Creating personalised systems that people can scrutinise and control: drivers, principles and experience

JUDY KAY, University of Sydney
BOB KUMMERFELD, University of Sydney

Widespread personalised computing systems play an already important and fast growing role in diverse contexts, such as location-based services, recommenders, commercial web-based services and teaching systems. The personalisation in these systems is driven by information about the user, a user model. Moreover, as computers become both ubiquitous and pervasive, personalisation operates across the many devices and information stores that constitute the user's personal digital ecosystem. This enables personalisation, and the user models driving it, to play an increasing role in people's everyday lives. This makes it critical to establish ways to address key problems of personalisation related to privacy, invisibility of personalisation, errors in user models, wasted user models and the broad issue of enabling people to control their user models and associated personalisation. We offer scrutable user models as a foundation for tackling these problems.

This paper argues the importance of scrutable user modelling and personalisation, illustrating key elements in case studies from our work. We then identify the broad roles for scrutable user models. The paper describes how to tackle the technical and interface challenges of designing and building scrutable user modelling systems, presenting design principles and showing how they were established over our twenty years of work on the Personis software framework. Our contributions are the set of principles for scrutable personalisation linked to our experience from creating and evaluating frameworks and associated applications built upon them. These constitute a general approach to tackling problems of personalisation by enabling users to scrutinise their user models as a basis for understanding and controlling personalisation.

Categories and Subject Descriptors: H.2.1 [Information Systems]: User/Machine Systems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Personalisation, User Model, User Modelling, Scrutability, Understandability, User Control, Augmented Cognition, Open Learner Modelling, Reasoning under Uncertainty, Inconsistency

ACM Reference Format:
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Personalised interaction has become possible because it is increasingly easy to collect large amounts of information about people and to exploit it to improve interaction with personalisation. Even ten years ago, there were many personalised services available on the web [Kobsa et al. 2001] and the trend has increased steadily [Pariser 2011]. Personalisation is likely to increase further, both in its extent on the web and in other contexts.
applications and services that operate on many devices. As computer hardware has become cheaper and networking and web access have become widespread, many people already have a whole collection of devices and information stores that constitute their **personal digital ecosystem**. Its increasing numbers of diverse and powerful devices include desktops, as well as mobile devices, such as phones, tablets and laptops, and emerging embedded devices, like tabletops, wall displays and digital appliances. This ecosystem can accumulate huge numbers of diverse forms of personal digital artifacts, such as documents, images, video, email, calendars, contacts, blog posts, tweets and posts at social network sites. It can also accumulate logs of user activity, such as web page visits, web search queries, accesses to digital artifacts as well as data collected by appliances and sensors that make it easy for the user to capture information such as weight or activity.

From the combination of personal data and logs of user activity, it is possible to create **user models**. Early definitions of this term described it as a set of beliefs about the user [Kobsa and Wahlster 1989] and emphasised that it is a distinct part of a personalised application or system, rather than embedded within the logic of the personalisation code. Since then, the large body of user modelling work has included diverse forms of user models. These range from simple collections of flags about the user to sophisticated models that aim to simulate the user’s reasoning. User models represent aspects such as: knowledge, interests, goals and tasks, background, individual traits and context [Brusilovsky and Millán 2007]. For example, one could build a model of the user’s friends by analysis of one or more of: the user’s page at social network sites, email, diary and phone contacts. That model may be a simple list of names, or a more complex representation that aims to model the strength and nature of each relationship. Consider, as another example, a system that teaches the user mathematics. Each time the user loads a new page of tutorial material, the system could infer a greater likelihood the user knows that material. As the learner does assessment tasks, each of these could alter the part of the model representing aspects tested. Such user models drive the personalisation in many classes of systems including personalised commerce, web and desktop search engines, recommenders, personal information management tool advisors, help systems, and teaching systems [Brusilovsky et al. 2007] as well as emerging pervasive computing environments. These drive personalisation that has the potential to improve the efficiency and effectiveness of interaction.

This paper is concerned with **long term user models**. These collect data about a person over a long period of time. Already, there have been many examples of deployed systems that build and use such models over several years. Some of the well known examples include commercial web-based services such as web search engines, customised advertisements on many web sites, and recommenders for diverse products. Notable among recommender systems is the very early, and now well established, Amazon which began with books and diversified to many other products. Others include Netflix for movies, Facebook and Google, which stores details such as the user’s search history. Desktop tools, such as ‘slife’ track the user’s activity on their own computer. This can help the user monitor their time management practices. Mobile applications, such as Google Latitude, maintain a location history for a user. In addition, there are many emerging devices and pervasive computing applications that create such models. For example, one can wear a small activity-sensing device. This can capture fine grained

---

2http://www.slifeweb.com/
3such as http://www.fitbit.com/
data such as steps walked and degree of activity-inactivity. Such devices are increas-
ingly designed to make it very easy for this data to be transferred to a repository. The
user can then review their level of activity over the long term.

These models can, and already do, drive personalisation in many commercial web-
based systems and there is emerging work in pervasive computing applications that
will personalise many other aspects. Broadly, there is a trend towards the capture of
large amounts of user model information, from the diverse set of devices that are in
the user's personal digital ecosystem. This information is held both within the many
storage elements of that ecosystem and within the stores of service providers. Since it
can be used directly for personalisation, this information can be seen as a form of user
model, albeit often simple but large. Equally, it can serve as evidence for building user
models. Indeed, companies such as Acxiom, bluekai, Zoominfo, Infogroup and pipl
\(^4\) are aggregating data about people, from a range of sources for use in various forms of per-
sonalisation and filtering. Widespread collection of personal data and personalisation
comes with some significant problems.

--- Privacy. User models that drive personalisation are personal information. Accord-
ingly, they are subject to the complex mix of national and international legislation
concerning proper management of personal information. This has a common require-
ment that people should have access to and control over their own data [Kobsa
2002; Iachello and Hong 2007; Kobsa 2007; Wang and Kobsa 2007; 2009]. In cur-
rent widespread personalised systems, the norm is that people can neither access
nor control their user models. The legislation reflects the broad agreement that this
is a problem. It is particularly so for the significant minority of people who have high
levels of concern about privacy [Chellappa and Sin 2005; Iachello and Hong 2007].
There is special concern about personalisation associated with pervasive and ubiqu-
quitous computing [Langheinrich 2002; Price et al. 2005] and mobile phones [Shilton
2009a]. It is notable that people appear to have become increasingly concerned about
their privacy being compromised in personalised systems [Westin 2003; Toch et al.
2012].

--- Invisibility. For the most part, current widespread personalisation operates invisibly.
It is at best difficult, and often impossible, for people to determine whether an inter-
face is personalised (or what aspects are personalised). This means people cannot
control personalisation applications and services. So, for example, one may receive a
highly filtered view of the world via personalised news services without realising that
this is so. Similarly, when two users visit the same web site, or use the same applica-
tion, they may not realise that they get different results because of personalisation.
Considerable public discussion has echoed concern about this risk of personalisation
and its potential to create “filter bubbles” [Pariser 2011].

--- Errors in user models. Most widespread user modelling relies on noisy and uncertain
information about the user. This is partly because the raw evidence often comes from
“observations” of the user, for example the links they appear to click on a web site.
It also comes from the complex inference processes driving many systems. These
sources of errors mean that user models commonly contain errors. Moreover, peo-
ple typically have no systematic way to fix these. People may resort to “hunting”
behaviours [Browne et al. 1990], as both the system and user attempt to model each
other. For example, a Wall Street Journal article [Zaslow 2002] reported several cases
where a TiVo created incorrect models, such as inferring the user was gay. It then
gave poor recommendations. The article reported how people tried to overcome this

by trying to find actions that would influence the system to alter its model. There are many sources of errors. For example, if someone else uses your account or computer, their activity may contribute to your user model, making it inaccurate.

— **Wasted user models.** Many devices capture large amounts of user modelling information about people, storing it in various computers, notably ones not owned or controlled by the user. Much of it lies dormant even though it could be useful to the user. For example, if a person buys a pedometer that wirelessly sends their data to the manufacturer’s site, the user can only access it there. They cannot readily combine it with their other personal information to gain a fuller view of their fitness efforts and progress.

— **Control.** In current systems, people are simply not in control of either the personal information that constitutes their user models or the personalisation based on them. This is a core problem that underlies several of those above. In addition, there is evidence that people willingly provide more personal information for personalisation if they feel that they are in control [Kobsa 2007; Toch et al. 2012].

In the next section, we introduce **scrutable user modelling** as a foundation for tackling the above problems. In Section 3, we present a series of case studies from our work to illustrate our vision for scrutable personalisation. It serves as an introduction to the Section 4 which describes the roles for scrutable user models. Then we present a set of broad principles for creation of scrutable user models and personalisation in Section 5, followed by an overview of our Personis implementations of these (Section 6). We conclude with reflections on the current state of scrutable personalisation and broad issues for making scrutability the norm for personalised systems.

2. INTRODUCTION TO SCRUTABILITY

Currently, most personalisation is inscrutable, meaning that the user has no way to discover the details of their user model and the associated personalisation. **Scrutable** user models are designed and implemented so that user can study, or **scrutinise**, the way they work, to determine what information the user model holds, the processes used to capture it, and the ways that it is used. It is similar to the terms **open** [Bull et al. 1995; Self 1999; Dimitrova 2003; Mitrovic and Martin 2006; Zapata-Rivera et al. 2007], **inspectable** [Zapata-Rivera and Greer 2004], **transparent** [Cramer et al. 2008; Hook 2000; Hook et al. 1996] and **intelligible** [Lim and Dey 2009b; Lim et al. 2009; Lim and Dey 2009a]. We chose the term scrutable because it indicates the real effort that a user must make as they study information about the user model and the personalisation processes, in order to understand them. It is closest in meaning to intelligible and may be considered to be a particularly comprehensive form of open, inspectable and transparent model. We now illustrate the notion of scrutable user modelling with two scenarios.

**Scenario 1**

Jim uses an editor inefficiently as he is unaware of its **undo** command. Its coach tells him about **undo**. Jim tries it and continues to use it.

**Scenario 2**

Mynews, Ann’s personalised newspaper presents headlines. She is surprised that today’s top one is **Greens Party Politician Misuses Travel Allowance**.

Scenario 1 worked well, with the coach giving helpful advice based on its model of Jim’s knowledge (and, in particular, lack of knowledge of **undo**). It used this to select a good teaching goal, **undo**, and presented it well. This scenario is in a long term learning context, one that is relevant to the many productivity tools that play such an

---

5"Capable of being understood through study and observation" http://www.yourdictionary.com/scrutable
important role in the workplace. It is reminiscent of Microsoft’s Clippy [Horvitz et al. 1998], arguably the first deployment of individual user modelling to large populations of Excel and Word users. It differs from our vision of scrutable user modelling. Clippy was not scrutable. The user could not see, correct or control their user model.

Scenario 2 is potentially less satisfactory. It is similar to the now widespread web personalisation across diverse products such as Facebook, Yahoo, Google and many personalised news sites. Mynews also has a model of the user, Ann. This represents her news interests and determines how Mynews selects and ranks news headlines. A scrutably personalised system aims to enable people like Jim and Ann to scrutinise their own user models to answer questions such as those in the following examples of six types of questions, in terms of our two scenarios:

1. How did the coach determine I didn’t know undo?
2. How did Mynews determine I am interested in that news article?
3. What else does the coach think I know (or don’t know)?
4. What else does Mynews think I am interested in (or not)?
5. How can I tell the coach what I want to know (or not)?
6. How can I tell Mynews what I am interested in (or not)?
7. How can I tell the coach what I want to know (or not)?
8. How can I tell Mynews what I am interested in (or not)?
9. Why did the coach choose to advise me about undo?
10. Why did Mynews choose that article as the top one?
11. What would the coach do if it judged I knew things when I did not?
12. What would Mynews do if it had errors in my interest model?
13. What do other people I work with know about the editor?
14. What items does Mynews give other people?

