The University of Sydney

Lies, damned lies and stereotypes: Pragmatic approximations of users

Technical Report Number 482

June 1994

Judy Kay

ISBN 0 86758 919 1

Basser Department of Computer Science
University of Sydney NSW 2006

Lies, damned lies and stereotypes: pragmatic approximations of users

Judy Kay †
Basser Department of Computer Science
University of Sydney
judy@cs.su.oz.au

Abstract

Stereotypes are a pervasive element of much work in user modelling. This paper discusses ways that stereotypes have been used, both in user modelling research and, by other names, in many other areas.

As a basis for better understanding of stereotypes for user modelling, the paper develops bases for their uses in such diversified areas as information filtering, help systems, advisors and the tailors of information presentation.

It also deals with major issues for deploying stereotypes: technical issues like the representation and acquisition of stereotypes and matching individuals to them and also socio-political issues.

The paper describes projects that have developed toolkits for the various technical components of the tasks. These include extensive and diversified approaches to building the stereotypes, determining which to apply to individual users and exploiting them in individualising the user's interaction with the machine. Also, mindful of the lies that are inherent in the approximation that a stereotype must be, it discusses tools and approaches for attending to the socio-political concerns.

Introduction

The stereotype is one of the common elements in much user modelling work. It captures default information about groups of people. This simple but powerful idea was introduced by Rich (Rich 1979, 1983, 1989) who used people's descriptions of themselves to deduce the characteristics of books that they would probably enjoy.

Since Rich's introduction of the notion of a stereotype, it has become a basic element in many user modelling systems. In particular, many user modelling shells support entities that are called stereotypes. Since these are the systems that are designed to use in a range of user modelling applications, we would expect them to represent the major approaches in user modelling.

For example, GUMS, a Generalised User Modelling System, (Finin 1989) supported a sophisticated stereotype mechanism. This maintained facts and rules in its stereotypes. It further distinguished between *definite* and *default* parts of a stereotype. The former must apply to all users in the class. So they act as a definition for that class (eg programmer stereotype requires that the person programs). By contrast, the default facts act as initial beliefs.

Similarly, BGP-MS (Kobsa 1990) supports a sophisticated stereotype mechanism. The stereotypes can be constructed with the aid of a graphical tool. This helps the system builder see the relationships between the various stereotypes. The system also checks the consistency of the structures created. It has a rule language for managing stereotypes.

Also Brajnik's UMT (Brajnik, Guida and Tasso 1990, Brajnik and Tasso 1992) supports a knowledge base of stereotypes which are used as default inferences about the user. Each application has a general stereotype plus more specialised ones for different classes of users.

Stereotypes are a basic information source in the um toolkit (Kay 1990) where they are used for initial default information to model the user when nothing better is available.

Stereotypes are a critical part of Orwant's Doppelganger (Orwant 1993) where he extends it with the notion of *communities* which are groups of users with many commonalities. A user may be classified as belonging to several communities and where the user's model has no explicit information about some aspect, its value is calculated across the communities the user belongs to.

It is rather remarkable that these user modelling shells have very little common. They take differing views of the tasks of user modelling and employ different representational approaches. Stereotypes constitute a strong point of commonality.

Given the apparent importance of stereotypes, it is useful to refine our understanding of what they are, what they are not and their relationship to other elements of user modelling. This paper does this, first by characterising the intuitive appeal of stereotypes, then by tightening the definitions and analysing different classes of stereotypes. From these, it is possible to identify important issues, both technical and non-technical for the effective deployment of stereotypes.

† Currently at Dept of Computer Sciences, University of Wisconsin, Madison

The power of stereoypes in filtering

To make this discussion more concrete, it is cast in terms of one set of approaches to a range of applications, like the *filtering* tasks that are of growing importance. In these, stereotypes can provide a powerful basis for agents that help people find what they want from large bases of on-line information, entertainment and the shades of infotainment that lie between these. The following scenarios will be used in the remainder of the paper.

Scenario 1

> The user can select a movie from a database of thousands. They request the names of a dozen movies that are likely to appeal to them. Clicking on one, they can have a plot summary and other details and brief film clips. From this, the user selects a movie to see.

Stereotypes can aid here with reasoning of the form: *People who enjoyed movie X will enjoy movie Y*; *People who hated movie W won't enjoy movie Z*; or *I like the same sorts of movies as the user, Smith, does*.

Scenario 2

> The user wants to learn the programming language C. They already know Pascal and Prolog quite well and know a minimal amount about using the Unix operating system. They have access to a WWW (World Wide Web) hypertext document about C and want to learn enough to write a C program that implements some graph algorithms.

Although this user could simply explore the hypertext document, it should be more effective if the presentation is customised so that it builds upon their substantial current knowledge of relevant programming constructs. This customisation could use stereotypes of the form *a user who knows Pascal can learn {C concepts}* - a set of C concepts - *from a terse comparative description*.

> **Scenario 3**
> An Italian is in Minneapolis for a year and would like to keep in touch with major Italian news items as well as the usual mix of local, US and international news. They are planning a trip to Turkey. They are not very interested in sport.

