Preprint: Final version of paper is available from

CONNELL, I., GREEN, T. & BLANDFORD, A. (2003) Ontological Sketch Models: Highlighting User-System Misfits. In E. O'Neill, P. Palanque & P. Johnson (Eds.) *People and Computers XVII, Proc. HCI'03*. 163-178. Springer.

Ontological Sketch Models: Highlighting User-System Misfits

Iain Connell[1], Thomas Green[2] & Ann Blandford[1]

[1] *UCL Interaction Centre (UCLIC), University College London, 26 Bedford Way, London WC1H OAP, U.K.*
{i.connell, a.blandford}@ucl.ac.uk
[2] *Oriel House, 27 Allerton Park, Leeds LS7 4ND, UK.*
greenery@ntlworld.com

Ontological Sketch Modelling (OSM) is a novel approach to usability evaluation that concentrates on both the user's conceptual model of the domain and 'working practices', and the conceptual model built into a device or a work-system. Analysing the degree of fit between these models can reveal potential problems in learning and use that are not revealed by existing HCI approaches. We show how OSM can identify such potential misfits between user and system. We also describe how an OSM analysis can be edited, conventionalised and viewed in tabular form, thereby allowing automatic highlighting of user-system misfits. Illustrative examples are a typical drawing application and a digital music library.

Keywords: ontological sketch modelling, usability evaluation, conceptual models, misfits, drawing application, digital music library.

1 Background to OSM

Ontological Sketch Modelling (OSM) is a novel approach to usability evaluation that concentrates on both the user's conceptual model of the domain and 'working practices' (that is: how the device fits in with the way the user works), and the conceptual model built into a device or a work-system. Analysing the degree of fit

between these models can reveal potential problems in learning and use that are not revealed by existing HCI approaches. Essentially, OSM aims not just to say whether a device might be easy to use, but whether the device does something that makes sense to the user. OSM is also exceptional among HCI evaluation methods in that the same approach can apply either to individual devices or to a whole work system: our investigations have ranged from digital watches to the London Ambulance Service dispatch system [Blandford et al. 2002].

Devices and systems can exhibit substantial mismatches between the user's preferred conceptualisation and that which is imposed. For example, users of a video link may be compelled by the system to define the Quality of Service in terms of bandwidth, while they think of it in terms of frame rate or image resolution. Similarly, the user of a drawing program (see below) may think in terms of representations of domain objects (e.g. "this part of the drawing shows the whale's tail") while the program supports only the manipulation of device objects (e.g. curves, text boxes). We call these mismatches 'misfits'.

Central to OSM's approach is the simultaneous representation, *in the same language*, of the user's conceptualisation of the domain and the conceptualisation that is built into the system or device. (Note that for brevity we may find ourselves referring to either 'the system' or 'the device'; in each case, we mean the whole gamut.) Misfits cannot be revealed by any approach to HCI that focuses solely on either the user or the device: both must be considered. Traditional task-centered user-modelling for HCI has had some very effective results, but in our view it cannot reveal misfits because it does not explicitly consider how the user's conceptual model relates to the model imposed by the device.

Application of OSM to several full-scale examples, two of which are described below, has shown that effective and useful insights can be readily obtained. We have also demonstrated that non-experts can be quickly taught the technique to a standard that is reasonably effective [Blandford & Green 1997].

In an OSM analysis the analyst describes first the visible entities and attributes that are linked within the device or system; then the entities contained in the user's conceptual model; and lastly those entities embodied within the device that are not part of the user's conceptualisation, but which users need to be aware of in order to use the device effectively. The resulting entities may be private to the *device* (the user cannot alter them), or they may be private to the *user* (the device does not represent them), or they may be *shared* (accessible to both the device and the user). All communication between the two worlds of device and user takes place through the shared entities. If the user-private entities do not fit well onto the shared entities, the device will have usability problems. If the device-private entities are difficult to discover or understand, the user is likely to have difficulty learning to work with them.

The OSM notation is informal and intentionally sketchy. However, we have developed an OSM editor (illustrated below) which enforces a consistent format and terminology. The editor output data can be displayed in whatever manner is convenient; at present we use a stylised tabular format that is simple but revealing. In this paper we show how such a format may be used to reveal user-system misfits. A certain amount of algorithmic analysis is also possible, but we

anticipate that the main effectiveness of the approach will lie in the insights gained by codifying and reflecting upon conceptual models and device features.

OSM's emphasis on conceptual modelling has roots in such work as Payne's 'task-entity analysis' [Payne 1993], which demonstrated how users' conceptual models of the time-management domain were ill-served by the electronic diaries then available; and in Moran's ETIT analysis [Moran 1983], which mapped 'external' domain tasks onto 'internal' device tasks. These studies, though ground-breaking, demonstrated only what could be achieved when the analyst was an expert in applied cognitive science. To be practically useful to non-experts, a methodology needed to be developed. Attempts at methodologies which have influenced OSM include the framework of 'cognitive dimensions' (CDs) developed by Green [1989] and others [Blackwell et al. 2001], some of which can be interpreted as misfits; 'Entity-Relationship Models for Information Artefacts' (ERMIA) developed by Green & Benyon [1996]; and 'Programmable User Models' (PUMs) [Young et al. 1989; Blandford & Young 1996].

