Ontology-Driven Automated Generation of Data Entry Interfaces to Databases

Alan Cannon[1], Jessie B. Kennedy[1] Trevor Paterson[1], Mark F. Watson[2]

[1]School of Computing, Napier University, Edinburgh, EH10 5DT, U.K.
{a.cannon, j.kennedy, t.paterson}@napier.ac.uk
[2]Royal Botanic Garden, Edinburgh, EH3 5LR, U.K.
m.watson@rbge.org.uk

Abstract. This paper discusses an ontology based approach for the automatic generation of data entry interfaces to databases. An existing domain ontology is mapped to a system domain model, which a domain expert can then specialise using their domain expertise, for their data entry needs as required for individual projects. Based on this specialised domain knowledge, the system automatically generates appropriate data entry interfaces with the aid of a presentation model. By constraining the data entered to a term definition ontology and utilising appropriate defined domain terminology the quality of the collected data can be improved. Compared with traditional model-based user automatic interface development environments, this approach also has the potential to reduce the labour requirements for the expert developer.

1. Introduction

Designing data entry interfaces which allow the capture of high quality data to databases without overburdening users remains a significant challenge in database and user interface research. It is common to present forms-based user interfaces to allow data entry to the database. These forms, whilst often automatically generated, are generally simplistic, being designed to conform to the structure of database tables (or views) and can only constrain data entered to conform to the system data type associated with the table attributes. In most databases many attributes are stored as character strings, for which it is difficult to ensure consistent use or data quality, especially in terms of their semantics related to the domain of the attribute. In order for data to be meaningful in the long term and to allow data integration across databases, it is important that the semantics of the data are captured along with the actual data. However, achieving this without placing undue requirements on users has proven difficult. In this paper we present a semi-automatic data entry interface generation tool to help improve the quality of data entry to databases. The system generates an interface that reflects the semantics of the data as captured in a domain ontology.

A domain in which the problem of capturing semantically well-defined data is important is that of biological taxonomy, the branch of biology concerned with the classification of organisms into an ordered hierarchical system of groups reflecting their natural relationships and similarities. In the Prometheus projects we are investigating

tools and techniques to aid plant taxonomists to capture and interact with their data. In particular we have developed database models for storing multiple classifications [1, 2] and data visualisation tools for exploring multiple overlapping classifications [3]. Currently we are developing tools for allowing plant taxonomists to describe the specimens used during the classification process. This classification process is based upon the identification and description of variation between different plant specimens. A key task for taxonomic projects therefore involves describing the characteristics of a number of specimens. Currently taxonomists capture these specimen descriptions using paper forms that are designed specifically for particular projects, to reflect the characteristics of importance for differentiating specimens in the plant group under study. These characteristics will vary between plant groups. Some electronic data formats have been developed for capturing taxonomic characteristic data [4,5,6], but do not address the semantic standardisation and quality of data stored [7].

During our research it became apparent that there was no agreed vocabulary used by taxonomists when describing their specimens. This meant that although specimen descriptions were comparable within one project, it was impossible to compare descriptions across projects undertaken by different taxonomists and possibly even by the same taxonomist at different times. This led us to develop a data model of plant descriptions and an associated ontology that defines the terms used in describing plants [7]. Taxonomy, as a traditional discipline, is resistant to changing working practices where extra time requirements to record higher quality data would be imposed on the individual taxonomist. This is emphasised by the fact that any improvement in data, tends not to benefit the taxonomist capturing it, as much as other taxonomists who interpret it later. We therefore wanted to improve the semantics and rigour of the recorded data whilst minimizing the burden of data capture and still allowing taxonomists to adequately describe their specimens. Although we describe our work in the context of plant taxonomy, we believe our approach is applicable to any domain where the capture of the semantics of the data in the database is important.

Creating appropriate, good quality data entry interfaces (DEIs) for databases is traditionally a difficult and time-consuming process for an IT expert. This mirrors the situation in GUI development in general. Two relevant strands of research do however continue to address the problem of improving the generation of UI. In one strand, research into model based (and other) user interface development environments [e.g. 8,9,10,11] attempt to combine abstract modelling with a more systematic approach to interface development. In these methods the UI developer investigates and models their understanding of the domain (as well possibly the task, presentation and layout models), to specialise the interface design for that domain. Abstraction in itself does not free the UI developer of the need to select appropriate interaction objects (although they may only be selecting abstract versions, with the details of the concrete coding being done automatically [12]). A convergent strand of research concerns automatic UI generation. Automatic UI generation is often based upon some form of model based solution or abstract design, which uses a presentation model to control the selection and layout of DEIs, based on the modelled tasks and/or domain (e.g. Janus [8] and Mecano [13] primarily use a domain model whereas Trident [14] and Modest [15] primarily use a task model). These approaches still require substantial investment by a UI developer, particularly if they are to be successful in creating a useful domain specific interface, and as Novak has observed '*Nobody will create ap-*

plications using specifications (models), if they can do it faster directly editing' [16]. This is doubtless one of the reasons that model based approaches have so far failed to achieve widespread commercial adoption, despite a strong research base [17].