Some examples of helpful stereotypes are: *Italians overseas are interested in news about Italy and exchange rates*; *People living in Minneapolis will want to know a good deal about local Minneapolis news as well as about general US and major international events*; *People uninterested in sport do not want to see any but the major sports items*.

Just this type of reasoning is the mainstay of the user modelling in architecture we have devised for large scale filtering systems (Kay and Kummerfeld 1994, 1994a). This is illustrated in Figure 1.

Our goal is to select objects from the store shown in the upper right. This is a distributed collection of many types of objects. For our scenarios, they include: movies; news items and associated film clips; multi-media teaching systems for C including text, graphical simulations of code and objects that can evaluate student exercise solutions.

This object store will generally be external to the individual user's environment. It will be under the control of various agents outside the filtering system. An important link between these outside agents and the filtering is the *publication* of information about new objects. This process means that object descriptors are stored in a Directory Server, as shown at the lower right.

The filtering system is shown in the lower part of the figure. It takes the object descriptions and user models and allows the selected object-descriptors through to the user's local environment where they are once again stored in a directory server.

Within the user's local environment, the filtering process is repeated, this time using the local user model. This would tend to be more detailed than the remote one and it may contain more sensitive information. In addition, more expensive filtering methods can be applied here.

Once the objects have passed their last filtering, they are passed to *OR*, the object requestor. This initiates a message based acquisition of the actual objects identified as needed by the filtering process. These are kept in the *Local Object Store*.

The final presentation to the user is via a user interface that acquires the objects via its server. It customises the presentation with the aid of the user model.

In this model, stereotypes play an important role. This is especially so for the remote filtering process. They can enhance the time-efficiency of filtering if the same filters can be applied for many users. For simplicity, the figure shows just one remote filtering process. In practice, there is generally a sequence of filters that operate for each user. Where there are large numbers of users as well as many objects to be filtered, the appropriately ordered filtering by stereotypes will be critical to effective performance.

Different types of stereotypes

As noted earlier, the user modelling community has made considerable use of 'stereotypes' and yet, the things that have gone by this name differ considerably. This section explores the special character of stereotypes and distinguishes them from other knowledge-based reasoning that