Broadly speaking we would argue that none of these methodologies achieved the right blend. CDs have been taken up by some practitioners but for some purposes are under-defined, while ERMIA and PUMs are too detailed and too difficult. In contrast, we claim that OSM is novel in that encourages its users to scrutinise and reflect on all parts of the system without forcing them to employ detailed and time-consuming new notations.

2 The Structure of an OSM

An OSM is not a formal model, in the fullest sense of that word, but it is highly conventionalised in structure. The central feature of the model is a list of *entities*. For each entity there is a statement of how it can be created and deleted (if possible), and—more importantly—whether or not the entity is explicitly known to the user, whether or not it is represented at the user-system interface, and whether or not it is represented within the system. These elaborations are illustrated below.

Each entity potentially has a set of *attributes*. For each attribute there is a similar statement of how its value can be set and changed, how far it is known to the user, and how far it is represented at the interface and in the system.

The methods for creating and deleting entities, and for setting and changing attributes, are *actions*. Each action can be moded and/or disguised (i.e. available but hard to find).

The model also contains *relationships* of several types. First, an entity can *consist-of* other entities: this is a term that covers many different types of possible relationship, which it would be counter-productive to distinguish in detail. There are two forms of 'constraint' relationship: one attribute's value can constrain another either as a *device-constraint*, present whether the user likes it or not, or as a *goal-constraint*, a state of affairs that is important to the user but is not necessarily imposed by the device. For example, it is a device constraint that all of a drawing must lie within the bounds of the drawing space; it is a potential goal constraint that certain parts of a drawing should be aligned neatly (if the constraint is violated the resulting drawing might not make proper sense, but the device will permit it). Lastly, it is necessary to note that changing one attribute may change another—e.g.

changing page margins in a word processor may change the number of pages needed for a given document.

The examples illustrated below feature entities and attributes rather than actions. Unabridged versions of the analyses from which they are derived may be obtained from the OSM web site [OSM 2003].

An OSM model can be displayed in any convenient form. In fact, we are still searching for the best visualisation technique for such a structure. At present we export the model into XML format and translate that into a stylised hypertext table. As will be demonstrated, this allows for the tabulation of the model in concise form; it also provides opportunity for the automatic highlighting of likely user-system misfits.

3 Procedure

In its current form an OSM analysis consists of the following stages.

i. The analyst gets to know the system and the users and constructs a paper-based analysis. This sets out in tabular form the main device and user entities and attributes, plus actions and relationships. A table of user-interface-system dependencies is also constructed.
ii. The analysis is then conventionalised as an OSM model, preferably by setting it out in our OSMosis editor (available from the OSM web site: OSM [2003]). The conventionalised model can be displayed in tabular format, preferably in hypertext, and if necessary can be revised. User-system misfits are highlighted automatically.
iii. The conventionalised model can also be converted to Prolog assertions to allow algorithmic analysis of further types of misfits. (This will not be illustrated here.)

3.1 Initial Analysis

The first stage of the initial (paper-based) analysis is to identify in tabular form the core entities, attributes, actions and relationships. We group entities and attributes into device and user. Actions are typically described in terms of interaction processes. Some relationships may arise immediately out of the initial device-user considerations (our first example, the drawing application, being a case in point).

In the second stage, for each entity and attribute so far described, we delineate the differences between the device/domain (system) and user concepts, and the ways in which these are mediated via the interface.

At these initial stages the analyst must rely on a certain amount of craft knowledge, depending on familiarity with the device and/or domain. For a simple device such as a drawing application the main device entities and attributes may be revealed by inspection and exploration alone. More complex systems such as digital libraries may require input from designers and domain specialists, as well as resort to manuals and help files. In order to elicit users' conceptual entities, techniques such as structured interviewing or verbal protocol analysis can be used, perhaps allied to observation. The second analysis illustrated in this paper made

use of interviews with musicians (see also Blandford et al. [2002] for an account of the interviewing and inquiry methods used in a study of dispatch methods at London Ambulance Service).

The first stage is heavily iterative, requiring frequent re-visiting of concepts and relationships. It is also likely that return will be made to these putative results after the conventionalising stage, for example in order to enforce consistency with the edited and tabulated version.

3.2 Conventionalising and Viewing the Model

The paper-based analysis is likely to include too much detail in some parts, along with gaps in others. It is also likely to contain inconsistencies and omissions. By conventionalising the model many of these problems can be eliminated, at the inevitable cost of omitting much free-form detail. Our experience indicates that a strict method of conventionalising is essential; otherwise, despite our best intentions, we have found ourselves cheating. Thus we transcribe the initial analysis into our OSM editor, OSMosis. We then export the result in XML format and translate it into HTML using an XSLT processor. The resulting structure can be viewed in various formats, among them the stylised tables illustrated below.

Since the quantity of the input to the conventionalised model is not restricted, the amount of detail to be included depends partly on the purposes for which the model is being made. However, a tool like OSMosis does confine the analyst to a certain degree of formalism. The examples below show how fairly extensive paper analyses (six pages, for even the drawing application) can be reduced to a more manageable quantity and viewed in shorthand format.