Database interface research tends to lag in regard to general database research [18] so whilst there have been advances in database applications and data modelling to effectively store complex data, there have not been equivalent advances in approaches which promote the ability to capture rigorous data for such applications. Ontologies are increasingly used to describe and define complex data. Using an ontology to control the data entry for a database has the potential to ensure that better quality data is captured and that data from differing data providers will be compatible. It may also allow a DEI to be created that allows domain users to enter data using terms with which they are familiar but which are clearly defined semantically. Existing ontology based approaches for data entry are, however, generally still limited to using automatically generated forms-based data entry interfaces, unless manual editing is used (e.g. [19]). These systems are designed to populate a knowledge base describing relationships between described instance items of interest, rather than regulating the capture of the description of a complex concept. An alternative approach for UI generation involves using an ontology to provide domain knowledge to a system that can automatically generate a user interface based on a presentation model to capture data. An ontology-based approach of this kind has been proposed in regard to universal UI design [20] but the approach has not been widely investigated, nor has it addressed the needs of rigorous data entry.

This paper presents an ontology-based approach to semi-automatically generate data entry interfaces to databases. The remainder of the paper is organised as follows: section 2 provides an overview of the conceptual approach and introduces the domain of biological taxonomy. The process of defining data requirements for the data entry interface is described in section 3. Section 4 considers the models and processes involved in generating the data entry interface. Section 5 concludes the paper.

2. Overview Of The Approach – Concept and Application

Our basic approach for generating data entry interfaces to databases, for the capture of descriptive data about concepts of interest, is shown in fig.1. In the design of our system we have adopted a model based approach in that we have task, domain and presentation models. The task model is pre-determined to be the general task of data entry for a database. The *data entry task model* (see fig.1) is encapsulated within the system. The only modifiable aspect of the task model is the order in which the data entry task is completed (see Section 3).

2.1 Domain Models

We use a series of domain models to represent domain knowledge (see fig.1). The *abstract domain model* is transformed into a *concrete domain model* by mapping an existing domain ontology to it. A domain expert specialises this *concrete domain model* to create a *specialised domain model* for a given project of work (see Section 3).

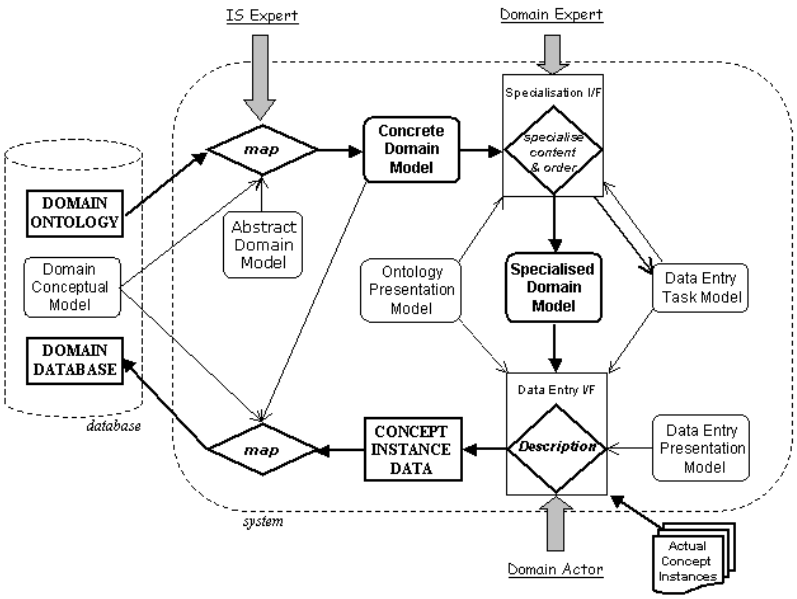

Fig. 1. Ontology Driven Automated Generation of Data Entry Interfaces Approach

2.1.1 Abstract Domain Model Our system is designed to capture data concerning any high level concept that may be sub-divided into a hierarchy of defined constituent sub-concepts ('*description objects*') that are themselves described by instantiating *attributes* that they possess. Each *attribute* of a *description object* can be instantiated within the limits of its *value constraints*. These *value constraints* might restrict entered data (such as the data type or numerical range of entered data), or define selection from a limited set of *value objects*. A *value object* represents a defined concept that can be used to instantiate an *attribute*. Additional entities (modifiers and units of measurement) allow more detailed description of *attributes* and their value constraints. This data format is captured in an a*bstract domain model* (fig.2). This data format could be widely applicable, representing both physical and abstract concept domains (e.g. a control system process or academic department).

2.1.2 Domain Ontology. Initially, in order to instantiate the *abstract domain model* with actual domain knowledge, we map an appropriate domain ontology to it, to create a *concrete domain model* (see fig.1). Our approach assumes the existence of an appropriate domain ontology, which does not necessarily have to be created solely for this system. Ontology is a widely used term, with a variety of meanings [21]. The commonly quoted definition '*a specification of a conceptualisation'* [22] is generally appropriate for our usage. Specifically, the domain ontology is a semi-formal, constrained and structured form of natural description language, with defined terms and possible relationships between them. Even so defined, ontologies can contain many different objects and relationships with various semantics.