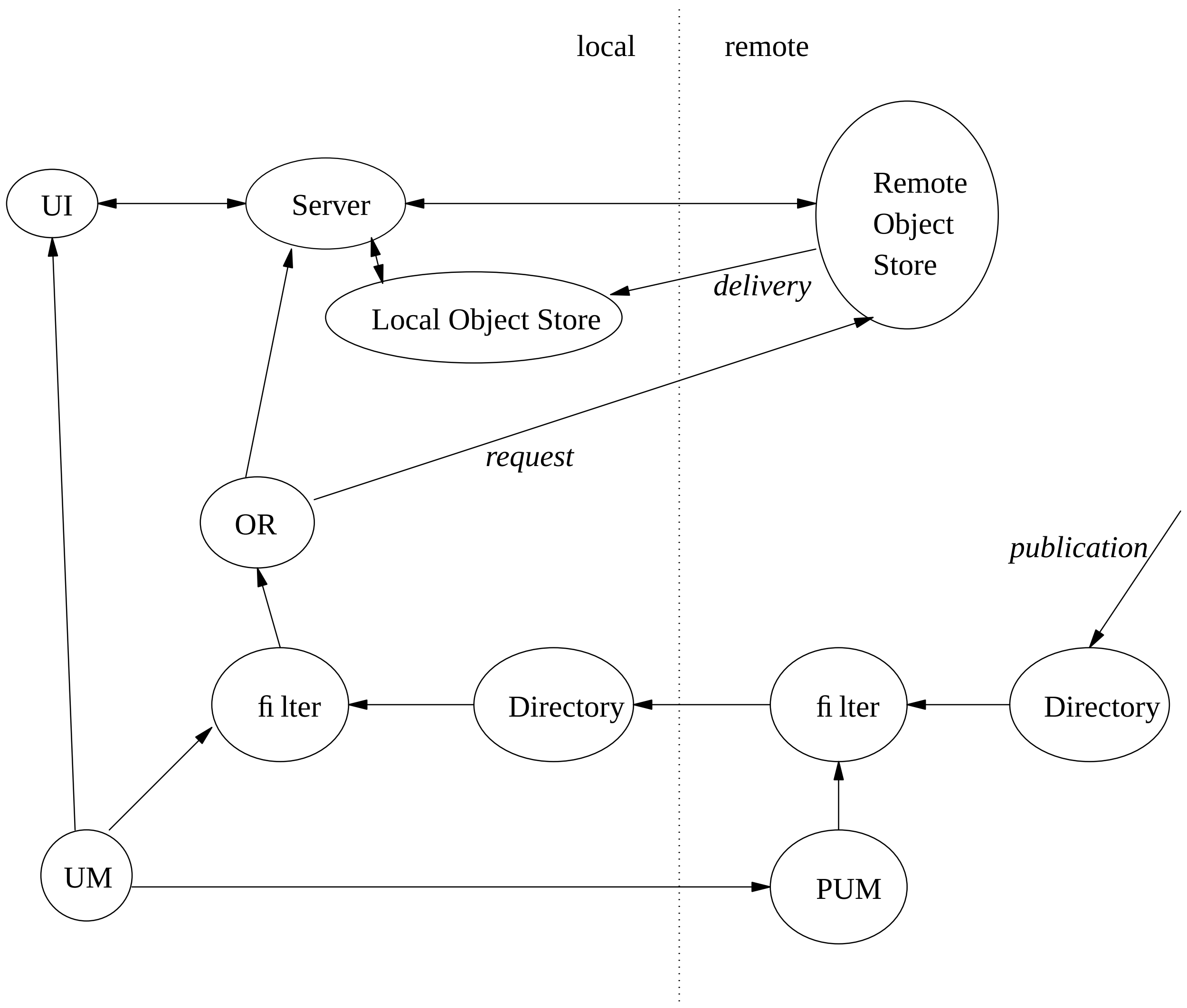

Figure 1: Architecture for a fi ltering system.

is also important for user modelling. From there, it identifi es different classes of stereotypes, some of which have been heavily used to date as well as promising but little explored ones.

User modelling systems that use stereotypes maintain a stock of them in a *stereotype database*. These are the basis for *stereotypical reasoning*.

Typical knowledge-based reasoning has the effect shown as inference 1 in Display 1. If its antecedent is true, it infers the conclusions. These must be retracted if it becomes false. Also, there may be some uncertainty associated with the inference.

In stereotypic reasoning, the antecedent is the function of *triggers* that activates the stereotype (inference 2 in Display 1). The triggers are usually attributes that are readily available. Typically, they include user attributes that defi ne the stereotype. For example, a stereotype for a movie buff would include a trigger indicating that this person has seen many movies.

Once a stereotype is active, we infer the conclusions associated with it (inference 3a in the display). Note, however, that we typically would not expect all the conclusions to apply for every user. It is an essential characteristic of stereotypes that the conclusions are defaults and it is expected that at least some of them will be overridden when other, more reliable, information becomes available about the user. Indeed, they could all be overridden for some users. That is part of the statistical nature of stereotypes: for some users they are inaccurate but their power is that they are useful for many users.

We should also expect some of the attributes in the trigger can be inferred when the stereotype is active: we call these the *essential triggers* (Inference 3b). For example, a stereotype for a ‘programmer’ may have various triggers including ‘being employed as a programmer’, ‘having completed studies as a Computer Science major’

antecedent	–> conclusions true (with certainty/probability A)	(1)
fn($triggers_X$)	–> activate $stereotype_X$	(2)
$stereotype_X$ active	–> $conclusions_X$ true	(3a)
$stereotype_X$ active	–> $essential-triggers_X$ true	(3b)

Display 1: Elements of stereotype reasoning

or ‘having written substantial programs’. Either of the first two of these might trigger this stereotype and its conclusions as well as the third trigger. Since the third trigger is an essential attribute of a ‘programmer’ it is also one of its conclusions. These are Finin’s (Finin 1989) definite parts of the stereotype. This means that essential triggers could be regarded as playing a role on the antecedent or conclusion side of the inference, depending upon whether their values are known.

The remaining element of using stereotypes is reviewing their activation. This may happen either when too few of the triggers are true or when an essential trigger is found to be false. It may be that the system had poor information when it assessed $triggers_X$ for this user. Or, the user may change stereotypes. For example, a user who was a ‘beginner’ may become an ‘intermediate’ level user. Both are common in user modelling. When it is found that an active stereotype is not to be applied, its conclusions are retracted.

This makes stereotypes a form of knowledge-based reasoning where:

- trigger sets often include essential triggers and if their stereotype becomes active when their value is unknown, they are inferred to be true.
- any essential trigger becoming false deactivates the stereotype;
- the conclusion set of a stereotype is a set of default assumptions.

These characteristics are inherent in the statistical nature of stereotypes. They are supposed to represent attributes that apply for many users in a class. This is important for the development of tools for stereotypic reasoning. Firstly, it is to be expected that definition of stereotypes will be heavily dependent upon ad-hoc and statistical approaches. Most of their power will *not* come from strong domain theories. Only the essential triggers are consistently likely to come from domain knowledge.

When is a stereotype not a stereotype?

This section examines the classes of stereotypic reasoning that apply for our filtering task as well as reasoning in this domain that is of a different character. It is written in terms of the task of the movies filtering task. Figure 2 illustrates the discussion.

First, we note that our goal is tightly bound to the database of movies. We want to select from it just those movies that the user will enjoy and we want to filter out all others.

Our user modelling goal can be stated as establishing the values of a set of user model components, each being the user’s actual or inferred rating of a movie. We call these *domain-object* (**DO**) components because they are directly linked to the objects in the domain. These are the actual objects that will be stored in the Object Store (and their descriptions will be in the Directory) of Figure 1.

