hierarchy *sentient agent* → *cognitive agent* → *human* → *internet user* (the → symbol can be read as "may be a"). Domain ontologies address more limited subject areas. Some researchers are trying to reconcile upper ontologies with domain ontologies; others are developing intermediate, industry-specific reference ontologies. Rather than converging on a single, universal ontology, the field is producing many distinct ontologies, each with particular strengths and scope [7].

The Semantic Web is partially based on ontologies. It envisions an alternative or addition to the conventional web which will carry data using ontology-like structures. According to this vision, programs will be able to interoperate and share data freely, using ontologies to mediate data exchange. The goals for the Semantic Web are ambitious: it will "enable machines to comprehend semantic documents and data" [8]. While a full theoretical treatment of the Semantic Web has yet to be elucidated, the idea is that, if conceptual structures can be defined in sufficient detail and expressed in machine-readable metadata, then software applications will be able to share and use data intelligently and seamlessly.

The implicit assumption is that more detailed and rigorous definition of conceptual structures can create machine comprehension; the computer will "understand" what the data means sufficiently well to be able to process it appropriately. But how machine comprehension can arise from more detailed definition of concepts has yet to be explained. This appears to be a rather grand challenge similar to those previously addressed in artificial intelligence research.

Setting aside the claims about machine comprehension, can ontologies provide a basis for data sharing? In theory if two applications conform to the concepts defined in a particular ontology then exchange of data becomes easy, because the applications are conceptually compatible. However, it requires that applications be designed from the ground up to conform to the particular ontology, which is an expensive proposition. Also, no standard ontology has emerged. Multiple competing ontologies and microformats exist, each with its own peculiar take on the relevant domains, and it is not particularly easy to use them in combination. If applications are built to match a variety of different ontologies, the risk is that this lack of standardization will perpetuate the present problem of ad hoc conceptual structures and legacy "information islands".

An alternative approach is to use ontologies as a basis for mapping the conceptual models of applications to one another. Defining conceptual structures in more detail does not in itself convey significance or create machine comprehension. But richer definition of concepts (such as *customer* and *account*, in our example) might help link them to terms in a common ontology. This would allow programs to read the data, but it would not tell programs how to process the data appropriately—unless specific programming were to be done for each concept or term in the ontology. That is also an expensive proposition and probably less likely because ontologies contain many thousands of terms, most of which are irrelevant to any particular application.

To summarize, the development of ontologies may assist in reconciling mutually-incompatible conceptual structures and so allow applications to exchange data. But ontologies are in effect better "information plumbing"; they cannot tell applications how to process any given type of data. Each distinct ontology defines a set of concepts which must be programmed into a software application before the application can process the corresponding data meaningfully. The metadata (in RDF, for example) is useful because it signifies the type of data, but this is useful only inasmuch as the receiving application *already understands* how to deal with data of that type.

3.2 Potential for Basic-Level Categories to Facilitate Interoperability

Because of the fundamental limitation that applications must already share concepts in order to exchange corresponding data, the use of ontologies does not substantially improve our ability to link applications on a large scale or more rapidly. We are still held back by the need to program into each application the methods that it must use to handle data corresponding to each concept. There is no such thing as common sense or general knowledge for a computer application which would tell it how to handle specific types of data.

However, the discussion in Section 2.1 alluded to some results from neuroscience which may be helpful in thinking about this problem. The results suggest that a small number of categories are hardwired in the brain and perhaps innate. These correspond to concepts that people seem to understand intuitively: other people, food, tools, actions and so on. They are neither particularly generic nor particularly abstract but instead couched at a basic or everyday level [9]. One hypothesis is that these common basic-level concepts, together with other common experiences such as emotions and sensations, allow us to think and communicate about more complex ideas.

Applying the same idea to software applications, we could envisage a simple set of shared "innate" categories for software applications to adhere to, providing a basis for at least some limited data exchange and interoperability, without the need for programming. It is generally a straightforward matter to map an existing conceptual model to a simple set of basic-level categories [10]. For example, data about people could be tagged as such, as could data about places, documents, organizations, and so on. Once an item of data had been tagged according to this set of simple basic-level categories, certain default programming would be applicable. For example, places can be represented as maps; documents can be downloaded and opened. Other possible basic-level categories include systems, physical objects, conceptual objects and categories [11].