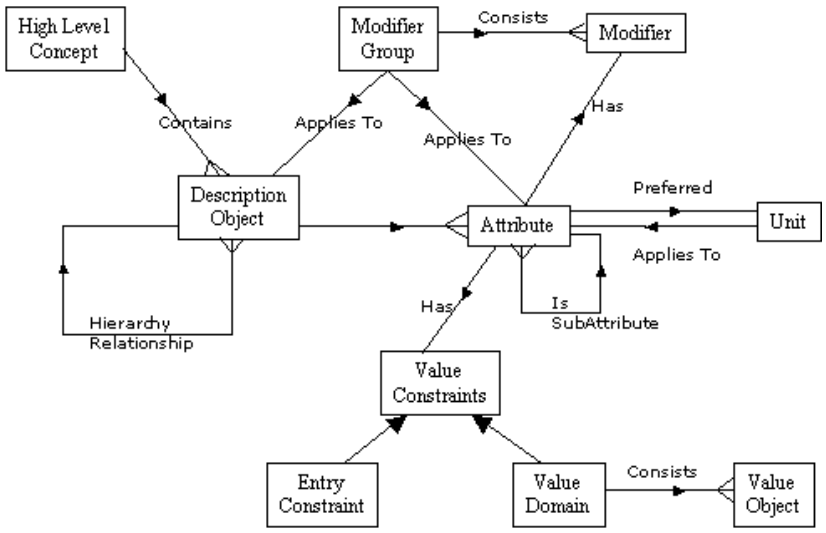

Fig. 2. Abstract Domain Model: the conceptual model for controlling domain knowledge in the system. The hierarchy of *description objects* is formed using the Hierarchy Relationship. Some details, such as synonym relationships, are not shown for clarity

In our example domain, an ontology to control the description of a set of plant specimens is being used as a domain ontology. Classical plant taxonomists describe plant specimens in terms of their observable characteristics, and interpret patterns of shared characteristics to evaluate relatedness between specimens in order to define taxonomic groups and compose a hierarchical tree (taxonomy) of relationships between these groups. As part of a project which attempts to standardize the composition of taxonomic descriptions, a defined terminology for the description of flowering plants (angiosperms) has been created in collaboration with taxonomists from the Royal Botanic Garden Edinburgh [7]. The ontology is composed of 'Defined Terms' (terms with associated definitions and citations) and relationships between these terms. As shown in fig.3, there are three major subclasses of defined terms used to create descriptions of biological specimens: Structure terms representing all the possible anatomical structures of a given specimen (e.g. petal, stamen); Property terms, which represent aspects of a structure that might be described (e.g. length, shape); and State terms which represent the actual values for a qualitative property of a given structure (e.g. round, yellow). In our description model 'quantitative properties' are scored by numerical values. 'State Group' relationships in the ontology capture permitted relationships between groups of States and the set of Structures that they may be used to describe. 'Is-part-of' forms the central organising relationship for the ontology, and allows representation of a hierarchy of all the possible structural relationships found on any given specimen (e.g. a blade is part of a leaf, or part of a leaflet, which itself is part of a leaf).

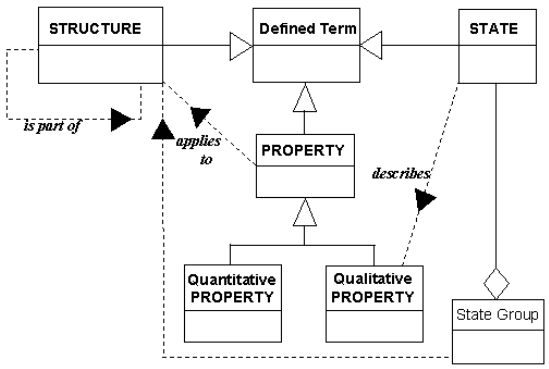

Fig. 3. Major terms and relationships represented in the angiosperm domain ontology

2.1.3 Concrete Domain Model. The potential variation in composition of domain ontologies makes their automatic adaptation for a domain model a nontrivial task [23] that defies automatic mapping of the domain ontology, thus requiring the intervention of an IT expert actor. The IT expert makes a mapping between the *abstract domain model* and the particular domain ontology's conceptual model. This allows the system to derive the *concrete domain model* from the imported domain ontology. It is only necessary to perform this mapping once for a given domain conceptual model. Where the database schema and ontology conceptual model are based on the same domain conceptual model, this mapping also allows the system to format the entered data for transfer to the database application. Where this is not the case, a second expert mapping would be required for each database schema

In order, to perform the domain ontology mapping, a number of key objects and relationships need to be identified or derived. At the fundamental level, *description object*s need to be identified along with a primary description object hierarchy interrelationship and root *description object*(s) (to form a *description object hierarchy*). *Attribute* objects must be identified or derived from ontology terms and/or relationships between *description object*s and possible *value object*s. In addition, the applicability of *attributes* to *description object*s is identified. *Value object*s must be identified from the descriptive terms that could form possible values of a *description object* via an *attribute* relationship (*value object*s can also be *description object*s themselves or instances of *description object*s). Beyond these basic terms and relationships, the *abstract domain model* can have modifiers, units, synonym relationships and various other aspects mapped to it (see section 3 for examples).