They are shown as the circle at the upper left of Figure 2. The lines leading into the **DO** circle are the direct sources of reasoning from other parts of the model. The other lines conclude about components that are not part of our goal set but they can affect them indirectly, in two stages of inference.

One approach to achieving our goal is to build a set of user model components that represent *attributes* of movies that the user likes. I call these *domain-attributes* (**DA**) because they model the user’s attitude to attributes of the objects of this domain. Note that these are the attributes stored in the published descriptions of the objects.

For the movies domain, these include the genre, subject matter, actors, director, awards and other material that is commonly available about movies. The filter can get this information from the directory server.

Because they are established by outside agents, **DA** components are beyond the control of the user modelling system and are tightly coupled to the domain objects. The **DA** components are particularly important as they are the basis of a quite simple filtering mechanism: provided we have good descriptions of the domain objects, a good model of the user’s preferences for the elements of those descriptions should be the basis for a straight-forward evaluation mechanism for the objects.

Of course, movie preferences, can be a gestalt assessment of the movie. In that case, as with other art forms, **DA** components are less effective than in domains like news filtering or learning C, where the attributes of an object capture its relevance and usefulness for the user very well. In spite of this, our prototype movie advisor

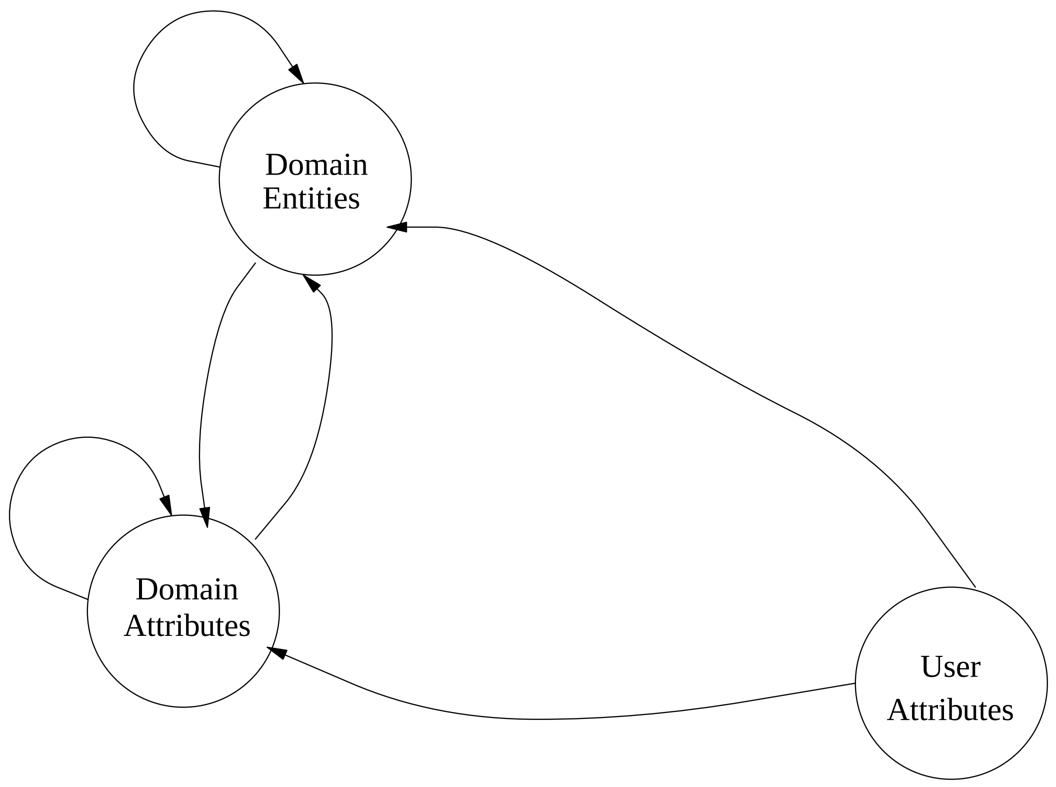

Figure 2: Classes of user model components

was judged by many users to perform quite well. Positive results for **DA** components in this domain make them a very promising basis for filtering.

The third, and last, group of user model components are called *user attributes*. (**UA**) These include the user's age, sex, education, income level and all other aspects we want to model about the user.

For the movies filter, they are the components under the control of the user modelling system. These might even include movie attributes not in the **DA**. However, this will only be useful if the filter is able to determine which movies have those properties. In a domain like movies, where the objects themselves involve huge amounts of effort, the relative cost of defining **DA** attributes is negligible. This makes the **DA** attributes very extensive: powerful filtering should be possible without movie attributes as **UA**.

Since most current user modelling work has not considered architectures like that in Figure 1, it has not distinguished user model components that are tightly coupled to the domain (**DO** and **DA**) as being different from other user attributes (**UA**).

Note that **DO** and **DA** components must be represented with links to the world outside the user modelling system. These enable the application, in our case the movies filtering system, to find the objects and their attributes.