Inspection of the model can highlight the potential user-system misfits which have been revealed by the initial analysis, particularly the delineation of the device/domain, interface and user dependencies. This in turn can feed back into the paper analysis and the initial model. The misfit-highlighting process has been automated and is illustrated for the examples below.

3.3 Algorithmic Analysis

Finally, algorithmic analysis can reveal further misfits. These have mostly been derived from Green's cognitive dimensions framework [Green 1989]. For example, in that framework *viscosity* denotes resistance to change: a change that the user conceptualises as a single operation turns out to require multiple actions (*repetition* viscosity); alternatively, an action may turn out to require several cleaning-up operations to restore internal consistency (*knock-on* viscosity).

It is relatively easy to convert the conventionalised OSM model into Prolog assertions and scrutinise the model for examples of viscosity and other such misfits. Details will not be given here because to date this has formed only a minor part of our analyses.

4. Example Analyses

As illustrations of the OSM analysis process, we present two representative examples. The first, a drawing application, will be familiar to most readers. In this application most of the key activities take place in the system; the challenges for the user are to understand the tools that are available and how to manipulate the display objects. The second example is an internet-based digital library. This is an application where the user's understanding and requirements reside largely 'in the head', and need to be expressed in a form suitable for accessing system information. As noted above, we have completed analyses of both smaller and larger systems; for example, Blandford et al. [2002] report on the application of OSM to a multi-person work system.

For reasons of space, we focus on entities, attributes and relationships and omit the details of the OSM editor (as well as the Notes transferred from the paper tables). The complete analyses are available from the OSM web site [OSM 2003].

4.1 Drawing Application

Common features of typical desktop drawing applications are a drawing space, a set of drawing tools (e.g. palette, toolbar(s), panels), and the means to change and combine created drawing objects. In OSM terms, the drawing objects are entities whose attributes can be manipulated and changed using the appropriate tools.

4.1.1 Initial Analysis

The first main device entity is the Drawing Object out of which drawings are realised. This is a *shared* entity, meaning that it is part of both the user's conceptualisation and the device's representation. See Figure 1. The object has several attributes, the first of which is its configuration or object type (straight line, rectangle, oval, polygon, free line). Drawing objects are created by selecting the appropriate tool from the palette or toolbar (etc.) and inserting the object into the drawing space; once created, an object can be changed by manipulating its size, shape, fill colour (etc.) attributes. Some of these changes can be made 'directly' (e.g. by dragging the object's grab handles), some 'indirectly' (e.g. via dialogue boxes initiated from menu options), and some only by means of certain palette or toolbar selections (e.g. by switching between 'drawing' and 'text' cursors).

In relation to the latter, we find that there are device constraints (in the OSM editor jargon, 'constrained_by_device' relationships) between the currently selected tool and the drawing object: not only does the tool dictate the drawing object to be created, but certain attributes of a drawing object can be changed only by particular tools. See Figure 2. In addition, the main user entity, a Drawing, can comprise only the drawing objects which the palette allows us to create (though drawing objects can be grouped together). We denote this as a 'consists_of' relationship (Figure 2).

Entity	Type	Create by	Delete by	Notes
Drawing Object	Shared	Select palette tool and insert in drawing space	Select object and delete	To create an object the appropriate tool must first be selected from the palette

Attributes	Instances	Set by	Change by
Configuration (object type)	Straight line, Rectangle, Oval, Polygon, Free line	Select tool from palette	Select a different palette tool
Size		Select tool	Select grab handles and drag
Shape	As per object type	Select tool	Cannot generally be changed
Fill colour	As shown on colour palette	Select from colour palette	Re-select
Orientation	Horizontal, vertical	Object defaults	Use 'rotate' tool
Grouped	Grouped, not grouped	Select several objects & group	Ungroup

Entity	Type	Create by	Delete by	Notes
Drawing	User	Add, manipulate and combine drawing objects	Select and delete	A drawing consists of drawing objects which can be combined together

Attributes	Instances	Set by	Change by
Size		Within drawing space	Expand and reduce
Shape		Combine drawing objects	Re-combine drawing objects
Orientation	Horizontal, vertical	As for drawing objects	

Figure 1. A main device entity (Drawing Object) and a main user entity (Drawing) for a drawing application. See text for explanation.

Relationship	Actor	Type	Acted On	Notes
Define_object	Palette tool	Constrained_ by_device	Drawing Object	The drawing object to be created depends on the palette tool which is first selected
Define_actions	Drawing Object	Constrained_ by_device	Palette tool	Some actions which can be performed on a drawing object are defined by the object type
Consists_of	Drawing	Consists_of	Drawing objects	

Figure 2. Two device-constraint relationships and a consists-of relationship for a drawing application. See text for explanation.

For each of the entities and attributes so far described we now set out the main components of the dependencies between the user, interface and system. See Figure 3.

In the current OSM configuration, we judge the user's conceptualisation of each component (entity or attribute) as being Explicit, Implicit, or Absent, and the system's as either Present or Absent. If a component is explicit or implicit to the user but absent from the system, we record that as a definite user-system misfit. Similarly, if a component is present in the system (domain or device) but absent from the user's model, that also represents a misfit.