In mapping the angiosperm domain ontology (fig. 3) to the *abstract domain model* (fig. 2), specimens represent the *high-level concepts* that are described. Their constituent structures map to *description object*s, and their is-part-of relationships form the *description object hierarchy* relationship. Properties and state group relationships form the *attributes* of a *description object*. States form *value object*s, which belong to an *attribute* and which are constrained over a *'value domain'* defined by the permitted grouping relationships between structures and states.

2.1.4 Specialised Domain Model. A data entry interface based on the whole angiosperm domain ontology would be too large for usability and would cover a much larger number of structures and characteristics than a taxonomist would utilise in any one taxonomic project. Individual projects would typically be restricted to only a small subset of the angiosperm group of plants. Additionally, taxonomists are interested in different sets of specimen characteristics dependent on the focus of their work. The exact data requirements of a given taxonomic project must therefore be established. Normally, taxonomists do this by creating paper-based templates for each project, which have entries for the major describable characteristics of the specimens that they wish to record. Our system provides an equivalent to this process by allowing taxonomists to create *specialised domain models* based on the angiosperm domain model, which enables the system to present them with data entry forms based solely on the data and semantics relevant for their particular project.

2.2 Presentation Model

There are two presentation models in the system one for ontology presentation and one for data entry (see fig.1). In order to allow the expert taxonomist to create a *specialised domain model*, the system uses a modelling tool (the s*pecialisation interface*) which presents the entire angiosperm domain model for exploration and editing. The ontology presentation model is used in this tool, to provide a general layout presentation for displaying ontologies based on the *abstract domain model*. This presentation model is also utilised to display aspects of the *specialised domain model* in the final data entry interface. The *ontology presentation model* was derived by user requirement and evaluation testing and is now captured within the system. The *data entry presentation model* determines the layout and selection of interaction objects for the data entry interface. Different *data entry presentation models* could be utilised as plug-ins. User testing has been used to develop one such model.

3. Specialising the Data Entry Requirements

3.1 Domain and Task Model Specialisation Tool

This section describes the process by which domain experts specialise the data requirements for a particular project. The result of this process is the *specialised domain model* and the default task ordering of the *task model*. The specialisation tool consists of an interface (fig. 4) that allows users to interact with the domain model and task order. A domain expert can determine which *description object*s (i.e. plant structures) they wish to include in the Data Entry Interface, the *attributes* of those *description objects* they wish to use, and the specification (constraints) of those *attributes*. The system interprets the *task* and *concrete domain models* (Table 1) using its *ontology presentation model* to determine the layout and interface interaction objects, including the use of colour or icons to indicate summary information (e.g. greying out excluded elements or warning icons for *attributes* with no possible values).

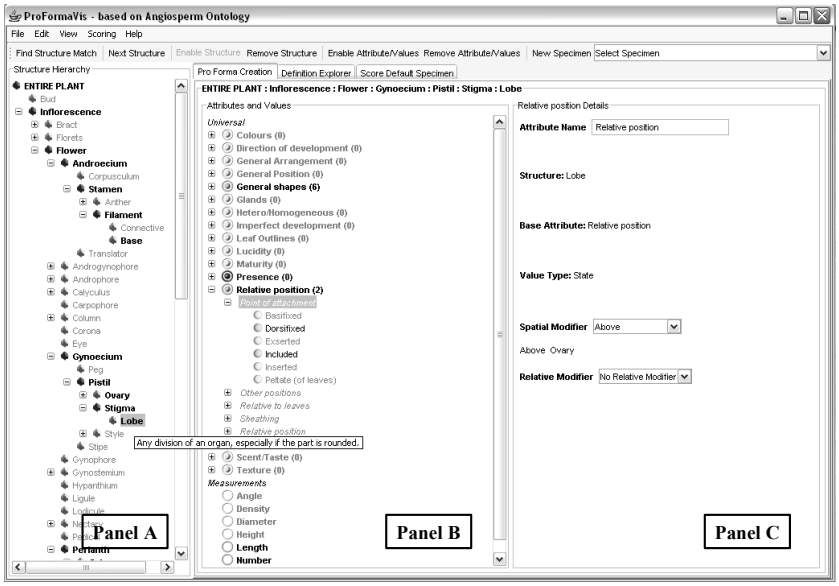

Fig. 4. Specialisation Interface Screenshot: 4A represents the *description object hierarchy*. 4B represents the potential *attribute*s and their related possible *value objects* for a selected *description object*. 4C allows specification of the selected *attribute* details

3.1.1 Specialising the Description Object Hierarchy A view of all the *description objects* (i.e. plant structures) is presented in a *description object hierarchy* (fig. 4A), through which the user can select description objects for inclusion in the DEI. The *description object hierarchy* view is primarily based upon an interpretation of elements of the *concrete domain model* as summarised in Table 1.