The first three question types relate primarily to the user model. The next two deal with the way that the model is interpreted and used. The last involves the models of other people. We now discuss these to illustrate our vision for scrutable user modelling.

Scrutability of the user model

To answer Question Type 1, the user needs an interface to the relevant part of their user model, to see both the evidence source and the interpretation processes determining the value of that part of the model. For Jim’s case, this may be:

— evidence source: the editor has monitored your activity for three years;
— interpretation process: in all that time, you never used undo; from this, the coach infers you do not know about it.

Cases like Mynews are usually rather complex and so the explanation offered to Ann may look like:

— evidence source: Mynews tracks the headings you click on;
— interpretation process: it counts the words in all these articles and compares them with the words in Greens Party Politician Misuses Travel Budget to calculate an interest score.

A fuller explanation would allow Ann to see the actual word scores in that article and then to see how each of these was determined (in terms of a list of all the articles that contributed to the score). If a scrutable user model can enable people to find answers to Question Type 1, this is a basis for tackling the problems of Errors in user models as well as the Invisibility Problem.

Question Type 2 calls for an overview of the whole user model for this application. Jim needs an interface showing the other parts of the model used by the editor coach.
This may be large, reflecting the complexity of the editing tool (and many other software tools that are in wide use today). The model used by Mynews is likely be far larger, showing the full set of interests modelled. Scruetable user models that provide answers to this class of question can help address aspects of the Invisibility Problem and may enable the user to identify Errors in User Models. The overview of a person's user model could enable them to realise there are useful aspects they did even not know about. This could be a basis for learning more and so could tackle the problem of Wasted User Models.

Question Type 3 relates to the issue of Control. Once users can see their user models, and find an error or omission, they should also be able to correct it. For example, if Jim actually does know about undo but has never needed it, he may wish to correct his model. Of course, this could create the possibility that the user will corrupt their model. Indeed, we observed one person did this in our studies of people using a scrutiny interface for the sam editor [Kay 1995; 1999]. One may argue that this is the user's right; after all it is their model (as privacy legislation indicates). Alternatively, it could be the basis for beginning a negotiation [Bull et al. 1995]. Our approach has been to simply treat the user as one evidence source and to add their self-assessment to the model, then allow multiple interpretations of the available evidence [Kay 1995].

Scruatability of the personalisation - use of the model

Answers to Question Type 4 involve both the user model and the application decision processes. In Jim's case, this question calls for an explanation of the coach's choice of the teaching goal, undo, over other things he does not know. For Jim's coach, answering this question may involve:

— the editor model as a whole (as in Question Type 2)
— an explanation of how this was used to select this teaching goal.

In Ann's case, it must explain why this article ranked higher than others:

— the Mynews model as a whole (from Question Type 2);
— Mynews ranking strategy including its model for important and current news.

This explanation would enable Ann to see why her recommendation differs over time and the ways it differs for other people. A system supporting Question Type 4 would help address the Invisibility Problem and potentially problems due to Errors in user models in terms of the way that applications interpret and actually use the model.

Question Type 5 would enable the user to do what-if experiments, exploring how the results would differ after certain changes to the model. The point here is that if the user model is both scrutable and temporarily alterable by the user, this also gives an indirect mechanism for the user to explore aspects of a personalised application that is not itself scrutable. Answers to this class of question address the Invisibility Problem, provide a new use for the model, addressing Wasted User Models and providing a form of Control.

Sharing user models

The sixth class of question involves seeing other people's models. This can be useful in several ways. First, it may help the user see normative models. A simple form of this is typically available in current learning management systems (LMS) which can show the individual learner their place in the class, with the class average and grade distribution. This question relates to sharing the model, since those other people would need to make their models available, even if this is only in aggregate, blurred or anonymised form (as, for example, in [Berkovsky et al. 2007]). Answers to this type of question re-
late to the *Privacy Problem* and are important for enabling people to really understand their own model, so tackling the problem of *Wasted User Models*.

### 3. SCRUNTABLE PERSONALISATION CASE STUDIES

We now present three case studies which illustrate our exploration of how to tackle some of the challenges of creating scrutable user models and personalisation. For each, we begin by discussing the problems it tackles. Then we explain the motivation and user view for the main application used in evaluating the system (each was built on a general purpose framework for creating a class of scrutable personalised systems). We conclude with discussion of the evaluation of the scrutability.

#### 3.1. Case Study: SASY (Scrutably Adaptive Hypertext System)

SASY is a framework to support creation of web pages that are personalised. It tackles the *Invisibility Problem* by providing users with the means to understand and control the personalisation. It supports a very simple but useful form of web personalisation based on selective presentation: all users see some information but other parts appear for some users and not for others. It was used in the earliest web-based personalised systems [Kay and Kummerfeld 1994] and can support personalisation of both the presentation and navigation [Brusilovsky 1996] but would be unwieldy for adaptive ordering or recommenders.

Consider the screen in Figure 1. It was part of an online tutorial system about unix file permissions [Czarkowski and Kay 2006; Czarkowski 2006]. When a user views a page like this, they have no way to know that it has been personalised at all. It is just possible that they will recall that on their first login, they answered questions like: *Do you like to have jokes included in teaching materials?* They may also notice that this page has no jokes. And they may infer that the underlying system has used their answer to the question to omit jokes from this page. Of course, this is rather unlikely, especially if the user answered the question some time ago and if there are jokes only on selected pages.

![Fig. 1. Example of the way that one would typically see a personalised web page. The user has no way to determine that there is personalisation. If they did know there was personalisation, there is no way to determine what has been personalised or how the user model affected the personalisation.](image-url)
Now consider the screen in Figure 2. Note that the largest cell, at the left, has the same material as the previous screen. In addition, it has a link at the lower left: How was this page adapted to you? In our first attempt at supporting scrutability, this was the sole link for users to scrutinise the personalisation [Czarkowski and Kay 2000]. In user studies, people simply did not find this [Czarkowski and Kay 2006]. This situation poses a dilemma for the interface designer. On the one hand, scrutiny support should be unobtrusive, so that it does not distract the user from the task at hand. On the other hand, it cannot be considered to really support scrutability and empower people if they cannot work out how to use it.

This is why we introduced the very noticeable and explicit information in the cells at the right. The Personalisation cell draws the user’s attention to a brief form of the user model components and their values. Note that this interface presents only the current values for the components. For example, the mainGoal allowed the user to indicate whether they were learning this material for the first time or just revising it. The evidence for this component comes from the user’s answers to the start-up questions as shown in Figure 3. If the user does not answer, no evidence is added. If they select either of the two other answers, that causes a piece of evidence to be added to the model for mainGoal. At future sessions, their last answer appears and if they alter that, new evidence is added with that value. In addition, the system supports quizzes and the user’s performance on these contributes to the model of the user’s knowledge. The lower right cell of Figure 2 has a link to the user model.

We now consider what the user sees if they click on the link How was this page adapted to you?. This transforms the page to appear as shown in Figure 4. Content included by the personalisation is now highlighted in yellow and the excluded content appears, highlighted in green. In addition, mousing over either of these regions gives a popup, as shown in the figure, with an explanation of the reason for the inclu-
Creating personalisation that people can scrutinise

This first part of the start up questionnaire that primes the user model values at first login.

![Fig. 3](image1.png)

This case study illustrates scrutable personalisation. As the form of personalisation is quite simple, it is feasible to present the small user model on the same screen as

![Fig. 4](image2.png)

The second component in the Personalisation cell, level, has the value advanced.

This case study illustrates scrutable personalisation. As the form of personalisation is quite simple, it is feasible to present the small user model on the same screen as

ACM Transactions on Interactive Intelligent Systems, Vol. X, No. 4, Article XXX, Publication date: July 2011.
the main content. It is also feasible to show the user the nature of the personalisation. Even so, creating the scrutiny interfaces proved to be quite challenging. We concluded that a major reason for this is that people are unfamiliar with the notion that they can scrutinise how personalisation operates [Czarkowski and Kay 2006]. They ignored, or tolerated, obvious errors in the personalisation, as they are obliged to do in most interfaces they use.

SASY is a framework for building scrutinably adapted hypertext. It evolved iteratively as we created prototypes, then evaluated them to refine the next prototype, so gaining understanding of how to create scrutiny interfaces that people could use effectively [Czarkowski 2006]. We created various adaptive systems including a guide to the personalisation, a holiday planner, TV guide and tutoring systems for parsing [Czarkowski and Kay 2000] and unix [Czarkowski and Kay 2006]. We used two main approaches for our laboratory trials: one asked the user to pretend to be a hypothetical user and the other asked participants to make use the system as they wished. The first was valuable in that we defined the user's high level goals and needs and all participants were highly comparable. The second did not have this attribute, as people could, and did, take many different paths through the personalised information space; the merit of this approach is that it is more authentic. Another possible disadvantage of the first, hypothetical user, approach is that it may have imposed additional cognitive load on participants. Both approaches were extremely valuable in our refinement of the interfaces.

In the main summative evaluation, we created a tutorial for unix file permissions and made it available for students in an HCI class where they needed this knowledge for practical work. The tutorial started with a pre-test, followed by three modules, each with its own test. There was a final post-test and a questionnaire about the personalisation, scrutiny support and seeking free comments. The tutorial system was used by 105 students, with 72 completing the all aspects meaningfully. We had set defaults in the model to values that we expected to irritate users, motivating them to scrutinise the personalisation. So, the system presented jokes and hints, additional advanced material, additional quiz questions, treating them as if they had no knowledge if they had even 1 incorrect answer in the pre-test. At least one use of a scrutiny tool was made by 81 (77%) of students, with 13 (12%) doing 9 or more, with this spread across the tools: the profile tool; the evidence tool with most of this for the model of domain knowledge; using the personalisation highlighting tool and changing the profile. We analysed the timing of scrutiny actions: 108 (28%) were just after completing a module (by 32 users) and most of the remaining actions were before starting a module. (Just 14 actions, 3.6%, were at other times.) We concluded that the use of the scrutiny tools was quite high, in the context of students using a tutorial system for a real learning need. We also concluded that the users were more interested in scrutinising the model of their knowledge than the personalisation of jokes, hints, advanced concepts and extra quiz questions.

The questionnaire indicated that the users did not mind (71%) or notice (22%) the jokes — for hints, the numbers were 60% and 21%; this helps explain why only 4 altered their profiles to remove hints and 8 to remove jokes (of the 72 completing all tasks). About half, 37 (51%), agreed they could inspect what was included or removed in their personalised view (and 13 of them actually did so) and could change their model to control this (and 14 had changed their profile). On the other hand, 11 (15%) disagreed or strongly disagreed. Of these, 7 had used the profile tool to change attributes, and 3 had used the Highlight tool. When asked if they were able to inspect and control the personalisation, 43 (60%) agreed and 7 (10%) disagreed. We asked students who had inspected the personalisation for their main reason. The responses selected were: 42% Curiosity, 6% I did not like how a page was personalised to me, 6% a belief about me
was wrong and 6% selected Other. Those who had not used the tools were asked the reason for this; 36% selected the response Did not feel compelled to do so, 31% selected Trusted the system's default personalisation and 4% selected Other, with 2 commenting too much else to do! and didn't think it would make a difference.