In the many domains where important information about the user comes from a knowledge source outside the user modelling system, the distinctions shown in Figure 2 are relevant. A user modelling system has no choice but to make do with the definitions of attributes that are outside its control. This is important because we would like to separate systems into small manageable units and making the domain inference unit quite separate from the user modelling is one way to do this.

Are all the inferences shown in Figure 2 amenable to stereotypic reasoning? Most can be.

Certainly the inferences **UA** –> **DO** are just those that systems like KNOME (Chin 1989) make when the user who is a beginner is inferred to know only the simple elements of Unix. Note that the elements of Unix are defined outside the system and the domain expert's representation of them is also. But a Unix advisor needs to track which of these the user knows and uses. For simplicity, Figure 2 shows only inferences that reason directly onto domain objects and attributes. But KNOME's *double stereotypes* also support the opposite inference, **DO** –> **UA** where a user is found to know about some sophisticated aspects and this is used to infer they are experts.

When GRUNDY (Rich 1979) used people's descriptions of themselves to deduce the characteristics of books they would enjoy, it used stereotypic reasoning **UA** –> **DA**. In fact, this is the form that is closest to the usual

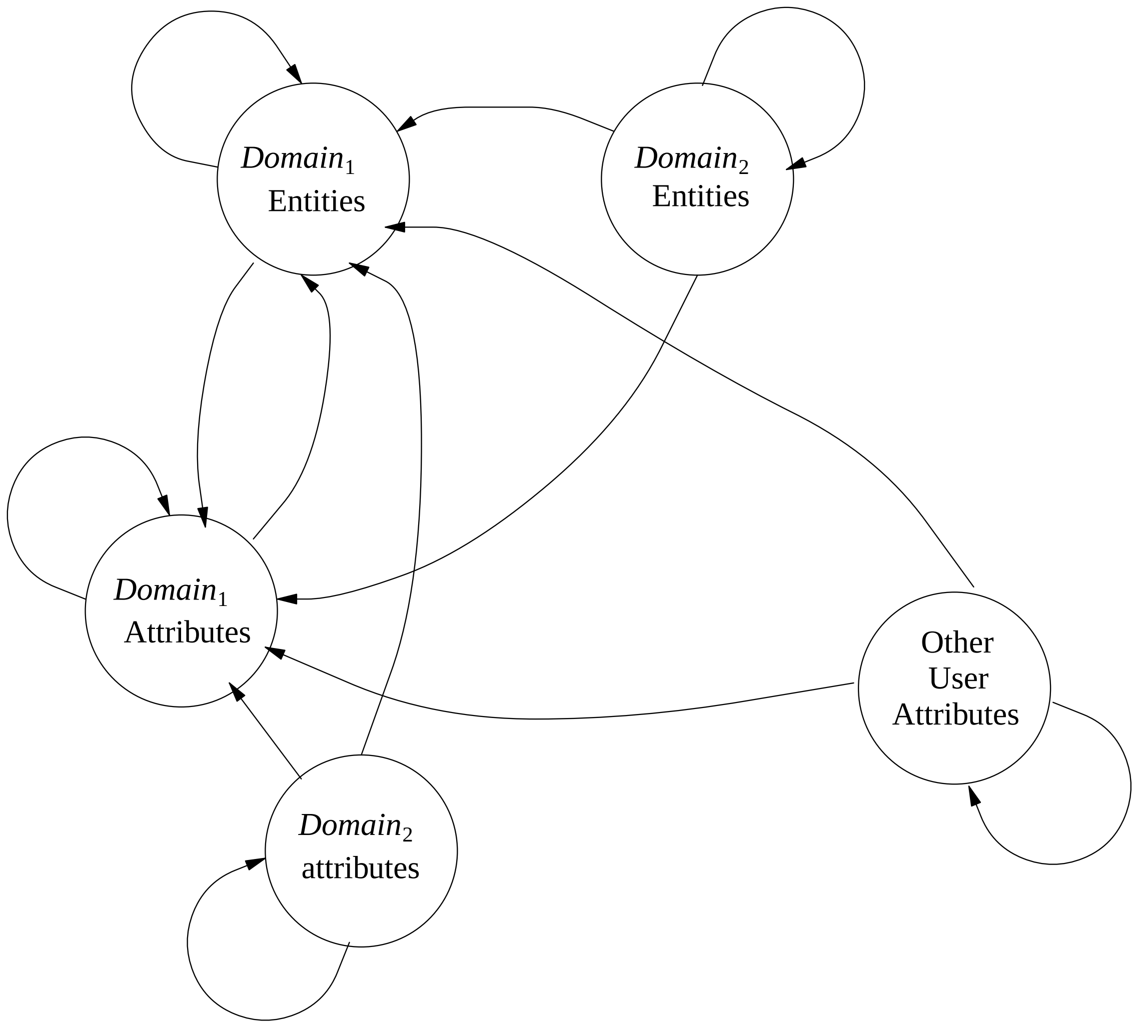

Figure 3: Stereotype reasoning across domains

english use of stereotype. Information that a user is 'sensitive' activates a stereotype that infers several classifications of books that should appeal to them. GRUNDY also used **UA** –> **UA** to infer more user attributes that would feed inferences about **DA** preferences.