In addition to interpreting the *concrete domain model*, the *task model* is interpreted to provide the order of the *description object*s (presented as the order they appear in the *description object hierarchy*) which can be modified by the user. Any alterations to the default task order are captured in the *task model*. This functionality is provided to reflect the working practice of taxonomists, who, within the general task of describing specimens, may want to specify the order in which they describe the particular characteristics of the specimen, to fit with traditional biological description methodologies.

3.1.2 Specialising Description Objects By selecting *description object*s in the hierarchy, the user can access its potential *attributes*. A second hierarchical view is presented for the selected *description object*, (fig.4B), which allows users to select *attributes* for inclusion and alter *value constraints* by specifying the set of possible *value object*s. This *attribute-value* hierarchy accesses the *attributes* and their potential *value object*s by interpreting the *concrete domain model* (Table 1). The details of a selected *attribute* can be further specified (fig. 4C), such as adding spatial modifiers to

accurately represent where in a *description object*, an *attribute* is being measured. *Attributes* can also be more radically varied by adding relative modifiers which can change the nature of an *attribute* by relating it to another *description object* and *attribute* (e.g. to capture the ratio of leaflet-length to leaflet-width). Other *attribute* specification is aimed at affecting the data entry interface without modifying the fundamentals of the data that will be exported to the database. This includes influencing the selection of interaction objects (e.g. by varying the importance of multi-media display), or affecting presentation aspects (e.g. changing the display name of the *attribute*).

Description objects can also be declared to be concrete. Normally data entered about a *description object* is generalised data reflecting all actual instances of the *description object*, however sometimes it is preferable to enter data about individual actual instances of the *description object*. This allows taxonomists to record a volume of related numerical data suitable for later statistical analysis, if desired.

Table 1. This table shows the mapping of the *concrete domain model* to the s*pecialisation interface* using the *ontology presentation model*. The *task model* controls node ordering

Specialisation Interface Element	**Ontology Presentation Model INTERPRETS Concrete Domain Model Element**
Description Object Hierarchy (fig. 4A)	
Node Identities	*Description Object*s
Node Presentation	*Description Object* Fixed Data Elements (e.g.name)
Node Presentation /Interaction	*Description Object* DEI Inclusion Status
Tree Structure	*Description Object* Hierarchy Relationships
Attribute-Value Hierarchy (fig. 4B)	
Attribute Node Identities	*Attributes* (of selected *description object*)
Value Nodes	*Attribute* to *Value Object* Constraints
Node Presentation	*Attribute* Fixed Data Elements (e.g. name)
Node Presentation /Interaction	*Attribute* and *Value Object* DEI Inclusion Status
Tree Structure	*Sub-attribute* and *Attribute-Value* Relationships
Attribute Details Specification (fig. 4C)	
Element Presentation	*Attribute* Fixed Data Elements (e.g. value type)
Element Presentation /Interaction	*Attribute* Modifiable Data Elements (e.g. relative modifier)
Element Presentation /Interaction	*Attribute* Modifiable Display Elements (e.g. name)

3.1.3 Specialisation Restrictions. The domain expert in this specialisation process cannot transform the domain model in such a way as to make it inconsistent with the original domain ontology, for example, they cannot use an *attribute* for a *description object* that does not have an appropriate relationship in the domain model. This ensures the data exported by the system is compatible with the data model underlying the original ontology. The domain expert cannot directly alter the *data entry presentation model* (for example by choosing the actual data entry abstract interaction objects, although they can alter the data in the *specialised domain model* upon which determi-

nations are made by the *data entry presentation model*). This ensures a modelling split between data determination and presentation, thus avoiding confusion between the two different processes.

3.2 Domain Expert User Considerations

Editing domain models in model-based approaches is usually a task reserved for IT experts. In our case, a domain expert is performing this operation, so the interface must be configured towards a user who is not necessarily familiar with modelling or ontological terminology. The *ontology presentation model* in the system attempts to aid ease of understanding by using appropriate domain terms based on the domain ontology mapping (e.g. referring to structures instead of *description object*s). In addition, the *ontology presentation model* generally interprets the ontology to attempt to allow easy navigation and feedback, with easy access to related domain knowledge (e.g. access to definitions). Object definitions drawn from the domain ontology provide users with the knowledge to make informed editing choices, thus supporting the eventual capture of semantically good quality data. Definitions are provided in a variety of means, including mouse over pop-up summaries (see fig 4A) and a definition viewer gives additional details of selected terms on request. Evaluation by taxonomists from the Royal Botanical Garden Edinburgh showed that the *description object hierarchy* captured in the domain model allows users to navigate the potentially large description space, using their own domain knowledge.

4. Data Entry Interface

This section describes how data is captured based on the *specialised domain* and *task models*. As the details of the data to be entered have been determined by a domain expert actor, there is no further direct intervention by an IT expert before the *data entry interface* (DEI) is generated. Some fixed design decisions (such as basic architecture layout) are captured in the system, the DEI being generated by interpretation of the *specialised domain, task* and two presentation models. Working with one high-level concept instance (i.e. plant specimen), a domain actor enters data for the *attributes* of one *description object* (i.e. structure) at a time.