We designed the evaluation to provoke users to scrutinise, by setting what we expected would be poor defaults. Only 2 (2%) users scrutinised evidence associated with jokes, none for the hints, five for inclusion of advanced concepts, and 14% for extra quiz questions. Similarly small numbers altered their profiles; 4 (4%) to remove hints; 8 (8%) to remove jokes; 1 to remove advanced concepts; 6 to reduce the number of quiz questions. Most of the scrutiny activity was associated with the knowledge model, with 37 (35%) of the students doing so, including 11 (10%) checking our incorrect inference they did not know a concept, when they had just correctly answered a quiz question on it with 5 changing their profile. Overall, our provocations had a small effect on user behaviour. This is partly explained by the fact that the users mainly devoted their attention to the learning task at hand. (The post test quiz showed learning gains.) Another reason that the jokes, hints and extra material would not have been particularly irritating is that the interface clearly coded them. This is good interface design, making it easy for people to find what they want to read. One of the criticisms levelled at personalisation researchers is that poor interface design may be the reason that the personalisation is needed; we wanted to avoid that.

Our overall evaluation approach involved a series of lab trials and then an authentic field trial to gain deeper understanding of how people would respond to and use the scrutiny interface. An important part of the design of this study is that the tutorial system met real learning needs. In light of this, we conclude that there were quite high levels of scrutiny: 81 users (77%) scrutinised in some way, 13 (12%) scrutinised 9 or more times and use of all the scrutiny tools. Most scrutiny actions (96%) were just before or right after students finished a learning module, session, pre- or post-test. Thirty-seven (35%) students changed their user model. Notably 10% of scrutiny actions were before the first pre-test, indicating the scrutiny tools were noticeable and encouraged some exploration at first use. The questionnaire responses indicated that 43 users (60%) agreed it was useful to be able to inspect and control the personalisation – 6 (8%) disagreed. Broadly, this class of evaluation is important for gaining understanding of how to create frameworks, interfaces and applications that support scrutatable personalisation.

### 3.2. Case Study: JITT (Just in Time Training system)

Our second case study tackles the Invisibility Problem as it has an interface to its user model, and this has links enabling the user to scrutinise details of the evidence informing it. This can avoid the problem of Wasted user models as it enables the user to learn from the system’s model of their knowledge. It provides two forms of Control. One allows the user to select the interpretation of the user model evidence, potentially avoiding some forms of Errors in user models. The other allows the user to select the personalisation mechanisms.

The system was called JITT (Just in Time Training system). It was inspired by the needs of people who work within organisations that use complex workflows. JITT aimed to provide context-sensitive training materials, relevant to the current stage in a long term task. While such systems typically work within commercial contexts, our demonstrator instance was for the process of writing examination papers in one organisation. This task is very important, especially for large classes. From an institutional perspective, it is critical for assessing the effectiveness of teaching (and learning). From a student's perspective, errors in exams and unexpected, non-standard formats can be very distressing and unfair. There are seven stages in this case: writing ques-
Fig. 5. Main screen of the Just in Time Training interface.

...tions; designing the grading scheme; checks by a domain expert and by administrative staff. Typically academics do this just twice a year, and they may forget the recommended processes, making it representative of many tasks in diverse workplaces.

Figure 5 shows the main screen for the user (judy) who is part way through the process of writing two examination papers. These are listed in the cell at the upper left. If the user clicks the link for one of these, the screen moves to a cell with its details. These provide links to the training materials for each task. The upper right cell, “Important Items” is a list of links to documents that the user has identified as important in previous interactions. The other cell at the right shows the documents viewed with the most recent at the top.

We can already see the two current activities for the user on the screen. Both involve writing exam papers but the user is at different stages in each. For the first of these (SOFT2004 exam), we can see what the user needs to do next: “The marking scheme must be prepared”. Below this are links to the available learning materials, in this case one document. Clicking the link “View workflow” below this takes the user to the screen shown in Figure 6. This is a simple user model interface showing a graphical display of the workflow with the system’s models of the user’s current state in the workflow. In the underlying system, each node has an associated list of concepts that a person needs to know to do that stage of the workflow. JITT also maintains a user model which represents the user’s knowledge of the set of concepts for each workflow. JITT has an algorithm which determines learning paths for the user. This begins by setting learning goals (the concepts needed for the current workflow step). For each concept that the user does not appear to know well enough, it finds the sets of documents that each give a “learning path” that will enable the learner to move from their current knowledge state to the goal state. In general, the workflow and the concept and document spaces can be very large. There may be many learning paths. JITT can support multiple teaching strategies which can rank these differently. The user can...
choose among these. For example, one may have short abstract explanations where another had longer ones with detailed examples. In the case of the exam workflow, the user model is small and there is a quite small space of training materials, all in the form of documents.

If the user clicks on the “Profile” link in Figures 5 or 6, this takes them to the page shown in Figures 7 and 8. In the top of this page (Figure 7) the user model displays the user's knowledge as one of 6 levels, with all five boxes white for cases where the user is modelled to have no knowledge of a concept, and all blue for mastery. For each component of the model, there are links to the “raw evidence” and to the “resolver explanation”. The latter is a text description of the way that the evidence was interpreted to give the knowledge level value displayed. The term resolver refers to the task of resolving the value from a set of evidence. Figure 8 shows the system is using “Trust Most Recent”, which simply takes the value of the last piece of evidence. The user could also select the “Simple Average”. (Other options appear when the user clicks the widget.) Below this, the figure shows a cell for “Teaching Agent Selection”. This is currently the default “JITT Base”.

Clicking on the “raw evidence” link in Figure 7 gives the screen in Figure 9. The top piece of evidence was a self-assessment. When the user is presented with training material, JITT asks the user to assess their own understanding of each of the concepts in the material. This is valuable from a learning perspective since it helps the user appreciate just what aspects the material was intended to teach. It is also valuable in helping the learner reflect on how much they actually feel that they have learnt. A learner may well think that they have read a document carefully but not realise that they did not really take in one of the concepts listed. Equally, they may have simply decided to skim some parts of the document. This makes the self-assessment valuable both from a learning perspective and as a way for the learner to contribute to their learner model. In this case, the learner has rated themself as knowing this concept at the highest level, “could teach it to others” [Tamir 1984]. The second piece of evidence was added to the model by JITT when the user was presented with a training document. JITT’s resolvers weight such evidence as weaker indications of learning than the self-assessment scores.
JITT aimed to provide a framework for personalised teaching with scrutability of the model and user control to select the evidence interpretation (resolver) and teaching strategy. To evaluate it, we designed a user study to assess:

— usability of JITT, including the scrutiny elements;
— understandability of the available choice in Teaching Agent and Resolver;
— broad user responses to the scrutability.

ACM Transactions on Interactive Intelligent Systems, Vol. X, No. 4, Article XXX, Publication date: July 2011.
We conducted a think-aloud evaluation where 10 participants played the role of a new staff member writing their first exam. Since this is a time-consuming task, we provided a complete draft exam, with marking scheme and other associated materials for a course based on the C programming language. We asked participants to use JITT to identify any problems in the draft, according to the set policies. There were 12 problems and finding these required participants to make use of key concepts across the workflow. After completing this task, participants completed a questionnaire for background information.

Given the nature of the task, participants needed considerable programming expertise. We recruited five academic staff and 5 tutors (teaching assistants) of whom three had graded exams (T1,2,4). All had considerable familiarity with the examination process.

We analysed the success and behaviour of the participants in finding the 12 problems, and in using the system. The performance results indicate that participants engaged with the task, making serious use of JITT. The five tutors T1-5 found 6, 3, 10, 10 and 5 errors respectively (average 6.8) and the academics A1-5 found 5, 2, 2, 6, and 5 respectively (average 4.2). The tutors did better than the academics for eight of the twelve problems with academics sometimes distracted, notably by disagreements with the policies. As one might expect, the academics sometimes relied on experience, rather than the system. Observations of participants, as they thought aloud, indicated that they could discover and understand all elements of the interface. Free form responses to the questionnaire indicated that all agreed the system presented materials in a useful way and that JITT would be useful both when writing an exam for the first time and as a refresher later.

We now consider the scrutability and control aspects. All participants demonstrated understanding of the scrutability interfaces by using them to see their model, the evidence and to consider the resolvers and teaching agents available. All participants were able to explain the difference between the two available teaching agents. All but one participant preferred the one providing complete information, where the personalisation simply reduced the font for materials assessed as known. Participants were asked about the value of making the user model available. Table I summarises the results, grouped by the problems we identified in the Introduction (and shown in italics). Overall the evaluation integrated assessment of the usability and learning value of JITT along with the scrutability. This aspect of the evaluation design is important since these are integrated in any authentic use and they interact. The observations indicated that users were able to use all elements of the interface, including the scrutiny and control elements. An important caveat for these results follows from the highly

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Knowledge Level</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-10-09 21:09:15</td>
<td>could teach to others</td>
<td>self-assessment after viewing CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:24</td>
<td>Positive</td>
<td>viewed content CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:25</td>
<td>Positive</td>
<td>viewed content CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:15</td>
<td>understand well</td>
<td>self-assessment after viewing CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:43</td>
<td>Positive</td>
<td>viewed content CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:44</td>
<td>Positive</td>
<td>viewed content CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:51</td>
<td>could teach to others</td>
<td>self-assessment after viewing CONFIDENTIAL labelling of exam papers</td>
</tr>
<tr>
<td>2003-10-24 11:45:57</td>
<td>Positive</td>
<td>viewed content CONFIDENTIAL labelling of exam papers</td>
</tr>
</tbody>
</table>
Table I. Summary of JITT Question: What value, if any, did you see in making the user model available?

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
<th>Participant Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Invisibility</strong></td>
<td>good to know what system believes</td>
<td>T3 T5 A4</td>
</tr>
<tr>
<td></td>
<td>user can see if it is accurate</td>
<td>T3</td>
</tr>
<tr>
<td></td>
<td>if model large and difficult to understand</td>
<td>A5</td>
</tr>
<tr>
<td><strong>Errors in user model</strong></td>
<td>important when something goes wrong</td>
<td>A3</td>
</tr>
<tr>
<td></td>
<td>user can see if it is accurate</td>
<td>T3</td>
</tr>
<tr>
<td><strong>Wasted user models</strong></td>
<td>could help identify knowledge gaps</td>
<td>T4 T5 A1</td>
</tr>
<tr>
<td></td>
<td>user can see how much the system thinks they know</td>
<td>T3</td>
</tr>
<tr>
<td><strong>Privacy</strong></td>
<td>not helpful for user but helpful for supervisors</td>
<td>T1</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>allows change of resolver if they think it is more accurate for them</td>
<td>T3</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>very valuable</td>
<td>T2</td>
</tr>
<tr>
<td></td>
<td>not essential but useful as the system guides you</td>
<td>A2</td>
</tr>
<tr>
<td></td>
<td>unclear in this case</td>
<td>A4</td>
</tr>
</tbody>
</table>

technical background of the participants. Even so, JITT seemed usable and useful and the participants saw value in the scrutability of the model, with participants’ free comments linking this to all five of our identified personalisation problems.

3.3. Case Study: Personalised Indoor Location Information Interface

This case study tackled the Invisibility Problem in the context of a pervasive computing application. It aimed to provide personalised information about people’s indoor location, using sensor information available without requiring people to carry special devices. This system relied on many sensors, which gave noisy and inconsistent evidence about people’s location. It also made use of quite complex reasoning, based upon a personalised, semi-automatically generated ontology which modelled the user’s conception of the space and their links to other people.

From the user’s perspective, the system enabled people who work in a large building to determine the location of other people who work there. This had two main roles. One was as a form of “calm”, ambient display that people could glance at a few times a day to gain awareness of who is around. The other goal was to enable a person to locate a particular individual. For example, suppose Alice and Bob work on different floors. They are both working late towards an impending project deadline. Alice reaches a point where she wants to speak with Bob. If he is at his desk, she will walk there. If he is the tearoom, she will call his mobile phone. However, if he is at home, she will send email; since it is late, she would not want to call him at home.