At the interface, an entity or attribute is judged to be one of the following: Absent, Direct, Disguised, Delayed, Hidden or Undiscoverable. By these terms, we mean:

- Absent: not represented;
- Direct; represented and easily interpreted by the user;
- Disguised: represented, but hard to interpret;
- Delayed: represented, but not available to the user until some time later in the interaction;
- Hidden: represented, but the user has to perform an explicit action to reveal the state of the entity or attribute; and
- Undiscoverable: represented only to the user who has good system knowledge, but unlikely to be discovered by most users.

In general, if an interface component is disguised or delayed, this may represent a misfit, depending on the other two components; if, however, it is deemed to be hidden or undiscoverable, we record that as a misfit independently of the other components.

In this example, Figure 3 shows that both drawings and drawing objects have been judged to be directly represented at the interface and part of the user's explicit or implicit conceptual model. However, some drawing object attributes are deemed to be either disguised or hidden at the interface, and the user's drawing itself is absent from the system model. In particular, the 'Grouped' attribute is shown as both absent from the user's conceptual model and hidden at the interface. This will be highlighted as a serious misfit when the OSM model is conventionalised. See the next Section.

Entities & Attributes	User	Interface	System
Drawing	Explicit	Direct	Absent
Orientation	Explicit	Direct	Present
Shape	Explicit	Direct	Present
Size	Implicit	Direct	Present
Drawing object	Implicit	Direct	Present
Configuration	Implicit	Disguised	Present
Fill colour	Explicit	Direct	Present
Grouped	Absent	Hidden	Present
Orientation	Explicit	Direct	Present
Shape	Explicit	Direct	Present
Size	Explicit	Direct	Present

Figure 3. Part of a User-Interface-System table (showing entities Drawing and Drawing Object) for a drawing application. See text for explanation.

4.1.2 Conventionalising and Viewing the Model

We now transcribe the essential elements of the paper-based analysis into the OSM editor, OSMosis. Figure 4 shows sample editor content, for the entity 'Drawing_object' and the attribute 'Grouped' (featured in Figures 1 and 3). The leftmost dialogue box shows the full entity-action-attribute-relationship listing; the

middle dialogue box the six attributes of the highlighted entity (illustrated in Figures 1 and 3); and the rightmost box the system-interface-user dependencies, plus 'set' and 'change by' actions, for the highlighted attribute.

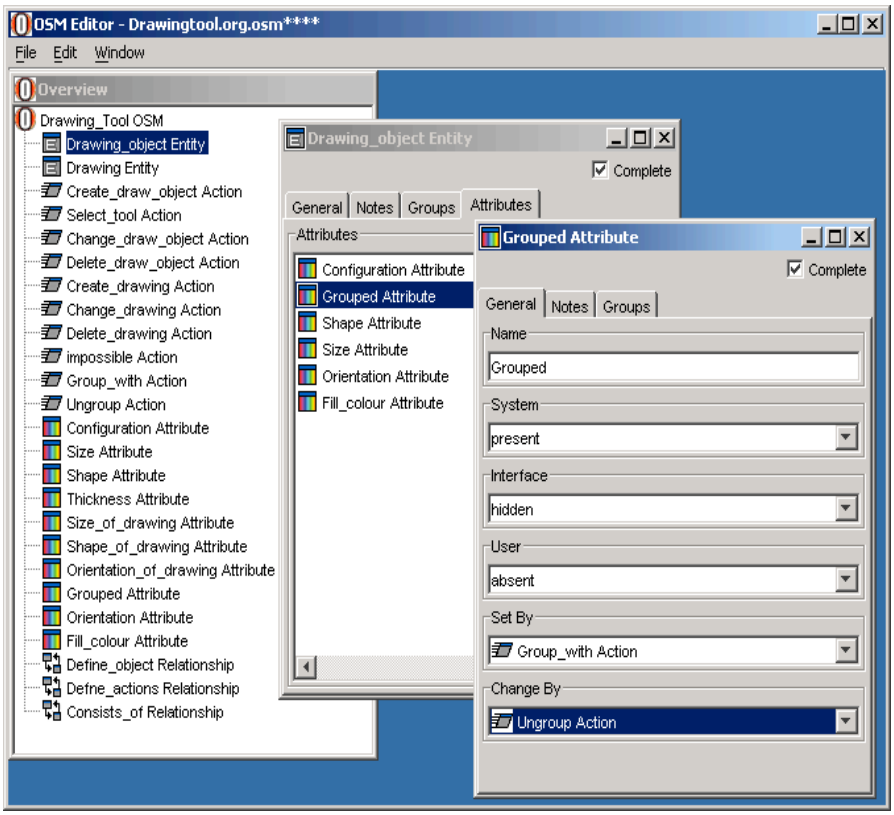
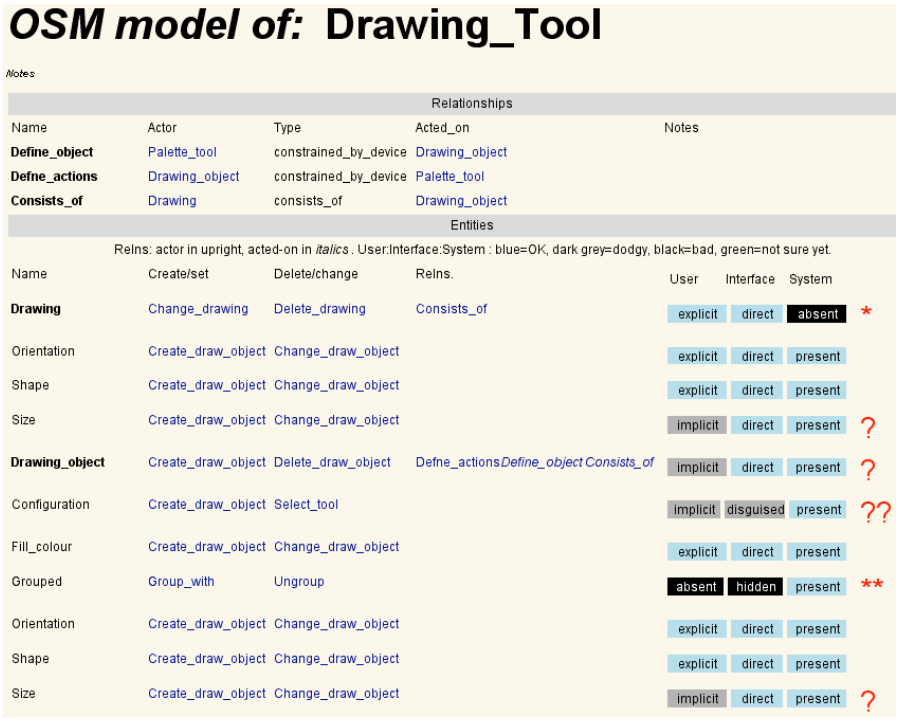