Now consider the inferences between **DO** and **DA**. Given that **DA** are the characteristics of the domain elements (in our example, movies) this is not subject to stereotype reasoning. They are completely out of the control of the user modelling system.

Our movie filter will use these inferences. For example, if the user is believed to loathe violence in movies, the filter will discard movies that are classified as violent. This uses the non-stereotypic inference **DA** –> **DO**.

Now the onto-inference, **DO** –> **DO**, offers a particularly promising set of potential uses for stereotypic reasoning. These enable one set of user modelling information to permit inferences about similar ones, like *if you liked movie X, you will enjoy movie Y*.

The appeal of this is partly due to the reduced sensitivity of a simple collection of movie ratings, compared to personal information that would be part of **UA**. Given large sets of such user ratings, stereotypes can be identified.

For the movie filter, it might operate like this. Many users provide a set of ratings of movies. From these we build stereotypes, using techniques like cluster analysis.

Another approach is to identify *opinion leaders* for particular stereotype groups. They are *typical* of particular classes of users. From their ratings of movies, and those for a new user, we can activate suitable stereotypes for the user. Then, as new movies appear, they are rated by our canonical stereotype member and we use this to filter for the individuals in that class. These opinion leaders would serve part of the role of the film critic. Different users would be able to associate a number of these with

	trigger	mapping	people who (dis)like	will (dis)like
Direct Domain	books	$DO_{book} DO_{movie}$	$\{book_i\}$	$\{movie_j\}$
	movies	$DO_{movie} DO_{movie}$	$\{movie_i\}$	$\{movie_j\}$
Domain Attributes	books	$DA_{book} DO_{movie}$	$\{attribute_i\}$ in books	$\{movie_j\}$
	movies	$DA_{movie} DO_{movie}$	$\{attribute_i\}$ in movies	$\{movie_j\}$
			people who have	**will (dis)like**
User attributes	user	$UA_{general} DO_{movie}$	$attribute_i$	$\{movie_j\}$

Table 1: Mappings between different types of user model components

their own filter.

Inferences **DA** –> **DA** might be possible from domain knowledge (for example where some attributes are specialisations of others) or it may be useful to employ stereotypes so that an interview of the user could ask about a modest number of movie attributes and stereotypically infer others.

The situation becomes rather more interesting (and complex) when we take account of other domains. Suppose, for example, that we already have a filter for books. Figure 3 shows the extended classes of inferences about the movies domain ($Domain_1$) when the books filter is domain 2.

Table 1 summarises the meaning of Figure 3's direct inferences onto DO_{movie}. For example, the first line is the mapping from DO_{book} to our goal, DO_{movie} which corresponds to an inference like this: *if the user likes one set of books –> they will probably like a certain set of movies*.

The rows that involve only the movie domain have already been discussed. The new mappings are in the first and third rows. Both of these are clear candidates for stereotypic reasoning.

A similar table can be constructed for the mappings onto DA_{movie}. The most important of these is DA_{books} to DA_{movies}. This maps the meaning of the **DA** components across the domains. Much of this can be based upon a mapping between the semantics, and some may be by stereotypic reasoning.

Inferences across domains are a particularly promising area for stereotypes. This is partly because mappings between disparate areas, created by different agents and with different semantics, seem to be most amenable to ad-hoc and approximate solutions. From the user's point of view, such mappings are desirable, especially if the user has invested considerable time training the book filter: it is desirable that some of the effort should be transferable to other, very similar domains, like a movies filter or advisor.

Existing heavy users of stereotypes

We all do stereotypic reasoning: for better or for worse, we develop default inferences about people and then, on the basis of a little information, we assume much more until we recognise that we need to alter our assumptions. One of the dangers of such stereotypic reasoning is that people build inaccurate stereotypes and are often are slow to recognise the need to relinquish assumptions.

Implicit modelling of users is an inherent design constraint on most interactive software.

There are many industries using stereotypic reasoning as we have defined it: they are deeply involved in constructing, selling and using stereotypes. A brief discussion of some of these is instructive for finding directions for stereotypes in user modelling.

Lending and credit agencies are reliant upon models of the sorts of people who are poor risks. These are typically constructed by various statistical techniques but machine learning is also useful.

Newspapers present the news that editors think will attract their reader population. The structure of a newspaper also reflects the use of sets of sub-stereotypes: for example, the sports section matches the paper's sports sub-stereotype.

Advertising of a product is based upon stereotypes of what appeals to the various populations that are targeted. Different advertisements target different sub-stereotype groups, but presumably none is intended to offend any group.

The widespread sale of mailing lists is based on the stereotype that people whose names are collected in a particular way will generally be interested in a range of items.