In our series of systems and interfaces called Locator, we tackled several technical and interface challenges. We created systems level support for capturing sensor data [Carmichael et al. 2005; Assad et al. 2007], reasoning about available evidence to infer the user’s location [Carmichael et al. 2005; Assad et al. 2007; Niu and Kay 2008] and personalising the interface that presents the information [Niu and Kay 2010].

Figure 10 shows the version of Locator that makes use of ontologies for several of these aspects. It automatically built a layered ontology to help reason about the noisy, unreliable, uncertain and inconsistent sensor evidence [Niu and Kay 2008]. It created personalised ontologies [Niu and Kay 2010] to personalise the information presented.
Creating personalisation that people can scrutinise

The top half of the screen serves the role of the ambient display. The user has selected Level 3 of the building and this shows dots representing the location of several people. The brightness of the dot indicates the freshness of the location information, with the bright red dots being based on recent sensor data and the lighter (pink) ones being older. Below this is a detailed description of the location of these users (anonymised in the figure). This information reflects the granularity of the actual best location estimate. So, for example, User A is modelled as being at “his desk” while the information for User B is only accurate to the level of the wing. Three people have been hidden as Locator predicts that this user is not interested in their location. Locator aims to present place names that are meaningful to the user. So, for example, if the user works closely with user L, they are likely to know L’s workplace desk or office. This makes it more natural to refer to that location as L’s desk than use the desk identifier.

If the user clicks a person’s name, Locator brings up the explanation shown in Figure 11. This enables the user to scrutinise the personalisation of the description of User L’s location as being at “his desk”. It presents the system’s reasoning. First it reports the user model value for the location of L as a particular desk (3W32). It then explains the reasoning for using “his desk”: that 3W32 is L’s desk and that the system infers that this user knows L’s desk. It also provides the evidence it used to infer this; that L appears to be a colleague and the user seems to be familiar with the area of L’s desk.

Figure 12 shows the interface as the user scrutinises the personalisation by clicking all the “why” links to drill down to further details of the explanation. The first of these reports the evidence used to infer the location. This is from the sensor (called niu@pc-g61b-1). User L had installed this on their desktop and it sends a piece of evidence about the user’s location once a minute. (Such evidence can be noisy if other people use that machine. The system had several other forms of sensors, such as sensors for Bluetooth associated with phones and laptops.) The second explanation is for the association between User L and Desk 3W32. This came from mining the “postgrad student directory”. The third explanation reports the two pieces of evidence for the system’s reasoning that this user and L are colleagues. First, their work areas are near
Fig. 11. Locator interface after user has decided to scrutinise further, by clicking the personalised description of a place. This presents the explanations for the personalisation.

Fig. 12. Explanation from Figure 11 as the user scrutinises the personalisation, by clicking all why? links to show all the explanations.

each other; this was determined from the automated analysis of the building plans and was based on the reasoning that people whose work areas are close are likely to know each other. The second piece of evidence is that both the user and L are on the same internal email list. The final explanation, for familiarity with the area, is based on evidence that the user has been in this area recently.

The discussion to this point has glossed over the mechanisms for reasoning about the sensor data for location. We now briefly outline the ways that our ontological reasoning mechanisms support this. Figure 13 shows another version of the Locator interface. As in the interface already described, it had a similar ambient display of the building with
Creating personalisation that people can scrutinise

Fig. 13. Another version of the Locator interface where the user could select the resolver they wished to use to interpret the sensor evidence about user's locations.

dots showing the users. It also had a list of the users, their current location and the last time they were detected by one of the sensors. In addition, there is an ‘explain’ link that presents a screen with the last ten pieces of location evidence for that user.

In this case, Locator offered users the option to decide which resolver the system should use in its interpretation of the location evidence. At the upper left, we see that the system is currently using “MostRecentEvidence”, a particularly simple resolver which uses just the newest location evidence for each user to conclude their location. In order to introduce the more complex resolvers, we use the examples of evidence shown in Figure 14. The ones described as “niu-mobile” are based on Bluetooth sensors that detect the user Niu's mobile phone. (When the user decided to join the system, they registering their phone, and then the system created a phone model, associating the phone's MAC-address with the user so that evidence about this phone would contribute to the location component of the user's model.) The second piece of evidence came from a login sensor, which determined that the user had logged into a particular terminal. The fourth piece of evidence is, as in the earlier example, a sensor the user installed on his desktop.

We now briefly describe the key technical and interface issues associated with the resolvers we created to interpret evidence like that in Figure 14. These make use of reasoning based on a layered ontology. The most general layer was hand-crafted and it models buildings in general; so it could be reused for a system in other buildings. The next is built automatically, by tools that analyse available resources, such as building plans, mailing lists and other documents like the building and personnel induction manuals. While this model is not useful for other buildings, the processes for building it could be. The final layer is dynamic, coming from the sensor information. (For detailed descriptions of these resolvers and their evaluation, see [Niu and Kay 2008] and

ACM Transactions on Interactive Intelligent Systems, Vol. X, No. 4, Article XXX, Publication date: July 2011.
for details of the creation of the ontologies, see [Niu and Kay 2010].) These resolvers can conclude that the first three pieces of evidence are not actually in conflict, with the second being a finer-grain location within that from the first and third. The last five pieces are in genuine conflict since levels 1 and 3 are different places. We created resolvers that can deal with this by taking the majority value and combining this with the ontological model. We also created resolvers that take account of the known properties of the sensors [Carmichael et al. 2005]. For the case of Bluetooth sensors, these include: they sometimes fail to detect a device; they take up to 10 seconds to detect it; and they work through concrete floors, meaning that a Level 3 sensor may detect people on either that floor or the ones above or below it. In summary, in this work, we created quite complex resolvers for interpreting noisy, uncertain, unreliable and conflicting evidence. We did not attempt to create comprehensive explanations of these complex processes. However, we did create interfaces that enable a user to decide whether to use any of these or a far simpler one that is readily understood. Our evaluations indicate that different resolvers were more accurate for different people [Niu and Kay 2008] with simple resolvers that are easy to explain being effective for some people while the more complex ontological resolvers were more effective for others.

We designed the user study to evaluate Locator’s scrutiny support [Niu and Kay 2010]. It assessed: the accuracy of the user modelling; the value of the personalised place names; understandability of the personalisation explanations. For the evaluation, we created an non-adaptive version of Locator. We designed a small qualitative think-aloud study. To reduce inter-subject variability, half the participants used the adaptive system first, the others used the non-adaptive one first. All completed an online post-questionnaire. With the non-adaptive Locator, participants were asked to identify locations of people and places familiar to them (assessing the personalised location labels) and name people they preferred to be hidden (assessing the accuracy of the adapted hiding). For the adaptive system, they were also asked to state What kind of information do you think the system needs to gather in order to display “X’s desk” instead of “Desk 3W32” for Xs location? and then to explore the system to assess if they were right. (To do this, they needed to use the scrutiny elements.) They were asked to determine why one of the personalised elements was displayed and why another was hidden. The nature of this system meant that the personalised display was different for each user. Participant responses to the questionnaire indicated: the explanations were understandable; they are important for personalisation (although one participant rated this 5 on a 7-point Likert scale, commenting they did not care how the system achieved its personalisation); and that the system explained what participants wanted to know (although for the case of hidden people, three participants wanted more detail of the reasoning in cases where the system was incorrect.) Comments pointed to a preference for more control, including setting the threshold sensitivity for displaying people (one participant) and providing a way to indicate a person should not be shown (two participants) and five of the eight participants wanted more control of the

<table>
<thead>
<tr>
<th>Person</th>
<th>Event Time</th>
<th>Location</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>nui</td>
<td>1 minute ago</td>
<td>Level 1 West</td>
<td>nui-mobile</td>
</tr>
<tr>
<td>nui</td>
<td>1 minute ago</td>
<td>Desk 3W32</td>
<td>Login Sensor</td>
</tr>
<tr>
<td>nui</td>
<td>3 minutes ago</td>
<td>Level 1 West</td>
<td>nui-mobile</td>
</tr>
<tr>
<td>nui</td>
<td>5 minutes ago</td>
<td>Desk 3W32</td>
<td>device_nui_at_pcg0101</td>
</tr>
<tr>
<td>nui</td>
<td>3 minutes ago</td>
<td>Level 3 East</td>
<td>nui-mobile</td>
</tr>
<tr>
<td>nui</td>
<td>3 minutes ago</td>
<td>Level 1 Middle</td>
<td>nui-mobile</td>
</tr>
<tr>
<td>nui</td>
<td>4 minutes ago</td>
<td>Room 304</td>
<td>nui-mobile</td>
</tr>
</tbody>
</table>

Fig. 14. Example of conflicting evidence about a user’s location.
information presented, to select who to show/hide, add missing evidence and modify incorrect system beliefs. Four participants expressly valued being able to discover and scrutinise the reasons for hidden information. Overall, all participants were able to use the scrutiny interface to do the four tasks involving finding people and then explaining how the various aspects of the personalisation worked. The questionnaire showed participants found the explanations were understandable, gave the information they wanted and are important. The nature of the user modelling meant that participants had elected to use one of our Locator interfaces (and so had a user model). This limits the generalisability of the results. Nonetheless, the work points to the feasibility of providing scrutable interfaces that are usable and likely to be valued by users.

4. ROLES FOR SCRUTABLE USER MODELS

To this point, the paper has introduced our vision for scrutable personalisation with scenarios and case studies. We now move to the general roles for scrutability, summarised in Table II. It shows how these relate to the problems identified in the Introduction using the following abbreviations, Privacy for the Privacy Problem; Invisibility for the Invisibility Problem; Errors for Errors in user models; Wasted for Wasted user models; and as The Control Problem is linked to all of these, we omit it.

Table II. Roles for scrutable user models

<table>
<thead>
<tr>
<th>Control of the modelling processes</th>
<th>Problems addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inflow — information allowed into model (ground inferences)</td>
<td>Invisibility</td>
</tr>
<tr>
<td>Reasoning — inferences within model (internal inferences)</td>
<td>Invisibility</td>
</tr>
<tr>
<td>Use — parts of model available to an application</td>
<td>Privacy, Invisibility</td>
</tr>
<tr>
<td>Reuse — multiple applications using parts of model</td>
<td>Privacy, Invisibility, Wasted</td>
</tr>
</tbody>
</table>

| Validity | |
| Correctness — enabling the user to validate/rectify model | Errors |

| Re-purposing | |
| Navigation — user model as overview of large information spaces | Invisibility, Wasted |
| Influence — based on explanation of personalisation | Invisibility, Wasted |
| Learning — user models as mirrors | Wasted |
| Usability — personal, comparative, long-term usability | Wasted |
| Programmer accountability — programmer mindfulness of user | Invisibility |

Control of the modelling processes

Privacy of personal data means that people should be able to restrict its use as they wish [Palen and Dourish 2003]. This makes it tightly linked to issues of control. If a user can scrutinise their own model to gain an understanding of it and the associated processes, this can be the starting point for controlling these. There is an increasing body of work on providing user control in personalised systems. For example, in a user study for an adaptive news system [Ahn et al. 2007], users liked the control available when the user models (or profiles) were open to them. Similarly Paramythis, Weibelzahl and Masthoff identify scrutability as one of the criteria on which to evaluate personalised systems [Paramythis et al. 2010]. Scrutability has important roles for control associated with user model interoperability [Carmagnola et al. 2011]. Table II has four rows for the main aspects of control of the modelling processes.