Figure 4. OSMosis editor with sample content for a drawing application. See text for explanation.

Once thus transcribed we can export the model in XML and view it in a stylised hypertext table, as partially illustrated (minus the Actions and without any transcribed Notes) in Figure 5. Misfits are flagged against the 'User Interface System' table just described. In this example the only serious misfit (denoted **) relates to the attribute 'Grouped', others being recorded as either less serious (denoted *) or merely potential misfits (denoted ?? or ?). (The user-interface-system table is also colour-coded to enhance visual interpretation.)



OSM model of: Drawing_Tool

Notes

Relationships

Name	Actor	Type	Acted_on	Notes
Define_object	Palette_tool	constrained_by_device	Drawing_object	
Defne_actions	Drawing_object	constrained_by_device	Palette_tool	
Consists_of	Drawing	consists_of	Drawing_object	

Entities

RelIns: actor in upright, acted-on in *italics* . User:Interface:System : blue=OK, dark grey=dodgy, black=bad, green=not sure yet.

Name	Create/set	Delete/change	RelIns.	User	Interface	System	
Drawing	Change_drawing	Delete_drawing	Consists_of	explicit	direct	absent	*
Orientation	Create_draw_object	Change_draw_object		explicit	direct	present	
Shape	Create_draw_object	Change_draw_object		explicit	direct	present	
Size	Create_draw_object	Change_draw_object		implicit	direct	present	?
Drawing_object	Create_draw_object	Delete_draw_object	Defne_actions*Define_object Consists_of*	implicit	direct	present	?
Configuration	Create_draw_object	Select_tool		implicit	disguised	present	??
Fill_colour	Create_draw_object	Change_draw_object		explicit	direct	present	
Grouped	Group_with	Ungroup		absent	hidden	present	**
Orientation	Create_draw_object	Change_draw_object		explicit	direct	present	
Shape	Create_draw_object	Change_draw_object		explicit	direct	present	
Size	Create_draw_object	Change_draw_object		implicit	direct	present	?

Figure 5. Part of a stylised hypertext table (tabulated model) for a drawing application.

In this first example analysis, then, the main source of user-interface-system misfits has been attributed to the inferred absence of a system concept for a drawing, the strong relationship between drawing objects and palette tools, and the absent-hidden-present nature of drawing object grouping.

The second example which follows illustrates how a different set of misfits may be focused more on the user's expectations of how system information should be organised, in this case in an online digital library.

4.2 New Zealand Digital Music Library

The New Zealand Digital Music Library [NZDL 2003] is part of the New Zealand Digital Library (NZDL). It holds a large number of digitised melodies which can be retrieved over the internet and played back on the client computer. Melody files may be browsed or searched. Searching may be by title ('text query') or by tune matching ('music query'), the latter including sung or played inputs. We summarise the results of an OSM analysis (full version available from the OSM web site—OSM [2003]) carried out on the Music Library in the form in which it existed between October 2001 and January 2002. (Since then, partly as a result of this and related analyses, e.g. Blandford & Stelmaszewska [2002], the interface has been revised.)

4.2.1 Initial Analysis

The NZDL Music Library consists of several sub-collections, each of which contains indexed melody files. A Melody File is thus the main device entity, whose attributes include file name and file size. Each sub-collection's melody files have particular component types and playback formats. Neither a Melody File nor its attributes may created, deleted or changed by end users. See Figure 6.