There are so many groups building and using stereotypes that we have a considerable base of tools and issues to explore as we apply them to user modelling tasks.

Technical issues for using stereotypes

As Display 1 indicates, there are three main steps in using stereotypes. These involve defining:

- the triggers that activate a stereotype;
- its conclusions;
- and when to retract it.

Constructing stereotypes means defining the stereotype elements that effect each of these. In the user modelling toolkits of the future, the current support for stereotypic reasoning will be enhanced by a range of mechanisms for constructing stereotypes.

The highly statistical nature of the enterprise immediately suggests the techniques that are most likely to be effective. We should expect a growth in the availability of tools for machine learning based construction of stereotypes. This direction is hinted by the tools in Doppelganger (Orwant 1993) that use a range of learning techniques suited for different types of data.

Similarly, following the lead of BGP-MS (Kobsa 1990) ad-hoc stereotypes will be easier to build with good interfaces to construction tools. A similar role can be played by knowledge elicitation tools like cm, which was initially devised to assess learner's deep knowledge (Kay 1986, Kay 1991) but also provides a graphical interface for defining user preferences and assessments of an object (Cook and Kay 1994).

Inter-domain reasoning could be based on a powerful knowledge representation that was the basis of both domains. If this is not a possibility, stereotypes might help. Even where the semantics of two domains are defined consistently, it may need stereotypic reasoning to for inferences between the **DA**s of each. Where the semantics of the two domains were defined independently, stereotypes capture the most natural form of inference.

One form of more subtle inter-domain reasoning is where the two domains are actually the same, for example movies, but the **DO** and **DA** definitions were done by different agents. For example, there may be several 'publishers' of movies and their descriptions. From the user's point of view, it is natural that a filter be able to reason across these 'different' domains. Some elements will map across very simply but where the **DA** definitions are different, it may be that machine generation of stereotypic mappings is the most effective course.

Although this paper has focused on tasks like the movies filter, the architecture of Figure 1 is being used for teaching systems as well. The area of student modelling already makes extensive use of stereotypes at a number of levels. For example, the *expert model* captures a 'typical' expert's view of the domain. In practice, there are differences between expert's domain models but the expert stereotype is useful, especially since the notion of stereotype allows for the fact that there may well be some differences between actual experts, but considerable commonality.

Stereotypes of common misconceptions and their bases are helpful when a teaching system attempts to interpret student actions. The construction of useful learner stereotypes should be possible with similar tools to those needed for our movie filter.

For example, in a series of studies of users of a text editor (Thomas, Benyon, Kay and Crawford 1991, Benyon, Kay and Thomas 1992) we have been building models of user's (very slowly) growing knowledge. To do this, we have monitored a large group of users over three years. Stereotype groups are identifiable and these allow prediction of an individual's knowledge in the long term. Varying forms of statistical analysis enables one to identify a collection of stereotypes that characterise the acquisition of editing knowledge across the population.

We have also been studying the use of various teaching strategies to enhance learning (Parandeh-Gheibi and Kay, 1993) and would like stereotypic reasoning to support selection of the approach to offer each user. With proper tools to manage the monitoring and analysis it should be cost-effective to construct such stereotypes in a range of such applications.

Perhaps the simplest technical benefit of stereotypes is one of efficiency: if a large number of user's models can have a substantial common part stored once in a stereotype, less storage is needed and if the same reasoning is to be applied to many users, we can devise ways to do this process efficiently.

Socio-political issues for using stereotypes

Current widespread uses of stereotypic reasoning in various areas suggests some of the concerns to become important for stereotypic user modelling. For example, most societies recognise the influence of the media and have various forms of control. One can expect similar control to be appropriate where machine based filtering occurs.

At one level, computer use of stereotypes has the benefit that the process is identifiable and can be examined. This offers the potential of greater accountability.

However, the transparency of a complex system is so poor that the possibility of really understanding a sophisticated filtering process is small. This means that we must define systems to be highly transparent and accessible. The user also needs to be able to take real control of their user model. This has been a critical design constraint for the um toolkit (Cook and Kay 1994).

There are substantial bodies of literature concerning the effects of stereotypic reasoning by people, including, for example, negative stereotyping based on gender and race. We need to be cognisant of these as we build systems.

For example, in education, it was observed that a teacher who thought that children were particularly bright treated them in ways that caused them to become brighter than their peers. The teacher was employing stereotypes to reason from student attributes (brightness) to affect the way they saw their students and in the selection of teaching strategy and the ways to interact with them.

When we build teaching systems that offer different teaching strategies, like those beginning to appear (Parandeh-Gheibi and Kay 1993a) how do we take account of this finding? What are dumb-student stereotype models doing to learners? Should we treat the learner as bright, even if they prefer a learning approach used for the ‘less bright’?

Suppose we run a machine learning program on a large collection of user models and the conclusion is that a particular group is identified as having preferences that are considered unpalatable? What tools will help is recognise these?