— Inflow involves access to details such as the source of each piece of information in the model and the time the information was added.
— *Reasoning* refers to the processes used to reason from the raw information. Some of this may be very simple. For example, in Figure 15-left, the key at the bottom of the figure shows a mapping between the number of Google searches each day and the colour saturation in the display. In many cases, the user model inferences are far more complex than this. Then it is more challenging to create interfaces to support scrutiny of the inference processes. For example, in a learning context based on a Bayesian user model [Zapata-Rivera and Greer 2004], learners could explore a visual form of the model graph, showing links between these and the size of each node reflecting its value.

— *Use* involves enabling the user to see all the uses (and potential uses) of the user model. This is the foundation for the user to control what information may flow out of the model. So it is a basis for controlling the privacy of the model. It means the user can see which parts of the model are available to a particular application and how those are used within the application. This is an essential foundation for controlling the ways that the model is used in practice. We saw this in the three case studies.

— *Reuse* refers to use of the same parts of the model by different applications. Many personalised applications have their own user model, and this is not available to any other application. Yet user models may be useful in multiple contexts. Scrutability and user control can facilitate such reuse.

### Validity

One obvious role for scrutability is to enable the user to find and then address *Errors in user models*. In fact it is so obvious that it scarcely needs discussion; it is an essential aspect of user control. This role for user models was recognised early. For example, Csinger described his video customisation system [Csinger 1995] as scrutatable as it showed the user parts of the user model so they could correct these. Similarly, the Doppelganger user modelling shell [Orwant 1994] and applications build upon it used a range of automated mechanisms to acquire information about the user. Orwant argued for user access to the model to ensure its accuracy and created several interfaces to display various models. Much of the work on Open Learner Models, discussed below, gives the learner the opportunity to correct or challenge the system’s model of them.

### Re-purposing user models

Table II identifies five roles that scrutatable user models could play, by using electronic personal data, the problem of *Wasted user models*. People are increasingly leaving digital footprints as they interact with systems. That data can be transformed into scrutatable user models that have important uses that go beyond personalisation.

*Navigation* refers to the potential for the user model to serve as an interface to an information space. An elegant example of this appeared in ELM-ART [Weber and Brusilovsky 2001], a system which teaches about programming in LISP. The “course-map” for the system was rather like a contents page in a conventional book. However, it actually showed the underlying user model. It colour-coded each entry, using a traffic light metaphor. It used green for the aspects the learner was modelled as ready to learn. Red indicated those where the learner lacked the required pre-knowledge. Black marked those already mastered. We can see a quite similar notion in Figure 15. On the left is the calendar visualisation of a person’s search activity on each day. Clicking the day presents details of that activity. This is a visualisation of the user’s Google search activity. In such interfaces, a scrutatable user model can facilitate navigation of a large information space; at the same time, navigation enables the user to see their user model.
Creating personalisation that people can scrutinise

Fig. 15. Examples of user models for navigation. At the left, a model of user’s daily Google search activity. At the right, the Narcissus user model navigates a large group-information-space of integrated wiki, issue tracker and source code control system, showing daily activity on these. In both cases, the more intense the colour, the more activity on that day. By clicking a cell, the user navigates to associated information.

In an example that provides greater user control, Narcissus [Upton and Kay 2009] is a visualisation of a model of user activity on a group collaboration platform with integrated tools: a wiki; version control system; task allocation and tracking (often called issue-tracking). An example screen is shown at the right of Figure 15. Narcissus shows each user’s activity, on each day of the period requested, for each of these tools in terms of the colour intensity. The figure shows models for 5 users. For each, the three parts correspond to the three tools. Clicking on any square presents a list of the links to that activity (shown at the right of the screen) and clicking on any of these takes the user to the actual content, for example the actual diff-display showing just which lines the user added to the wiki. In this case, the user can control the level of activity associated with each colour intensity level. This helps users navigate what can become a very large and complex information space.

Influence refers to the potential for scrutability to serve as a positive influence on people. Research has shown that people are more likely to accept personalised recommendations that are accompanied by an explanation [Herlocker et al. 2000; Tintarev and Masthoff 2007; Cramer et al. 2008]. This seems to link to issues of trust, as well as understanding and control. It constitutes an additional role for scrutability.

Learning. The role of scrutable user models for learning relates to the potential of a user model to operate as a mirror, showing the user a view of themself for the aspect(s) modelled [Kay 1997]. The SMILI framework characterises the purposes and diverse nature of such models that “invite the learner in” [Bull and Kay 2007]. Notably Bull has done extensive research on open learner models, their uses and varied interfaces. A selection from this diverse and substantial body of work is: negotiated models where the learner could challenge the system about parts of their user model and they were then invited to demonstrate their level of knowledge [Bull et al. 1995]; a mobile phone interface to support learning on-the-go so learners could share and compare models [Bull and Mabbott 2006] as a starting point for more learning; exploration of sharing of learner models [Bull et al. 2007]; creation of different interfaces for children and parents so that parents could support their children with maths homework [Lee and Bull 2008]; and helping university students understand their long term learning [Bull and Gardner 2009]. Mitrovic found students learnt more in an online course for SQL when they had open learner models, and there was a stronger effect.

---

6 The platform is trac: http://trac.edgewall.org/.
for weaker students [Mitrovic and Martin 2006]. Dimitrova demonstrated the power of a more complex form of interactive model as a learning aid [Dimitrova 2003]. John Self identified various forms of openness that can enhance learning [Self 1999]. OLMs offer the potential for improved support for several metacognitive processes, such as reflection, self-knowledge, self-monitoring and planning [Bull and Kay 2008].

Such learning, based upon reflection on a suitable presentation of a model of personal data can go beyond the formal classroom setting. This has been described in the following terms: “Almost imperceptibly, numbers are infiltrating the last redoubts of the personal. Sleep, exercise, sex, food, mood, location, alertness, productivity, even spiritual well-being are being tracked and measured, shared and displayed” [Wolf 2010]. In a recent critique of lifelogging, Sellen identified five key benefits for memory that such capture technologies offer: recollecting, reminiscing, retrieving, reflecting, remembering intentions [Sellen and Whittaker 2010]. While lifelogging differs from user modelling, it shares this potential to support these forms of learning and augmenting cognition. The notion of personal informatics [Li et al. 2010] is similarly based on the collection of personal information in a form that can support reflection, to achieve self-knowledge.

Personal usability

Usability. Typical HCI usability measures assess whether users can complete their tasks successfully and recover from errors, based on analysis (in discount usability evaluation methods such as cognitive walkthrough or heuristic evaluation) and studies (such as think-alouds and field trials). Monitoring actual use, over the long term, in the field, to build user models can provide new forms of usability assessment [Kay and Thomas 1995; Hilbert and Redmiles 2000]. Scrutability is key to several aspects of this. Personal Long Term Usability, based on an interface to a scrutable user model can enable a person to answer questions like: “What is my pattern of use of this tool?” and “What is there to know about this tool, compared with the way I have used it?”. This can be a foundation for individual learning by navigating the scrutable user model [Cook and Kay 1994; Cook et al. 1995]. If others are willing to make their models available, a scrutable model can support Comparative Long Term Usability answering questions like, “How does my model compare with that of others like me?”. Scrutability can support Use-based Recommendation. For example, Linton and Schaefer [Linton and Schaefer 2000] tracked use of Microsoft Word to build user models and used comparative models to drive recommendations of things that people may find useful to learn. The reasoning was that a person who never used a facility is more likely to benefit from learning about it if it is heavily used by people with a similar job role. A suitable form of scrutable user model could make this information available to people as a basis for planning their learning. Of course, these forms of usability differ from the classic HCI view of usability because they may lack the important link to the user’s intention and task. However, classic usability, with its focus on the generic user, does not give the user understanding of the usability of a tool for them, as an individual.

Programmer accountability and awareness of the user. In our earliest work in designing scrutable user models for people’s learning of a text editor [Kay 1990; 1995; Kay and Thomas 1995], the goal of scrutability made us mindful of the user who would view the model. So, for example, we departed from the typical distinction between knowledge and misconceptions. Instead, we defined the user model in terms of knowledge, for those aspects the system designer considered correct. We chose to use the term, belief, to model aspects where the user appeared to have a different approach from the designer. For example, we modelled whether people quit the editor by killing the window; we considered the quit command was a better way to quit (as it gives potentially valuable messages, including warnings about unsaved edit buffers). In time, we
learnt that our users did not read or understand such messages. So, based on their context, understanding and needs, their belief that it was appropriate to use the faster kill-window quit was reasonable. Similarly, we considered the user, even as we chose names for parts of the model. Broadly, we were mindful of the user as we considered each design decision about the user model. Such awareness is valuable in helping the system designer to maintain a focus on the user’s view and need.

5. PRINCIPLES FOR CREATING SCRutable USER MODELS

The last section discussed why scrutatable user models are valuable, in terms of the roles that they can serve. Now we describe how we can achieve scrutability in user modelling, starting with the broad set of principles listed in Table III. These come from the analysis that underpinned even the first of our work [Kay 1999], and they draw on the work of others, particularly work on user modelling shells and frameworks [Kobsa 2001; Kobsa and Fink 2006] as well as our own experience outlined in Section 7.

Table III. Overview of Principles for Scrutable User Modelling and Personalisation

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parallel design</strong></td>
<td>Design of representation and interfaces in parallel</td>
</tr>
<tr>
<td><strong>First class citizen</strong></td>
<td>User models exist outside a single application</td>
</tr>
<tr>
<td><strong>Context-based interpretation</strong></td>
<td>Each application and context may interpret the user model differently</td>
</tr>
<tr>
<td><strong>‘Client-side’</strong></td>
<td>Personal digital infrastructures redefine the notion of client-side</td>
</tr>
</tbody>
</table>

**Parallel design** refers to the need for the designers of personalised systems to consider both the interface issues and the user model representation and reasoning, together, from the beginning of the design process. This is essential if the user is to be able to scrutinise the underlying reasoning of the system at a high level of fidelity. By this we mean that the user should be able to see the actual details of the system reasoning and of the user model. Suppose a particular representation for reasoning is chosen and associated scrutiny interfaces created and it seems that people simply cannot use these interfaces effectively. It will be necessary to revisit both the choice of representation and the design of the interface. Broadly, this principle is in line with the considerable body of research in providing explanations in intelligent systems. (See, for example, the earliest expert systems which explained their reasoning in terms of the chain of rules used [Clancey 1983] to more recent work in conjunction with recommenders [Glass et al. 2008], architectures for explainable AI [Core et al. 2006], and the engineering of service-oriented adaptation [Koidl and Conlan 2008]). That work confirms that some representation and reasoning approaches are easier for people to understand. These are likely to be more amenable to scrutatable user modelling when suitable interfaces are created. For other representations, it may be infeasible to support scrutability beyond a high level description.

**First class citizen** This refers to the need to shift from the dominant existing paradigm where each application has its own user model, designed for its sole use. Figure 16 shows the transformation we propose. At the left, it shows the current situation: a user interacts with many personalised applications, each with its own user model. Each user model is locked away within its application. Our vision, illustrated at the right of the figure has two key differences. First, the user model is not trapped within any one application. Rather it is a separate entity that the user controls. Second, there is an interface that enables the user to scrutinise their user model to answer questions like those in our scenarios. Making the user model a first class citizen, under the user’s control, can facilitate reuse of the model.

**Context-based interpretation.** This refers to the challenge of enabling a user to determine how their model is used, or will be used, in practice. Even a simple user model...
may be used in different ways, depending on the context or application. This is important for supporting scrutability, since a user is likely to need to understand their user model as it operates in a particular context. This means the user needs only to scrutinise the small part of the model that is relevant to that context. It is also easier to create effective interfaces that are tightly coupled to interpretation in that context. This is especially so if the user model is interpreted differently in different contexts.