4.1 Navigating the Data Entry Interface

The DEI provides a view of the *description object hierarchy* (fig. 5A) and presents a series of windows with groups of interaction objects for data entry (fig. 5B). The default order of the *description object*s and *attributes* presented by the system to users for data entry is interpreted from the task model and can be overridden by the data entry user by selecting *description object*s directly from the *description object hierarchy* display (fig. 5A). This display is controlled by the *ontology presentation model*, as described in the previous section, although in this case it interprets the *specialised domain model* and is not editable. In the DEI context, the display uses colour, icons

and mouse-over text to indicate summary data about the data entry status of the current specimen (e.g. to represent whether a structure has been skipped over in normal task order). By providing navigation context and summary data for each *description object*, the user is assisted in making informed data entry decisions.

For each *description object*, the DEI generates a grouped set of *attribute* presentation units (fig. 5B), which are complex data entry interaction objects that contain all the data and interaction capability required to enter data for one *attribute* of the *description object*. The *data entry presentation model* controls the level of grouping. In taxonomy, we group all *attributes* for one plant structure in one window, which fits with their observational methodology. An appropriate grouping, combined with the hierarchy view and the nature of the pre-determined task, offsets one of the traditional drawbacks of automatic generation, that users require information from multiple objects in one window [24], as all the required information to make an informed data entry decision should be available. The layout of the presentation units within the grouped screen is controlled by a 'place one below the other strategy'. More complex layout strategies could be defined by the *data entry presentation model*.

Where a *description object* is concrete (see section 3), the group of presentation units refer to one actual instance of a *description object*. Multiple instances of the one *description object* are often captured in this case. The DEI generates multiple grouped sets of presentation units for the *description object*, as required by the DEI user.

4.2 Entering Data in Attribute Presentation Units

The user enters data for individual *attributes* of a *description object* using a presentation unit. A presentation unit consists of four major components to support data entry: three data entry interaction objects; and a display with the *attribute* data required for the user to make meaningful data entry choices (such as *attribute* name, current entry status). This data is interpreted from the domain model; the presentation is fixed and captured in the system.

The primary data entry interaction object controls selection or entering of the values. The implementation of this interaction object varies. The abstract implementation is determined by the *data entry presentation model*, which selects abstract interaction objects (AIO) from a system library. These AIOs are internally specialised by the relevant *attribute* and related data captured in the domain model to create the concrete interaction objects. This specialisation controls aspects such as internal layout management (for example the number and layout of checkboxes), the display of names and icons, etc. In order to determine an appropriate AIO the *data entry presentation model* accesses various defined criteria of the underlying *attribute* object data. One criterion used to determine this AIO, is value type, of which there are two basic choices – *value object* selection and text entry. Selection involves choosing from a set of *value object*s, whereas text entry allows the user to enter data as desired. As this approach is based on an ontology for ensuring quality of collected data, text entry is usually limited to numerical data entry, as free text entry would allow recording of data not compliant with the domain ontology. Data type is generally used as a criterion for making automatic IO selection choices; in this case, the data type of the allowed values of the *attribute* is used. Multi-media representation of descriptive terms

is very important in taxonomy, as in many other domains. The presence of multi-media definition representations is thus another criterion that can be used by the presentation model. The importance of the multi-media representations may also vary from *attribute* to *attribute*, and a tag can be assigned to an *attribute* to specify this in the *specialisation interface*. Other common criteria, which can be accessed by the presentation model, include data cardinality, data precision, numerical range constraints, etc.

4.3 Controlling Nuances of Entered Data

The remaining two interaction objects in a presentation unit control the adding of semantic nuances to the data. The first is an interaction object for adding applicable modifiers to the entered data (e.g. frequency modifiers like ‘rarely', 'usually', *etc*.). The available modifiers are based on the *attribute* links captured in the domain model (see fig. 2) and allow users finer control of the entered data; a second interaction object controls the interpretation of multiple values. Initially one presentation unit is displayed for each *attribute* requiring instantiation. The data entry process however might require additional presentation units being generated to capture nuances of description. A common example of this situation is in distinguishing 'AND-ing' from 'OR-ing'. This applies where a number of values for the same *attribute* are applicable to every individual physical instance of the *description object*, as opposed to the situation where the instantiated *attribute* has different values on different individual *description object*s (e.g. to distinguish between a specimen whose individual petals are white *and* purple versus a specimen whose individual petals are *either* white petals *or* purple petals). As the permutations of this situation can be quite complex, the user is required to instantiate one *attribute* presentation unit for every permutation of individual *description object* instances. The system can replicate presentation units to allow entry of these different permutations as required by the domain actor using the DEI. This process is made less intrusive by not basing the grouping of presentation units upon the available screen space, but instead allowing the expansion of effective screen space using scrolling. For example (as in Fig. 5B) a taxonomist entering data for a specimen which had some purely white petals and some petals that were both white and purple would select white in the petal colour presentation unit, and click on ‘Enter Another Score’ button. This would generate a copy of the petal colour presentation unit, where the taxonomist would select both white and purple in a single presentation unit.