Recall that the purchase order processing and accounting systems in our example contained the concepts *supplier* and *creditor*. If both of these concepts were identified with the innate category *organization*, then the two applications could exchange data meaningfully, identifying data in the categories *supplier* and *creditor* as data about organizations. The applications could then treat the data in a way deemed appropriate for processing data about organizations. Similarly, the concepts *purchase* and *transaction* could be identified with the innate category *activity*, allowing exchange and treatment appropriate for activities, and so on. By providing a generic level of programming to suit each shared basic-level category, at least some basic level of default operation would be possible on exchanged data without the two applications possessing complex shared conceptual schemas.

4 Conclusion

Today, each software application is structured around its own set of unique, ad hoc, concepts. This design practice guarantees conceptual incompatibility. There is a need to find alternative design approaches that will allow applications to work with different types of data more flexibly and share data more readily. Yet it is important to respect the uniqueness of each application's conceptual model; a one-size-fits-all conceptual model cannot easily be imposed.

It will become increasingly important for applications to be able to share data automatically, despite the problem of conceptual incompatibility. A way is needed of imbuing applications with the equivalent of common sense or general knowledge, which will allow them to offer a sensible response to new types of data. In ontology research, it is hoped that this will be achieved through more rigorous and more detailed definition of concepts, to ultimately enable a form of machine comprehension. But it is unclear how machine comprehension will be produced by more detailed definition of concepts. As for the less ambitious goal of using ontologies to map between conceptual models, even if ontology use in information systems were to become widespread, a critical mass of organizations would need to adopt a particular ontology before the benefits of standardization could be realized.

Overall, there is a tendency to think in rather non-specific ways about how ontologies and the Semantic Web might permit free exchange of data. Any exchange of data is constrained by the problem of conceptual incompatibility, and this cannot be overcome solely by the inclusion of more complex markup. It requires advance programming so that applications are able to handle the types of data to be exchanged. This cardinal rule constitutes a fundamental limit on conceptual interoperability and can be stated thus: *applications can meaningfully interoperate with respect to data of a specific type only if they have been programmed in advance to handle data of that type*. When data containing particular concepts is exchanged, applications have to be specifically programmed to handle the relevant concepts, regardless of what mechanism is used to transfer the data or construct the programs, and irrespective of what markup or metadata is included.

In conclusion, this work remains theoretical. However, the relatively limited progress to date on the Semantic Web (in comparison with the worldwide web, for example) may represent evidence of the inherent limitation discussed in this paper. Research is needed into ways of genuinely allowing heterogeneous applications to exchange data in the face of conceptual incompatibility. Whether or not ontologies are involved, the idea of using a simple set of "innate" categories should be tested because it may offer a more practicable approach than attempting to implement large and unwieldy ontologies in software applications.

References

1. Liebenau, J., Backhouse, J.: Understanding Information: An Introduction. Macmillan, Basingstoke (1990)
2. Crestani, F.: Application of Spreading Activation Techniques in Information Retrieval. Artificial Intelligence Review 11, 453–482 (1997)
3. Sowa, J.F.: Conceptual Structures. Addison-Wesley, Reading (1984)
4. Modell, A.H.: Imagination and the Meaningful Brain. The MIT Press, Cambridge (2003)
5. Eysenck, M.W., Keane, M.: Cognitive Psychology: A Student's Handbook. Psychology Press, UK (2005)
6. Mason, M.F., Banfield, J.F., Macrae, C.N.: Thinking About Actions: The Neural Substrates of Person Knowledge. Cerebral Cortex 14, 209–214 (2004)
7. Kalfoglou, Y., Hu, B.: Issues with Evaluating and Using Publicly Available Ontologies (2006)
8. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284, 28–37 (2001)
9. Pansky, A., Koriat, A.: The Basic-Level Convergence Effect in Memory Distortions. Psychological Science 15, 52–59 (2004)
10. McGinnes, S.: Conceptual Modelling: A Psychological Perspective. Ph.D Thesis, Department of Information Systems, London School of Economics, University of London (2000)
11. McGinnes, S., Amos, J.: Accelerated Business Concept Modeling: Combining User Interface Design with Object Modeling. In: Harmelen, M.V., Wilson, S. (eds.) Object Modeling and User Interface Design: Designing Interactive Systems, pp. 3–36. Addison-Wesley, Boston (2001)