We initially tried to create more general scrutiny interfaces [Cook and Kay 1994; Kay 1995; 1999]. For example, these showed user as knowing undo if they had told the system that they knew it. However, in such interfaces, there was no way the user could see that our coaching system interpreted the user model differently; it concluded that a person did not know an aspect if they never used it. In subsequent work, we have linked the scrutiny interface to the context of use. Often this is the particular application, or the current context within it, as in the first case study, where SASY showed just the parts of the model relevant to the personalisation on the current page.

“Client-side” control. We now consider what appears to be an emerging notion of ‘client-side’ control. Previously, client-side referred to personalisation where the user’s model was stored on the user’s own computer, rather than on a remote server. This has advantages for privacy [Kobsa 2007] but at a cost; the client-side model cannot make use of other people’s models as needed for collaborative recommenders and service providers may have a business model that is incompatible with it. The notion of client-side user modelling can be viewed as changing, as indicated in Figure 16 where the user model can be seen as a separate entity, and client-side control means that this is under the user’s control. As the user’s personal digital ecosystem involves increasing numbers of devices as well as multiple stores of personal information, client-side storage might be considered to include all the places where the user has control over the information store. This is relatively straightforward in the case of a personal desktop computer since client-side user models would be stored on that machine. However, rich network connectivity means that it can be very convenient to have personal informa-
tion stores that are on many different devices (such as phones and laptops) and also in the 'cloud'. In terms of commercial pragmatics, this broader view of client-side or client-controlled user modelling may be compatible with pragmatic issues described above. The shift needed from current practice is that the user should have access to and control over their user model, regardless of where it is stored. In terms of scrutability, this means that the user needs to be able to find all the relevant parts of their user model, and then explore them. Many major web-based services that store personal information already provide an API to that store. This makes it feasible for programmers (but not ordinary users) to create tools that access such user models and provide a comprehensive scrutiny interface that incorporates parts of the model, wherever they are stored. These tools could also copy the model to storage that is more directly under the user's control. The user model as first class citizen, controlled by the user, rather than particular applications, may constitute a new form of client-side user model.

6. ARCHITECTURE FOR EVIDENCE-BASED USER MODELLING

We now move further into the system builder's tasks and how to create scrutable user models. We argue for the high level architecture and representation illustrated in Figure 17 (refining the right side of Figure 16). This architecture shows the user, at the top, can interact with applications and also use a scrutiny and control interface. It emphasises management of information about the user in a similar manner to that advocated for privacy [Spiekermann and Cranor 2009].

When an application operates as an evidence source, it “tell”s evidence to the model, as indicated by the arrow, labelled with tell on the figure. Evidence may take many forms. For example, a GPS location sensor may send the user's current location as a pair of values giving fine-grained latitude and longitude. A wireless bathroom scale may send the user's weight in kilograms. An editing application may send details of each command used. And, taking an example from the widely used web applications, a click on the advertisement for a book may send evidence indicating interest in it.

When the user model receives the tell, its inflow control layer assesses whether the application is allowed to add evidence to the model. This is shown in the figure as a shaded block to indicate that it operates as a selection barrier, permitting the addition of evidence from allowed sources and blocking others. This is becoming an increasingly important element of emerging pervasive user modelling, based on automated collection of information by sensors [Langheinrich 2002; Spiekermann 2008].

Similarly, the arrow labelled ask operates when an application “ask”s the user model for information about the user. An outflow control layer determines what evidence, and what form of it, is allowed for this application. This layer is also shaded because it can restrict the information allowed to the application asking. For example, suppose an application asks for the user’s location and that there is a series of values from a GPS evidence source. At this stage, the available evidence may be filtered. For example, the user may want some applications, and some people, restricted to location values from the last day, within business hours and the location values may be translated to coarse grained values. This mechanism is to address the Privacy Problem.

The figure shows a second layer controlling outflowing values. This is not shaded as applications can decide which of the mechanisms they need to use here. This enables the application to select parts of the model and the particular interpretation tool. For example, in the case of location, there may be a choice of a symbolic location, such as {“at work”, “not at work”} or values of longitude and latitude.

We call this evidence-based approach accretion-resolution (AR) [Kay 1990; 1995; 2000a]. The evidence from allowed evidence sources accretes, meaning that it is simply added to the model. Later, when a value is needed by an application, this value is resolved by interpreting the allowed evidence. In the case of noisy, unreliable and un-
Fig. 17. Overview of architecture for evidence-based user model. The user interacts with conventional applications, or is tracked by sensors. These may “tell” evidence to the user model with an inflow control layer of the model determining what evidence is allowed into the main evidence store. Applications may also “ask” for information about the user. The result is determined by controlling the evidence to this application as well as its choice of the allowed mechanisms for selecting parts of the model and interpreting the evidence. The model can use “internal” inference to create new evidence. At the right is the scrutiny and control interface.

certain evidence, as is common in many areas of personalisation, the resolution process may be challenging (as, for example, in the Locator case study which used ontologies to improve reasoning about the user’s location and the people to display). AR supports the principle of Context-based interpretation identified in the last section.

There is an additional flow, marked internal in the figure. This uses inference to create new evidence about the user. Such sources of internal evidence should be distinguished from external evidence (also called ground inferences since they are the foundation for all internal inference). There are risks here for privacy and user control; when external, ground evidence informs internal inference and evidence, the latter should carry the same access controls.

6.1. Design of evidence representation

There are many ways that one could implement the core ideas of this architecture in a user modelling system and associated personalised applications. As part of our Parallel design of the representation and scrutability interfaces, we have refined the general AR approach, requiring that each piece of evidence has at least the following:

— a time stamp indicating when the evidence was added to the model (under control of the model) both to enable reasoning based on this timestamp and to support historic scrutiny;
— an optional time stamp provided by the source (for example to indicate the time the application actually collected the evidence) for the same broad reasons as the system timestamp;
— the user model value, for example true if the user indicates interest in a news article. In general, different sources may contribute different types of values, such as booleans, certainty scores or symbolic values.
Creating personalisation that people can scrutinise

The source identifier, useful in three ways: for the inflow control, allowing only registered and authenticated sources to contribute evidence; for reasoning based on the sources, for example, taking account of their reliability; and for scrutability since this should be available in explanations of the evidence and associated reasoning.

Evidence type - our analysis of interaction led to the following important distinctions [Kay 1995]:

- Explicit ground inferences: come directly from a person, for example, when they explicitly answer a question about their preferences or knowledge;
- Implicit ground inferences: are based on monitoring, logging and other forms of observation;
- Ex-machine ground inferences: track what the machine (actually an application) has told the user, so the system can avoid repetition and assess the success of its actions;
- Stereotype internal inferences: for statistically based inferences that are so important in user modelling [Rich 1983; 1979; Kay 2000b];
- Inferred internal inferences: based on other forms of inference within the model, typically based on knowledge bases such as those captured in Bayesian networks.

The evidence type can be used in explanations to the user and can be combined with the source identifier, for a light-weight form of interpretation of evidence (Figure 17).

There has been work that has used representations with some of these properties. For example, CUMULATE is a server which accepts data from learning systems [Yudelson et al. 2007]. Zapata-Rivera et al [Zapata-Rivera et al. 2007] used “evidentiary arguments” in different forms for different people in designing “active reports” for open learner models. An evidence-based approach was used to tackle semantic heterogeneity for interoperability in user modelling [Carmagnola and Dimitrova 2008]. In data fusion [Bleiholder and Naumann 2009], an approach that has much in common with accretion-resolution has been described as “conflict-ignoring strategies”.

6.2. Representation of the user model

Building from this abstract architecture, we now argue for a set of principles to guide the design of a representation to support scrutable user model reasoning. Table IV summarises these.

<table>
<thead>
<tr>
<th>Trade-off</th>
<th>Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>The representation should be as simple as needed for scrutability</td>
</tr>
<tr>
<td>Adequacy</td>
<td>It supports the reasoning needed for personalisation</td>
</tr>
<tr>
<td>Partition</td>
<td>Complex parts of the representation should be decoupled</td>
</tr>
<tr>
<td>Namespaces</td>
<td>It needs sets of context-dependent namespaces</td>
</tr>
<tr>
<td>Semantics</td>
<td>The model must be able to explain the meaning of each element modelled</td>
</tr>
<tr>
<td>Accountability</td>
<td>Keep the external evidence in original form</td>
</tr>
<tr>
<td>Inconsistency</td>
<td>Model may maintain evidence that is contradictory</td>
</tr>
<tr>
<td>Multiple interpretations</td>
<td>Each application may interpret evidence differently</td>
</tr>
<tr>
<td>Late binding</td>
<td>Interprets evidence only when needed</td>
</tr>
</tbody>
</table>

**Trade-off** refers to the trade-off between simplicity and adequacy, that may well be in conflict and the role of partitioning to address this trade-off. The Simplicity principle is motivated by the need to create interfaces that enable the user to scrutinise...
their user model so that they can understand it. It seems wise to start with the simplest representation that can still support the forms of reasoning needed. The Adequacy principle acknowledges the need for sufficient power within the representation to model the user effectively and to support reasoning as needed by the personalised applications. The Partition principle allows the design to have some parts of the user model that use simpler reasoning which is amenable to full scrutability. Other parts may use more complex reasoning for which it is difficult, or impractical, to create a comprehensive scrutiny interface. For example, much learner modelling uses a very simple overlay model (mapping the learner’s knowledge against the system’s model). It is relatively straightforward to create scrutiny interfaces for these. However, some parts of the reasoning may call for a more complex representation, such as a Bayesian Network, or one based upon sophisticated real-time machine learning. There is ongoing research in generating explanations for a range of reasoning mechanisms, for example [Lim and Dey 2010]. This principle proposes that these different mechanisms be kept separate so that the user could decide to disable parts that are not scrutinizable. This would match the observation that different people have different preferences in terms of control [Jameson and Schwarzkopf 2006]. It may be particularly important for pervasive computing applications where the convenience of personalisation may be valuable enough for many users to relinquish some control [Barkhuus and Dey 2003]. If inscrutable and uncontrolled aspects are partitioned, systems can be created to enable the user to determine whether they want to use these, or just the scrutinizable parts. A similar case has been made for design that ensures the user can understand intelligent systems, to make parts of them transparent and predictable [Hook 2000].

**Ontology** refers to the aspects associated with the semantics of the model. The Namespaces principle calls for partitioning the model into parts which each have their own ontology. This means that the concept can have different meanings in different parts of the model. The Semantics principle requires that a scrutability interface should be able to explain what is modelled about the user. In Scenario 1, this means the system should be able to explain the meaning of the part of the model for the user’s knowledge of the undo command. This goal motivated our work to create a scrutinizable lightweight ontology based on analysis of available online dictionaries [Apted and Kay 2004]. We used this to generate the ontology of a user model and for visualising it [Apted et al. 2003]. This meant that we could automatically generate explanations of the meanings of components of the model; this could make use of the dictionary definitions, already written for people to learn about the meaning of the concepts. In addition, we could create a visualisation of the relationships between concepts, and combine this with dictionary definitions and a description of the ontological reasoning [Kay and Lum 2005]. There may also be a need for personalisation of the explanations provided for the ontology. For example, we explained elements of the sam text editor differently, depending upon the user’s modelled knowledge and expertise [Cook and Kay 1994].