4.4 Exporting Data

Once a user has entered the data for one *high-level concept* instance (i.e. plant specimen), they can enter data for other instances of the concept. The data for each specimen can be exported to the database. This exported data is formatted using the mapping between the domain ontology conceptual model and the *concrete domain model*. The interface could also be reloaded from the database by a reverse procedure.

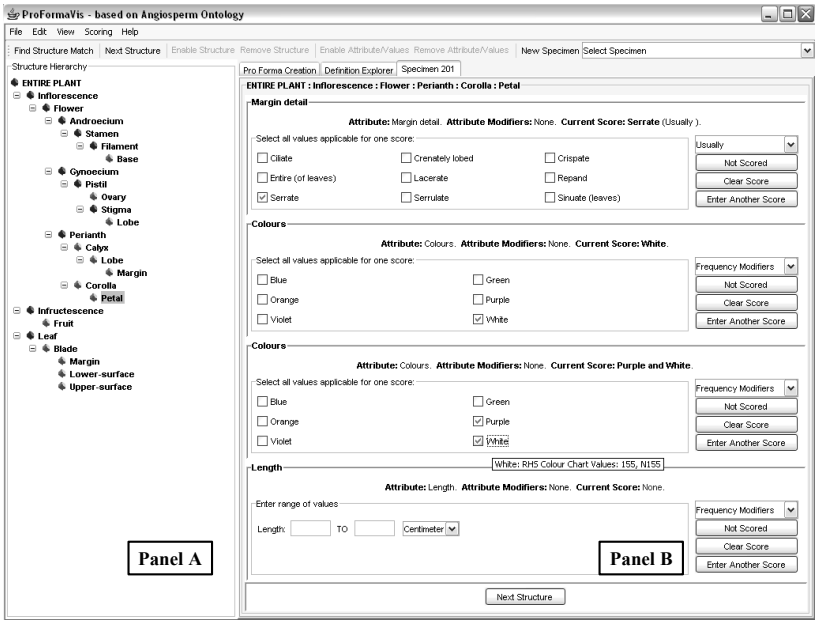

Fig. 5. Data Entry Interface: 5A is the *description object hierarchy* view providing both compositional context and a means of overriding the default data entry ordering. 5B is a group of *attribute* presentation units for the selected *description object*

5. Conclusions

The system described in this paper utilises domain knowledge from a domain ontology and domain experts to specify the data requirements of an automatically generated data entry interface to databases. This approach aims to improve the quality of data entered by users, without overburdening users or interface developers. In Szekely's retrospective [24], this work would fall into the model-based automatic interface generation approaches, specifically those that primarily allow users to access and specify a domain model. This system's domain model however, is based on an existing ontology to ensure that the semantics of the data are maintained. By tying the entered data to a domain ontology, semantically high quality data can be generated and be entered into a database based on a data model related to the original ontology.

Despite their potential benefits, model-based automatic generation approaches have not been widely adopted commercially and have been criticised as being unable to produce quality, appropriate interfaces [24]. Our approach, however, offers access to a modelling tool for domain experts to specialise the domain model rather than for interface developers. This specialisation can be done for individual projects, thus improving the appropriateness of the generated interface. An appropriate, good quality interface is a useful element for ensuring that captured data is an accurate representa-

tion of the intent of both the data entry user, and the project designer in a multi-user system. Furthermore, by limiting our approach to a descriptive data entry task, we have already gone some way to limiting the possible permutations of the interface, allowing the presentation model to be more appropriate. Focussed approaches also tend encourage wider adoption of new approaches than universal approaches that attempt to solve all problems at once [25]. Another contributory factor to the lack of quality in traditional automatically generated interfaces, lies with the difficulties of automatically selecting appropriate AIOs using a predetermined presentation model. By focussing on data entry tasks and allowing tailoring of the *data entry presentation model* to particular domains, this approach supports a more appropriate AIO selection.

By using the domain ontology and domain experts to create the *specialised domain model*, the approach provides benefits in the avoidance of time consuming and possibly distorted domain knowledge acquisition by a UI expert from a domain expert who must possess or acquire this knowledge to support their work in any case. A modelling tool has been developed for the system which has been tailored for easy and informed use by domain experts who are not familiar with modelling techniques.

Initial informal user evaluation for our approach has been positive. It has shown for example that taxonomists are able to navigate and interact with the *concrete domain model* in the *specialisation interface* to create effective *specialised domain models*; that taxonomists appreciate the value of access to the exact semantics of the domain terminology being used; and that data semantically consistent with the angiosperm ontology can be collected and exported to a database. More extensive user testing is planned, both in depth with the existing angiosperm ontology and with other domain ontologies. Our approach has been applied to the instance field of specimen description in plant taxonomy. We, however, believe the approach can be more widely applied in data entry applications, particularly where semantically high quality data capture is important and where there are variations in the data requirements for different projects. The provision of supporting tools for use by IT experts in mapping from *domain conceptual models* to the system's *abstract domain model*, and for generating alternative *data entry presentation models* will ease expansion of this approach beyond biological taxonomy applications.