Entity	Type	Create by	Delete by	Notes
Melody File	Device	Impossible for end user. (NZDL: retrieve from external source or create from score)	Impossible for end user. (NZDL: Remove from sub-collection)	Sub-collections consist of melody files held in different formats.

Attributes	Instances	Set by	Change by
How indexed	Title, alphanumeric	Impossible for end user. (Done by NZDL)	Impossible for end user. (Done by NZDL)
Name	Many and various		
No. of components	3 to 5		
Playback format	MIDI0, MIDI1, AIFF, etc.		
Size	Various		
Type of components	Melody, score, title, lyrics, etc.		

Figure 6. NZDL Music Library: entities and attributes for the device entity Melody File. See text for explanation.

In order to retrieve a melody file from a sub-collection by tune matching, the user must input (that is, sing or play) the tune into the client computer. That version of the melody is then matched against the stored data using a proprietary conversion system named Meldex [Bainbridge 2000]. Successful tune matching thus requires that the stored version of the tune be compatible with the converted version, which in turn requires that the sung or played input be in suitable form.

However, musicians' conceptual models of both tunes and melody file searching may be more complex. Figure 7 shows some potential descriptors (attributes) of a 'tune' derived from interviews carried out with musicians of varying IT and music information retrieval experience. Besides the melody itself, these include pitch, tempo, rhythm and tonality. (The same interviewees also offered a variety of additional criteria on which they might expect to perform 'text' searches, such as composer, year, period, artist.) Thus we judged there to be some potential misfits between the tune-matching requirements of the Music Library and those of its likely users. These are recorded in the User-Interface-System table using the same conventions as described above (Figure 3). When conventionalised, they are highlighted (flagged) in the stylised hypertext table, as illustrated in the next Section.

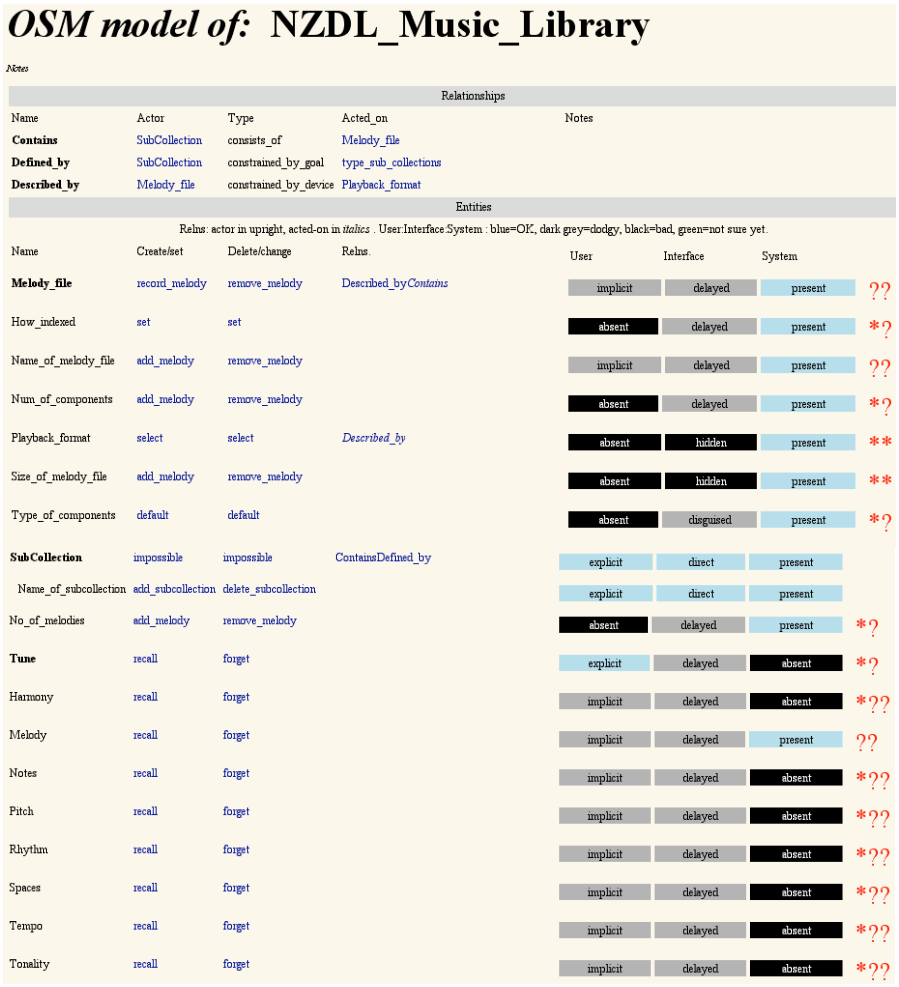
Entity	Type	Create by	Delete by	Notes
Tune	User	Recall	Forget	In order to make a match with a sub-collection melody file the user must first recall the tune

Attributes	Instances	Set by	Change by
Melody, pitch, notes, spaces, tempo, harmony, rhythm, tonality	(Requires specialised musical knowledge)	Play, sing or otherwise transcribe a recalled tune	Play, sing or transcribe differently

Figure 7. NZDL Music Library: Entities and descriptor attributes for the user entity Tune. See text for explanation.