**Accountability** calls for the representation to maintain all external evidence provided to the model. By contrast, many systems simply keep a single value for each component modelled, altering its value with each new piece of evidence, and discarding the details of the actual evidence. By keep the actual evidence, the system can explain the modelling processes. For example, the user may be surprised that a system operates differently today from the way they recall that it operated last week. So they may wish to see the value of the model as it was last week as well as now. In the case of our editor coach of Scenario 1, evidence came from analysing logs of use. Since evidence from internal reasoning might be regenerated, it may not be necessary to keep that.
Interpreting evidence involves the reasoning that the system performs, based on the available evidence. The Inconsistency principle reflects the fact that people may have inconsistent beliefs and that the system may have inconsistent information about the user. For example, this may be due to lack of context within the model, such as when the user likes to watch certain movies with children and has different preferences for movies to see in the company of other adults. There may be inconsistent evidence because of the nature of the aspect modelled, especially in learning contexts [Kono et al. 1992]. For example, a learner may do a series of tasks that measure their knowledge of a certain concept. They may make correct guesses when they actually do not know the concept. They may make slips when they know it. A typical learning process might start with many incorrect answers, but some correct ones and, as the learner reaches higher levels of mastery, they may start to consistently give correct answers (bar slips). The Multiple Interpretations principle relates to the Context-based interpretation Principle and the need for different interpretations of the same evidence about the user, depending upon the application and other aspects of the context. Late binding refers to delaying the interpretation of evidence until it is needed. Just as the actual evidence is important for accountability – discussed above – keeping it in full makes for flexibility of reasoning, with multiple interpretations available for different applications and contexts. This facilitates achieving the Inconsistency Principle.

6.3. Interfaces
Our last set of principles relates to the design of user interfaces for scrutability. These are listed in Table V.

<table>
<thead>
<tr>
<th>Interface Principles for Scrutable User Modelling and Personalisation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td><strong>Ontology</strong></td>
</tr>
<tr>
<td><strong>Scrutiny</strong></td>
</tr>
<tr>
<td><strong>Inflow</strong></td>
</tr>
<tr>
<td><strong>Give value</strong></td>
</tr>
<tr>
<td><strong>Outflow</strong></td>
</tr>
<tr>
<td><strong>Outflow process</strong></td>
</tr>
<tr>
<td><strong>Use</strong></td>
</tr>
</tbody>
</table>

Interfaces for scrutable user models should enable the user to determine what information the model holds and the processes that formed it. If a model is large or complex, the interface needs to provide an Overview interface as a starting point and additional Scrutiny interfaces to enable the user to drill down on the details. This follows the recommendation to present visualisations of large collections of information using an overview first, with zoom and filter, then details on demand [Schneiderman 1996]. For example, we created the VLUM (very large user model) interface [Apted et al. 2003; Uther and Kay 2003; Uther 2002; Kay and Lum 2005] for use in contexts where hundreds of aspects about them are modelled.

Taking the view that model values are determined from both external and internal evidence, the scrutiny interface should enable the user to see Inflow to the model. An example of our work related to this principle was a concept mapping tool; this was a user model elicitation interface, based on concept mapping, an established technique for people to externalise their conceptual understanding [Novak 1990]. Our Verified Concept Mapper (VCM) [Cimolino et al. 2004] enabled the user to lay out concepts to show hierarchical relationships (with more general ones higher) and link them to form propositions, such as ‘a whale’ is a ‘mammal’ or ‘a whale’ is a ‘fish’. When the user
was satisfied with their map, they clicked the analyse button and this presented them with:

— a list of the inferences the system had made and would store in the user model;
— things to check, where this was a list of propositions in the map or pointers to omissions.

So, for example, if the user had created the proposition ‘a whale’ is a ‘fish’, this would be on the list of inferences. The creator of the system could also add a message to the list of things to check, such as Are you sure you have linked ‘whale’ to the right concepts? This helped the user avoid being modelling inaccurately, when they accidentally clicked the wrong concept and create an unintended proposition. The spirit of the design was to both ensure the user could see what was inferred about them and to help them verify that they had thought about key aspects to be used for building the model. In this sense, it was a form of verified input to the user model.

At the point where the user can see both the current value of a component of the model and the evidence for it, they should be able to give a value. This adds a piece of evidence for that component. Since the A-R representation keeps all such evidence, resolvers can treat this as a self-assessment, rather than ground truth; such evidence can be interpreted in combination with other evidence, for example based on user behaviour. The Ontology principle calls for an interface that explains the meaning of the concepts modelled. For example, in our earlier scenario, it should explain what the undo command does (and hence what the model of knowledge of it means).

Interfaces for use of model have similar requirements to those just discussed for the viewing of the model itself. However, they must also take account of the context of use. This is likely to simplify the interface if the model may be used by multiple applications because the interface can focus on parts used by the current application. It may usefully restrict the displayed information to the current context. In other senses, the Overview and Scrutiny of use are as in the above discussion. The interface support for Outflow should enable the user to determine which parts of the model are released in general. That for the Outflow process must do this for just those parts used in this context. It must also show how these are interpreted. A scrutiny interface for Use should show what an application actually does in light of the user model values released to it. This constitutes a very pragmatic form of additional information about the truest meaning of the components of the model in terms of their actual use and the practical implications of their values for the user.

7. PERSONIS REALISATION OF THE ARCHITECTURE: INTEGRATING SYSTEMS, REPRESENTATION, INFERENCE AND INTERFACES

This section illustrates the ways that we have created systems based upon the abstract principles and architecture of the previous two sections.\(^7\) We summarise a selection of them in Table VI and refer to these as the “Personis” family (although individual versions and tools have various names.) Each row of the table refers to one system, with its key references. The italicised rows are for systems that were discussed in the case studies. We bring them together, ordered by the date of the main publication to show the relationships between them. The dates are important because they reflect the influences of that time and our growing understanding of the challenges of supporting scrutatable user modelling. We begin with the systems aspects (the second column) then discuss the other two columns together as they are so tightly linked.

\(^7\)We have described the systems and representation evolution from a learning perspective elsewhere [Kay and Kummerfeld 2010].
Creating personalisation that people can scrutinise

**Systems**

This dimension relates to the underlying operating systems, networks and hardware, the latter two having seen important changes affecting the ways that one needs to create personalised systems. One of these has been due to the increase in the availability of mobile computing devices. This means that people today may well own a personal smart phone, tablet device and portable computer and they may use multiple fixed desktop computers. We are seeing further transformations, with even richer personal digital ecosystems of devices. This is beginning to include various sensors, some worn or carried and others embedded in the environment. It may also include embedded devices such as tabletops and wall displays. There are important systems challenges if these are to provide effective personalisation [Krüger et al. 2007] as well as new possibilities for capturing information about the user. The growing diversity and complexity of these ecosystems creates new challenges for scrutability and user control [Shilton 2009b]. For example, they suggest the user may want to ask questions like:

— Where is my user model stored?
— How can I ensure that some user models (or parts of them) are stored only within my home?

Table VI begins with the **um toolkit** [Kay 1990; 1995]. Scrutability was a key driver for its design, leading to the basic accretion-resolution representation. It was intended to support learning by modelling use of the sam text editor. Learners used a single ‘large’ (at the time) computer as their main (often only) computer. They interacted via a terminal and had a strict quota on file space.

In this framework, the user model was stored as text files in a directory hierarchy in the user's own file space. This formed a directed acyclic graph of namespaces, we called contexts. For example, we created models of the student's knowledge of their main text editor (sam), their knowledge of unix and, for some users, their movie preferences. These were stored in a directory with each model in its own subdirectory. Within these were additional subdirectories, for sub-contexts. At the leaves, files held the model's components and the evidence associated with each. Some aspects, such as knowledge about regular expressions, was common to the editor and unix and these aspects were linked.

This approach used the access controls of the underlying unix operating system, at the level of directories and files. Each person's model was restricted to the user. When the user started a personalised application, it ran under the user's own id and so could access the model. We instrumented the sam editor to log each user's activity. We took care in designing this to model only knowledge of the editor [Cook et al. 1995], avoiding any details of the content edited. The log was stored in the user's filespace. (Users were informed how to stop the modelling processes at this stage, by ensuring the automated removal of this log.) A process ran each night to analyse the log and add new evidence to the user model. Later, a coaching program used this model to identify aspects that the learner did not know and from these, chose coaching actions.

The term, context, for the nodes in the hierarchical model tree reflects that it holds parts of the model relevant to one context, such as the sam editor context. The same component name can have different meanings in different contexts. For example, we used this to deal with the somewhat different ways that regular expressions work in unix and the sam text editor. Each context has a set of associated resolvers. Each defines one way to interpret the evidence associated with a component. In summary, contexts partition and structure the user model components and define the resolvers allowed for determining the value of components.
Table VI. Personis family evolution. The left column is the system name and key publications. The second column
summarises the systems level elements, the third has representation and inference mechanisms and the right col-
umn has interface aspects. Italicised systems were described in the earlier case studies.

<table>
<thead>
<tr>
<th>System name</th>
<th>Systems</th>
<th>Representation inference</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>um, sam editor study [Kay 1995]</td>
<td>Single Machine</td>
<td>Accretion-Resolution</td>
<td>Explanation system, Mirror</td>
</tr>
<tr>
<td>sam editor coach [Cook et al. 1995]</td>
<td></td>
<td></td>
<td>Individual Usability</td>
</tr>
<tr>
<td>SASY I [Czarkowski and Kay 2000]</td>
<td></td>
<td>Scrutable Hypertext</td>
<td>Subtle Link to Explanations</td>
</tr>
<tr>
<td>Personis [Kay et al. 2002]</td>
<td>Server</td>
<td>Partial user models, personas</td>
<td></td>
</tr>
<tr>
<td>VIUM/SIV [Uther 2002; Apted et al. 2003]</td>
<td></td>
<td></td>
<td>Large user model overview interface</td>
</tr>
<tr>
<td>VCM [Cimolino et al. 2004]</td>
<td></td>
<td></td>
<td>Scrutable Elicitation</td>
</tr>
<tr>
<td>Personis [Carmichael et al. 2005]</td>
<td>Ubicomp Generalisation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JITT [Holden et al. 2005]</td>
<td></td>
<td>Scrutable Teaching Strategies</td>
<td>Scrutiny and control interfaces</td>
</tr>
<tr>
<td>SASY II [Czarkowski 2006]</td>
<td></td>
<td></td>
<td>Application-Level Scrutiny and Control</td>
</tr>
<tr>
<td>PersonisAD [Assad et al. 2007; Carmichael 2009]</td>
<td>Distributed, Active</td>
<td>Re-active</td>
<td>Locator Interfaces</td>
</tr>
<tr>
<td>PERSONAF [Niu and Kay 2010]</td>
<td></td>
<td>Personal Ontologies</td>
<td>Scrutiny interfaces</td>
</tr>
</tbody>
</table>

In this implementation, each context file was stored as text that was intended to be human-readable. (Similar claims are made today when information is stored as xml but our format was simpler and clearer. Even so, it is notable that our approach shares the use of a text file format for storing information.) This was in the spirit of scrutability. While we created interfaces for scrutinising user models our design ensured a user could choose to view the model directly. (This approach clearly allows the user to edit the model, potentially corrupting it.) The form of the text files matched closely to the formatting of the scrutiny interfaces.

The Personis server [Kay et al. 2002] moved to a quite different architecture. A single server held the models for many users. A protocol enabled an application to establish a connection to the server, then to “tell” evidence to the server so that it could be stored with the relevant component and to “ask” for the value of a component. At this stage, we added the inflow and outflow control layers of Figure 17. This was an addition to the existing control in interpretation, based on the resolvers available for the model context. We also introduced two new selection mechanisms. The partial model defines a subset of the components within a single context. The other, more general mechanism, is a persona. It operates across contexts. In the earlier um toolkit, the context level defined the allowed resolvers and access; the persona allows for greater flexibility. It links a set of components with a resolver, to define one restricted view of the user. This
means that different applications can access their own subset of the model and its own interpretation of the evidence associated with that.