We would like to acknowledge and warmly thank BBSRC for funding of this research, and the Royal Botanical Garden Edinburgh for their help in evaluation and development.

References

1. Raguenaud, C., Kennedy, J., Barclay, P.J., The Prometheus Database for Taxonomy, 12th International Conference on Scientific and Statistical Database Management, 250-252, 2000
2. Pullan, M., Watson, M., Kennedy, J., Raguenaud, C., Hyam, R., The Prometheus Taxonomic Model: A Practical Approach to Representing Multiple Taxonomies, Taxon 49, 55-75, 2000
3. Graham, M., Watson, M.F. and Kennedy, J., Novel Visualisation Techniques For Working With Multiple, Overlapping Classification Hierarchies, Taxon 51(2), 351-358, 2002
4. Dallwitz, M.J., A general system for coding taxonomic descriptions, Taxon 29, 41-46, 1980
5. Maddison, D.R., Swofford, D.L. & Maddison, W.P., NEXUS: An extensible file format for systematic information. Systematic Biology 46, 590-621, 1997

6. CBIT, (2003) Lucid, developed by The Centre for Biological Information Technology: University of Queensland, Australia. URLs: www.cpitt.uq.edu.au; www.lucidcentral.com
7. Paterson, T., Kennedy, J., Pullan, M. R., Cannon A. J., Armstrong, K., Watson M. F., Raguenaud C., McDonald S. M., and Russell G., A Universal Character Model and Ontology of Defined Terms for Taxonomic Description, DILS 2004, LNBI 2994, 63-78, 2004
8. Balzert, H., Hofmann, F., Kruschinski, V., Niemann, C.: The JANUS Application Development Environment-Generating More than the User Interface. In Proceedings of CADUI'96, Namur: Presses Universitaires de Namur, 183-207, 1996
9. Elwert, T., Schlungbaum, T.: Modelling and Generation of Graphical User Interfaces in the TADEUS Approach. In Palangue, P., Bastide, R., (Eds.), Designing, Specification and Verification of Interactive Systems, Wien, Springer, 193-208, 1995
10. Szekely, P., Sukaviriya, P., Castells, P., Muhktumarasamy, J., Salcher, E., Declarative Interface Models For User Interface Construction Tools: The MASTERMIND Approach, In Engineering for Human-Computer Interaction, 1996.
11. Butler, K.A., Designing Deeper: Towards a User-Centered Development Environment Design in Context. In Proceedings of ACM Symposium on Designing Interactive Systems: Processes, Practices, Methods, & Techniques DIS´95, 131-142, 1995
12. Zloof, M.M., 'Selected Ingredients in End-User Programming'. In Proceedings of the Working Conference on Advanced Visual Interfaces, 1998
13. Puerta, A.R., Eriksson, H., Gennari, J.H., and Mussen, M.A., Beyond Data Models For Automated User Interface Generation. In People and Computers IX HCI'94, 353-366, 1994
14. Vanderdonckt, J.: Automatic Generation of a User Interface for Highly Interactive Business-Oriented Applications. In Companion Proceedings of CHI'94, 41 & 123-124, 1994
15. Hinrichs, T., Bareiss, R., Birnbaum, L., Collins, G.: An Interface Design Tool based on Explicit Task Models. In Companion Proceedings of CHI' 96, 269-270, 1996
16. Novak, G.S. (2003) Novak's rule: http://www.cs.utexas.edu/users/novak/
17. Traeteberg, H., Molina, P.J., Nunes, N.J., Making Model-Based UI Design Practical: Usable and Open Methods and Tools. In Proceedings of the 9th international conference on Intelligent user interface, 376-377, 2004
18. Carey, M., Haas, L., Maganty, V., Williams, J., 'PESTO: An Integrated Query/Browser for Object Databases'. In Proceedings of the 22nd International Conference on Very Large Data Bases, 1996
19. Gennari, J., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W., The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. 2002. URL: http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0943.pdf, http://protege.stanford.edu/index.html
20. Furtado, E., Furtado, J.J.V., Bezerra Silva, W., Tavares Rodrigues, D.W., da Silva Taddeo L., Limbourg, Q., Vanderdonckt, J., An Ontology-Based Method for Universal Design of User Interfaces, In Proceedings of Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends, Lille, 2001
21. Guarino, N., Giaretta, P., Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Mars, N.J.I. (ed.), Towards Very Large Knowledge Bases, IOS Press 1995
22. Gruber, T.R., A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5, 199-220, 1993
23. Wang, X., Chan, C.W., Hamilton, H.J., Design Of Knowledge-Based Systems With The Ontology-Domain-System Approach. SEKE 2002, 233-236, 2002
24. Szekely, P.,. Retrospective and Challenges for Model-Bases Interface Development. In Computer-Aided Design of User Interfaces, Namur University Press, xxi--xliv, 1996
25. Myers, B., Hudson, S. and Pausch, R., Past, Present, and Future of User Interface Software Tools, ACM Transactions on Computer-Human Interaction 7, 3-28, 2000