4.2.2 Conventionalising and Viewing the Model

The result of conventionalising and viewing the Music Library tune matching model is partially illustrated (minus Actions and Notes) in Figure 8. In this more complex example we have identified, amongst others: two serious misfits (flagged **), namely those related to playback format and melody file size (both absent-hidden-present); system concepts deemed absent from the user's model (indexing, number, and type of melody file components), flagged *?; and implicit user concepts for a tune other than melody itself, such as harmony, notes, pitch, rhythm, which were judged to be absent from the system model of the tune matching process (flagged *??). (These may, however, be revealed in the scores of retrieved melodies once displayed and/or played back on the user's computer.)



***OSM model of:* NZDL_Music_Library**

Notes

Relationships

Name	Actor	Type	Acted_on	Notes
Contains	SubCollection	consists_of	Melody_file	
Defined_by	SubCollection	constrained_by_goal	type_sub_collections	
Described_by	Melody_file	constrained_by_device	Playback_format	

Entities

Relns: actor in upright, acted-on in *italics* . User:Interface:System : blue=OK, dark grey=dodgy, black=bad, green=not sure yet.

Name	Create/set	Delete/change	Relns.	User	Interface	System	
Melody_file	record_melody	remove_melody	Described_by*Contains*	implicit	delayed	present	??
How_indexed	set	set		absent	delayed	present	*?
Name_of_melody_file	add_melody	remove_melody		implicit	delayed	present	??
Num_of_components	add_melody	remove_melody		absent	delayed	present	*?
Playback_format	select	select	*Described_by*	absent	hidden	present	**
Size_of_melody_file	add_melody	remove_melody		absent	hidden	present	**
Type_of_components	default	default		absent	disguised	present	*?
SubCollection	impossible	impossible	ContainsDefined_by	explicit	direct	present	
Name_of_subcollection	add_subcollection	delete_subcollection		explicit	direct	present	
No_of_melodies	add_melody	remove_melody		absent	delayed	present	*?
Tune	recall	forget		explicit	delayed	absent	*?
Harmony	recall	forget		implicit	delayed	absent	*??
Melody	recall	forget		implicit	delayed	present	??
Notes	recall	forget		implicit	delayed	absent	*??
Pitch	recall	forget		implicit	delayed	absent	*??
Rhythm	recall	forget		implicit	delayed	absent	*??
Spaces	recall	forget		implicit	delayed	absent	*??
Tempo	recall	forget		implicit	delayed	absent	*??
Tonality	recall	forget		implicit	delayed	absent	*??

Figure 8. Part of a stylised hypertext table for the NZDL Music Library.

In this second example the main source of misfits was attributed to the contrast between system melody file concepts which may be absent from the user's conception and hidden at the interface (absent-hidden-present), and user concepts for a tune which may be both delayed at the interface (until retrieved and played back) and absent from the system model for tune matching (implicit-delayed-absent).

5 Conclusions

OSM presents an evaluation methodology which appears to guide non-expert analysts successfully towards an understanding of how far a device or a system matches or falls short of a typical user's conceptualisation of that domain. By its

deliberately sketchy nature, it avoids drowning the analyst in detail, but it can be made almost as detailed, or as broad-brush, as desired.

The scope of OSM analysis is restricted to conceptual misfits. It has little to say about other issues, such as the rendering of the system image, the choice of communicative language (cf. heuristic evaluation, Nielsen [1994]) or speed of operation (cf. the keystroke level model, Card et al. [1983]). Conversely, those methods may have little or nothing to say about conceptual misfits.

Inferences about misfits between the system and the user's conceptualisation may be revealed both by the tabular format and by algorithmic analysis. Experience to date (illustrated by the examples presented in this paper) suggests that much is revealed by the former process, for it encourages the analyst to question assumptions about the fit between system and user. The algorithmic analysis, in contrast, is only effective when a reasonable amount of detail has been made available.

At present, our main concern is with relationships. Typically, analysts can readily unearth entities: user entities may be uncovered through the knowledge elicitation techniques outlined above, while system entities are quickly found by inspection. Attributes also appear to be speedily unearthed. But relationships can be easily missed, for they are neither externalised on the interface nor likely to be immediately grasped by the analyst. Unfortunately, it is often the degree to which a system helps its users to meet goal constraints that determines how useful it is. Much, therefore, hangs on the analyst's success during the first stage, that of constructing the paper-based model and eliciting relevant knowledge.

How does OSM relate to other approaches? Of the non-modelling approaches to usability evaluation, perhaps the nearest comparison is with Cognitive Walkthrough (CW) [Polson et al. 1992]. CW was intended to guide non-expert analysts towards a cognitive analysis by considering novice-user usability issues. Both CW and OSM require the analyst to make decisions about the potential user's knowledge, but their focus is different: CW considers each individual action step required to achieve a stated goal, but has nothing to offer about structural relationships. In our experience, CW does a good job of highlighting potential difficulties in learning to use a device, but has almost nothing to say about whether that device fits the user's conceptualisation, and therefore, in our view, nothing to say about whether the device is useful. OSM is almost exactly complementary in its approach. Current work is exploring the ways in which OSM and CW differ in their potential to identify misfits associated with everyday systems such as ticket vending machines. It is hoped that this will also demonstrate measurable differences between the two methods. Other OSM analyses, for example of the NZDL Music Library summarised in Section 4.2.1, have generated recommendations for changes to existing systems.