To enable users to carry their models on a mobile phone, we created the Secure Persona Exchange (SPE) architecture. It enabled the user to load a model that was provided by a trusted authority [Hitchens et al. 2005]. For example, a university could provide a signed model for a student's enrolment or transcript. The user could then send this to a pervasive application offering a service, such as bookshop on the campus. The authority could also sign "partial models" so that the student could release just this subpart of the model. The user could alter their model if they wished. But then, that version would no longer be signed by the authority and so the application may not trust it. In this work, the trusted authority was also responsible for defining the user model ontology and sharing this with service providers to create their own applications.

Pervasive computing has created a substantial body of research that aims to provide new services for everyday life by making use of many sensors and other devices in a user's environment. The sensors often collect information that aims to model people. For example, it may capture data about their physical activity. There have been many, many architectures defined, and some implemented, for managing pervasive computing's sensor data and applications. It is somewhat surprisingly that these had not been linked to work on user modelling frameworks until we adapted the accretion-resolution representation to model devices and places as well as people [Carmichael et al. 2005]. This meant that the Personis framework could represent and reason about all of these – people, places and devices – in a consistent manner.

The next stage of our work enabled Personis to support other key functions of a pervasive computing framework [Assad et al. 2007]. It could be distributed, with one model stored on one machine, another model on a different machine and so on. Of course, the system had to be able to find all relevant models as needed. This version of Personis was also active in the sense that the model could drive actions in the outside world. So, for example, it could start a music system playing and control it.

Our mobile phone version of Personis [Gerber et al. 2010] is, conceptually, a cut-down version of the main Personis framework. (It is a reimplementation in Java.) In terms of the core user modelling, this is partly an interim technology as mobile phones will soon have the power and flexibility to run the full Personis. In addition, it supports the dynamic download of arbitrary personalised applications. It only allows an application to access the model via a trusted “stub”. (This could be managed following an app-store model.) Then the mobile user can download untrusted applications that operate in conjunction with the appropriate stub. The framework's security mechanism runs the untrusted app in a sandbox, so that it cannot access either the user model or the network. The trusted stub acts as a mediator between the untrusted application and the user model.

User model representation, inference and interfaces
This section outlines the key ideas in the tightly linked technical and interface issues. We draw heavily on the systems from our earlier case studies. (These are italicised in Table VI). Taking the Simplicity Principle from Table IV, we began by designing and building systems to be as simple as possible, but still useful for personalisation. We then evaluated these, both from a technical viewpoint and to assess effectiveness of the interfaces for scrutability. This meant testing whether people could actually scrutinise the system to answer questions like those in Section 2. We typically needed several iterative cycles of refinement, both at the technical and the interface level. We now discuss the ways that we have tackled the representation and user interface challenges of scrutatable personalisation, summarised in Table VI. Although the table is
in chronological order, this discussion is organised in themes and in the discussion of interface issues, we refer to the principles in Table V.

The goal of a simple, explainable user model representation led to an analysis of the forms of evidence [Kay 1995] and design of the Accretion-Resolution representation. In the context of the sam text editor coaching system, we created long term models of users and interfaces that enabled users to see these and add evidence about their self-perception of their knowledge the Give value principle in Table V. The interface had an explanation subsystem to provide personalised explanations of the concepts in the model (the Ontology principle in Table V. This work also explored the notion of individual usability, discussed in Section 4 as a role for user models.

The next major step was the SASY I [Czarkowski and Kay 2000], which enabled the user to scrutinise an adaptive hypertext, as well as the model driving its personalisation (Table V – Use principle). This had subtle links to explanations, and people did not notice them. As we have described, SASY II [Czarkowski 2006] made the personalisation much more obvious. Usability studies as well as the authentic field trial indicated that it was quite successful in enabling people to determine what had been personalisation, what had been omitted from the personalised view, how the user model affected this and how to alter the model. It did, however, have only a simple form of personalisation with parts of the page selectively presented, under the control of user model based rules.

We now consider the main stages in the Personis user modelling framework. Like the um toolkit, this uses the accretion-resolution representation. The personis server [Kay et al. 2002] introduced representations of partial user models and personas. These were subsets of the model for use by particular applications. As indicated in Table VI, there was no user interface for these. This was partly addressed in first exploration of mobile user models on phone [Brar 2004; Hitchens et al. 2005]. This work tackled one of the on-going challenges of interoperability for user modelling, the definition of ontologies that can be understood by multiple applications [Carmagnola et al. 2011]. It also provided a phone interface for defining personas and small scale usability studies indicated people could control this to allow selective sharing of their user model [Brar 2004] (Outflow principle in Table V). The Ubicomp generalisation taking the same accretion resolution representation to modelling people, places and devices, was designed for scrutability and was used for the first versions of the Locator system described in the last case study. It had a limited scrutiny interface, with a link to the recent evidence used to infer a person’s location. This was addressed with PersonisAD when a carefully crafted control interface was created [Carmichael 2009]. User studies indicated that people were able to organise their friends into groups, and then control the choice of evidence filter and resolver for each group (Outflow and Outflow processes in Table V). The last Personis version in Table VI incorporated sophisticated ontological reasoning in the resolvers. As we showed in the Locator case study, this provided a scrutiny interface that enabled people to see why people were presented (or not) and to see explanations of the personalisation of location names.

We now turn to VIUM [Uther 2002; Uther and Kay 2003] a viewer for very large user models. This was designed to enable people to gain an overview of a large user model, with hundreds of elements modelled, (the Overview principle in Table V). Its successor, SIV [Lum 2007] used a light-weight ontology that was automatically generated by analysing online dictionaries [Apted et al. 2003]. As already discussed, this approach was chosen to exploit the dictionary definitions to explain concepts in the user model (the Ontology principle in Table V). It was used in a programming teaching environment [Kay et al. 2007] and in an HCI course [Lum 2007].

We have already introduced the remaining parts of Table VI. VCM [Cimolino et al. 2004] was discussed as an example of the Inflow principle in Table V. It elicited in-
formulation from the user, as many applications do. However, unlike most such applications, it showed the user what it intended to infer about them. We discussed JITT in the second case study. It supported scrutiny of the user model and control over the resolver and teaching strategy, supporting the Outflow processes principle in Table V. We used the Narcissus system in Section 4 to illustrate the role of a user model for navigation. It does this by providing a form of overview of the user model (Table V – Overview principle).

Overall, it is clear that Table VI has several remaining blanks in the interface column indicating much still to be done. At the same time, the systems described illustrate forms of each of the interface principles for scrutably user modelling and personalisation summarised in Table V.

8. REFLECTIONS AND RE-VISIONS

A core goal of this paper has been to share our vision for scrutability in personalisation and the benefits it has the potential to offer. We acknowledge that these benefits need to justify the effort and cost of creating personalised systems in a manner that supports scrutability. However, that cost can be reduced if we can build our understanding of the task, and establish a set of tools for creating scrutably personalised systems. That has been a key goal of much of our work summarised in this paper.

We began this paper by identifying key problems of personalisation that scrutably user modelling can help address. These related to: privacy, invisibility, errors in user models, wasted user models and lack of user control. To explain our vision, we presented two scenarios, one in a learning context and the other for personalised news. We identified several key classes of questions that people may ask about personalised systems and their user models. We then presented three detailed case studies of our work. The first, SASY, enabled people to determine how a web page had been adapted to them and to scrutinise and alter the user model. The second, JITT, enabled people to explore their user model, down to the detailed evidence and to control the choice of resolver that interprets the evidence as well as the teaching strategy controlling the personalisation. Locator, the third case study, made use of pervasive computing sensors and a rich ontological reasoning mechanism to personalise a location information display. It made use of the unified representation to model the sensors, places and users. These case studies were presented to provide concrete example of elements of our vision. Section 4 then moved to the broader roles that scrutably user models can play, as further motivation for the potential benefits of tackling the challenges involved in supporting scrutability.

The next two sections drew upon our own work and a broad range of other user modelling research to present design principles and a high level, evidence-based architecture. Researchers and systems builders can apply these, or at least consider them, when they design user models. We summarised a selection of our own systems, in terms of the research towards creating systems, representation and inference mechanisms and the critical interfaces for scrutably user models. We now can identify key challenges yet to be met.

Personal digital ecosystems, ‘client-side’ user model clouds

We are seeing a fast rise in the numbers of devices in people’s personal digital ecosystems. For example, a recent report [CISCO 2011] points to dramatic rise in ‘tablets, mobile phones, connected appliances and other smart machines’ making for a doubling of networked devices from 2010 to 2015, with many of these wirelessly connected. This will enable new forms of personalisation. It will also mean that parts of the user model will need to be stored on various devices as well as on servers that are accessible from a person’s devices. Conceptually, these user models will reside in a ‘personal user model...
cloud’. If people are to become masters of their own user models, these will need to become first class citizens, with scrutiny and control interfaces. This matches our extended view of ‘client-side’ personalisation to protect privacy based on the architecture described in Section 7. It will be challenging to create effective interfaces for people to control such user models.

**Sense making, sharing and scrutability**

When the user needs to make sense of their own user model, they may need comparable information for other people. For example, in our models of learner’s [Uther and Kay 2003; Kay and Lum 2005] one valuable form of the model showed the learner’s progress compared against the class as a whole, or against the top group in the class. In more general contexts, such as personal informatics [Li et al. 2010], this may mean that aggregated stores are needed. A reciprocity principle often applies in such cases, where the individual is allowed to see comparative information if they are willing to release their own data for aggregation. This creates the need for effective scrutiny and control of the parts of the user model to be shared. It also may call for mechanisms to blur the information released [Berkovsky et al. 2007].

**Scrutability toolkits for application builders, making for familiarity and pervasiveness of scrutiny mechanisms**

One of the challenges we faced in creating scrutability interfaces is that users did not expect to be able to scrutinise the personalisation [Czarkowski and Kay 2006; Czarkowski 2006]. In fact, people were so used to errors in personalisation that they were unsurprised by them. They simply accepted them. It may be too much to hope for such expectations to change in the near future. Beyond that, it will be far easier for people to scrutinise their user models and personalisation if scrutable user modelling becomes more common and if consistent in scrutiny interface elements emerge. Toolkits for scrutable personalisation will help here and also ease the burden on programmers to support scrutability.

**The increasingly pressing need for scrutability**

Personalisation is likely to play an increasing role in people’s lives. There is already widespread personalisation on commercial web sites and this is likely to spread both in that context and in emerging forms on various devices. We are beginning to see explanations in widespread web interfaces when the user is asked for personal information. Typically this is a link next a particular field, so that the user can see why this information is needed. That information often includes a link to the privacy policy for the web site. Our vision is for a deeper form of support for scrutability to become the norm. This is important on many levels, from the most basic requirement of ensuring that a person has control over their personal information and for privacy as well as the many potential benefits that a scrutable user model can offer for learning and navigating our increasingly complex information spaces.

**Acknowledgements.** Work reported in this paper was supported by the Smart Internet Technology Cooperative Research Centre, Smart Services Cooperative Research Centre and the Australian Research Council Discovery Awards. We thank Andrew Claypham for reviewing the final draft.

**REFERENCES**


ACM Transactions on Interactive Intelligent Systems, Vol. X, No. 4, Article XXX, Publication date: July 2011.
Creating personalisation that people can scrutinise


Creating personalisation that people can scrutinise


SHILTON, K. 2009a. Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection. Queue 7, 7, 40–47.

SHILTON, K. 2009b. Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection. Queue 7, 7, 40–47.