Conclusions

Stereotypes seem to be a special form of knowledge based reasoning that is particularly useful for reasoning about people. They are best used to establish default beliefs about the user while the system waits to collect something better. They may also offer users a shortcut in building their user model: users can simply choose the stereotype trigger sets that they like best and have a good enough model to let the system work effectively for them.

The other current users and students of stereotypes provide a wealth of methods and issues to pursue in the rich mine of future research into stereotypes for user modelling.

Acknowledgements

The filtering project and architecture are a collaborative project with Bob Kummerfeld. The considerable infrastructure for the distributed filtering and modelling processes as well as several of the user modelling tools rest solidly on his contributions. The work on prototype movie advisor was supported by Telecom Australia Grant Y05/04/34 (BLO/02/02/89).

References

Benyon, Kay and Thomas 1992.
D Benyon, J Kay, and R Thomas, “Building user models of editor usage” in *UM92 - Third Intl Workshop on User Modeling:*, ed. E Andre, R Cohen, W Graf, B Kass, C Paris, and W Wahlster, pp. 113-132, IBFI (Intl Conf and Research Center for Computer Science), Schloss Dagstuhl, Wadern, Germany, August 9-13, 1992 (1992).

Brajnik, Guida and Tasso 1990.
G Brajnik, G Guida, and C Tasso, “User modeling in expert man-machine interfaces: a case study in intelligent information retrieval,” *IEEE Trans on Systems, Man and Cybernetics,* 20, 1, pp. 166-185 (1990).

Brajnik and Tasso 1992.
Giorgia Brajnik and Carlo Tasso, “A flexible tool for developing user modeling applications with nonmonotonic reasoning capabilities” in *Proc of UM92: Third International Workshop on User Modeling,* ed. E Andre, R Cohen, W Graf, B Kass, C Paris, and W Wahlster, pp. 42-66, Deursches Forschungszentrum fur Kunstliche Intelligenz (1992).

Chin 1989.
D Chin, “KNOME: modeling what the user knows in UC” in *User models in dialog systems,* ed. A Kobsa and W Wahlster, pp. 74-107, Springer-Verlag, Berlin (1989).

Cook and Kay 1994.
R Cook and J Kay, “The justified user model,” *UM94 - 1994 User modeling Conference,* p. to appear, Boston, USA (1994).

Finin 1989.
T W Finin, “GUMS - a general user modeling shell” in *User models in dialog systems,* ed. A Kobsa and W Wahlster, pp. 411-431, Springer-Verlag, Berlin (1989).

Kay 1990.
J Kay, “um: a user modelling toolkit,” *Second Intl User Modelling Workshop,* p. 11, Hawaii (1990).

Kay and Kummerfeld 1994.
J Kay and R Kummerfeld, “An individualised course for the C programming language,” Basser Dept of CS Tech Rep - to appear (1994).

Kay and R Kummerfeld 1994a.
J Kay and R Kummerfeld, “Customisation and delivery of multimedia information,” Basser Dept of CS Tech Rep - to appear (1994).

Kay 1986.
J Kay, “Interactive student modelling using concept mapping” in *Proc First Aust Artificial Intelligence Congress,* p. 11 (1986).

Kay 1991.
J Kay, “An explicit approach to acquiring models of student knowledge” in *Advanced Research on Computers and Education,* ed. R Lewis and S Otsuki, pp. 263-268, Elsevier, North Holland (1991).

Kobsa 1990.
A Kobsa, “Modeling the user’s conceptual

knowledge in BGP-MS, a user modeling shell system," *Computational Intelligence,* 6, 4, pp. 193-208 (1990).

Orwant 1993.
K Orwant, *Doppelganger goes to school: machine learning for user modeling,* MIT MS Thesis, MIT Media Laboratory (1993).

Parandeh-Gheibi and Kay 1993.
N Parandeh-Gheibi and J Kay, "Supporting a coaching system with viewable learner models" in *Proc. Intl Conf for Computers Computer Technologies in Education,* ed. V Petrushin and A Dovgiallo, pp. 140-141, Kiev, Ukraine (1993).

Parandeh-Gheibi and Kay 1993a.
N Parandeh-Gheibi and J Kay, "Design of a coaching system with viewable teaching strategies" in *East-West AI Conference*, ed. P Brezillon and V Stefanuk, pp. 293-295 (1993).

Rich 1979.
E Rich, "User modeling via stereotypes," *Cognitive Science,* 3, pp. 355-66 (1979).

Rich 1983.
E Rich, "Users are individuals: individualizing user models," *Intl J of Man-Machine Studies,* 18, pp. 199-214 (1983).

Rich 1989.
E Rich, "Stereotypes and user modeling" in *User models in dialog systems*, ed. A Kobsa and W Wahlster, pp. 35-51, Springer-Verlag, Berlin (1989).

Thomas, Benyon, Kay and Crawford 1991.
R Thomas, D Benyon, J Kay, and K Crawford, *Monitoring editor usage: the Basser data project*, pp. 297-307, Proc MNCC/IFIP Natl Conf on Information Technology, NCIT '91, Penang, Malaysia (June 1991).