After applying OSM to several full-scale systems, two of which are illustrated here, we are confident that the technique has real-life potential for the illumination of a range of applications including both analogue tools (e.g. a fob watch) and digitally-based applications (e.g. desktop tools, online diaries, ticket vending machines, digital libraries) as well as work practice studies (e.g. ambulance dispatch, health care IT support). OSM goes well beyond the confines of the laboratory. However, it does not purport to supplant the analyst's judgement: in

many cases, the analyst will have to decide whether or not an action is easy to find, and will always have to decide how much detail to include. It is hard to imagine that any approach could totally remove the need for judgement; however, we persist in the belief that even inexperienced analysts know what questions they are trying to answer.

Acknowledgements

We are grateful for the information about the NZDL Music Library provided by David Bainbridge of Waikato University. The OSM editor was developed by Owen Green. The work reported in this paper has been supported by EPSRC grant no. GR/R39108.

References

Bainbridge, D. [2000], "The Role of Music IR in the New Zealand Digital Library Project", in *International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, Mass., October 2000.

Blackwell, A.F., Britton, C., Cox, A., Green, T.R.G., Gurr, C.A., Kadoda, G.F., Kutar, M., Loomes, M., Nehaniv, C.L., Petre, M., Roast, C., Roes, C., Wong, A. & Young, R.M. [2001], "Cognitive Dimensions of Notations: Design Tools for Cognitive Technology", *in* M. Beynon, C.L. Nehaniv & K. Dautenhahn (eds.) *Cognitive Technology 2001* (LNAI 2117), pp.325-341. Springer-Verlag.

Blandford, A. E. & Green, T. R. G. [1997], "OSM: an Ontology-based Approach to Usability Evaluation", *in Proceedings of an International Workshop on Representations in Interactive Software Development*, Queen Mary and Westfield College, July 1997, pp.82-91.

Blandford, A. E. & Stelmaszewska, H. [2002], "Usability of Musical Digital Libraries: a Multimodal Analysis", *in* M. Fingerhut (ed.), *Proceedings of the 3rd. International Conference on Musical Information Retriveal, ISMIR 2002*, Paris, October 2002, pp.231-237.

Blandford, A. E. & Young, R. M. [1996], "Specifying User Knowledge for the Design of Interactive Systems", *Software Engineering Journal* **11**.6, 323-333.

Blandford, A. E., Wong, B. L. W., Connell, I. W. & Green, T. R. G. [2002], "Multiple Viewpoints on Computer Supported Team Work: a Case Study on Ambulance Dispatch", *in* X. Faulkner, J. Finlay & F.D. Étienne (eds.), *People and Computers XVI*, pp.139-156. Proceedings of HCI 2002, London, September 2002. Springer.

Card, S. K., Moran, T. P. & Newell, A. [1983], *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum.

Green, T. R. G. [1989], "Cognitive Dimensions of Notations", *in* A. Sutcliffe & L. Macaulay (eds.), *People and Computers V*, pp.443-460. Proceedings of HCI '92, York, September 1992. Cambridge University Press.

Green, T. R. G. & Benyon, D. [1996], "The skull Beneath the Skin: Entity-Relationship Models of Information Artifacts", *International Journal of Human-Computer Studies* **44**(6), 801-828.

Moran, T. P. [1983], "Getting Into a System: External-internal Task Mapping Analysis", *in* A.Janda (ed.), *Human Factors in Computing Systems*, pp.45-49. ACM SIGCHI and Human Factors Society conference proceedings, Boston, December 1983. New York: ACM Press.

NZDL (New Zealand Digital Library) Music Library [2003], last accessed May 2003, http://www.nzdl.org/fast-cgi-bin/music/musiclibrary

Nielsen, J. [1994], "Heuristic Evaluation", *in* J. Nielsen & R. Mack (eds.), *Usability Inspection Methods*, pp.25-62. New York: John Wiley.

OSM (Ontological Sketch Modelling) [2003], last accessed May 2003, http://www.uclic.ucl.ac.uk/annb/OSM.html

Payne, S. J. [1993], "Understanding Calendar Use", *Human-Computer Interaction* **8**, 83-100.

Polson, P., Lewis, C., Rieman, J. & Wharton, C. [1992], "Cognitive Walkthroughs: a Method for Theory-based Evaluation of User Interfaces", *International Journal of Man-Machine Studies* **36**, 741–773.

Young, R. M., Green, T. R. G. & Simon, T. [1989], "Programmable User Models for Predictive Evaluation of Interface Designs", *in* K. Bice. & C. Lewis (eds.), *Wings for the Mind: CHI '89 Conference Proceedings*, pp.15-19. ACM conference on human factors in computing systems, Austin, Texas, April-May 1989. Reading, MA: Addison-Wesley.