Knowledge Elicitation Plug-in for Protégé: Card Sorting and Laddering

Yimin Wang[1], York Sure[1], Robert Stevens[2] and Alan Rector[2]

[1]Institute AIFB, University of Karlsruhe, Germany
{ywa,sure}@aifb.uni-karlsruhe.de
[2]School of Computer Science, University of Manchester, M13 9PL, UK
{rector,robert.stevens}@cs.man.ac.uk

Abstract. Ontologies have been widely accepted as the primary method of representing knowledge in the Semantic Web. Knowledge Elicitation (KE) is usually one of the first steps in building ontologies. A number of ontology editors such as Protégé have been developed to assist users in building ontologies efficiently. However, traditional KE techniques, such as card sorting and laddering, are not yet supported, but performed manually and outside of such tools. In this paper we present a methodology and a corresponding plug-in for Protégé that allows graphical ellicitation knowledge from documents using card sorting and laddering approaches. Our aim is to seamlessly integrate the KE techniques into the ontology building process to make ontology building more efficient and less error-prone. As a side-effect the persistent storage of card sorting and laddering results allows for later traceability of ontology development. KE largely depends on user interaction with the plug-in, therefore we employed user-centred design principles to capture requirements. After implementation, the plug-in was evaluated thoroughly against the requirements. The evaluation shows that this KE plug-in meets many of the user's expectations and indeed saves them considerable time when building ontologies.

1 Introduction

The explosion of digital knowledge makes finding accurate information effectively an increasingly important topic. Making knowledge explicit, e.g. in the form of ontologies and corresponding metadata, offers many opportunities to facilitate effective knowledge access. One of the first steps in building ontologies is usually a knowledge elicitation (KE) process, which is also known as an important branch of knowledge acquisition. Traditional KE is a kind of labour-intensive manual work, extremely time-consuming, and often not well connected to further steps in ontology engineering. What is needed are more usable, handy and in particular well-integrated toolkits for knowledge elicitation.

Several standard knowledge acquisition/elicitation techniques, such as card sorting and laddering, have been developed to help in organising domain expert's ideas into basic structures and to recover tacit knowledge. Card sorting has been used for several decades, and it is remarkably useful for finding out how people categorise things [1, 2]. Laddering was first introduced by Hinkle [3], a clinical psychologist, in order to

model the concepts and beliefs of people by an unambiguous and systematic approach. Most of these knowledge acquisition/elicitation techniques are visual or graphical. The traditional card sorting and laddering methods are, however, extremely difficult to manage and track back – you will find it nearly impossible to keep the record for hundreds of cards or paper pieces and go back to a prior status without a complicated series of actions, such as video tape recording, searching and playing back.

A key motivation for our work comes from the CO-ODE project[1] where we experienced a rather big gap between manually applied knowledge elicitation techniques, in particular card sorting and laddering, and building of ontologies with Protégé. Protégé [4] is one of the most popular ontology editors which supports many of the tasks of ontology engineering. Protégé enables users to create ontologies by defining concepts, specifications, relationships, annotations and other information within a certain domain. In our CO-ODE project tutorials users wanted to build a pizza ontology with the Protégé OWL Plug-in [5], however, often our tutees preferred to write the pizza terminologies and properties in a pile of cards and to construct the conceptual taxonomy by arranging the cards on the table. After that, the users recorded the outcome in Protégé by manual transfer from the hard copy on their real desktop. It turned out to be very common that users were getting confused and found it difficult to manage a whole table of cards. Finally, any re-sorting of the cards required careful adjustment of the ontology modelled in Protégé.

Our core contribution consists of a novel technique for integrating the KE techniques of card sorting and laddering into ontology building (cf. Section 3). The technique has been implemented in a corresponding plug-in for Protégé (cf. Section 5) that allows graphically eliciting knowledge from documents using card sorting and laddering approaches. Our aim is to seamlessly integrate the KE techniques into the ontology building process to make ontology building more efficient and less error-prone. As a side-effect the persistent storage of card sorting and laddering results allows for later traceability. KE largely depends on user interaction with the plug-in, therefore we employed user-centred design principles to capture requirements (cf. Section 4). After implementation the plug-in was evaluated thoroughly against the requirements (cf. Section 6). The evaluation shows that this KE plug-in meets many of the user's expectations and indeed saves them considerable time when building ontologies.

2 Related Work and Challenges

Related work includes two major aspects. The knowledge elicitation techniques deal with the theoretical issues, while the ontology engineering aspect aims to facilitate users in the process of building ontologies.

From the mid 1980s, people began to do research on expert systems as a subdiscipline of Knowledge Engineering, and it was also the starting point for the scientific research on KE. People tried to develop KE techniques to get knowledge with effectiveness, efficiency and correctness. A number of these methods were borrowed from cognitive science and other disciplines such as Anthropology, Ethnography, and

[1] `http://www.co-ode.org`

Business Administration [6, 7]. KE techniques were effectively used in early 1990s, with the popularity of graphical based personal computer system [8].

Card sorting is a comprehensive technique of knowledge elicitation methods and is now being used in several disciplines such as Knowledge Engineering, Psychology, and Marketing. In the field of KE, card sorting is considered to be one of the most effective ways for eliciting the domain expert's idea about the knowledge structure. Much evidence shows that card sorting has many positive aspects in making a useful and reasonable elicitation experiments, including helping the respondents to recall the domain concepts; providing a structuralized concepts pile for future processing – such as laddering; fast acting and easy handling [1, 8]. Figure 1 shows a real use case of card sorting.

Fig. 1. Traditional card sorting

Laddering has been widely used in the field of knowledge elicitation activities in recent years. The basic purpose of the laddering method is to elicit people's goals and values [8, 9]. People from knowledge elicitation community have developed a well-established range of formal semantics, procedures and notation for building ladders. Based on the Rugg and McGeorge's [9] categorisation, laddering can be used for three major purposes – they are using laddering to elicit sub-classes, explanation, goals and values. Laddering has been implemented as an independent tool in a software toolkit, called PCPACK [10] with integration of the CommonKADS method [11]. But the implementation in PCPACK lacks the compatibility with full support of the OWL language, and the well integration with state-of-the-art ontology engineering tools, like Protégé. In this paper, we are going to use the laddering method to build up the data and object properties for concepts, e.g. a typical conceptual ladder example is shown in Figure 2.

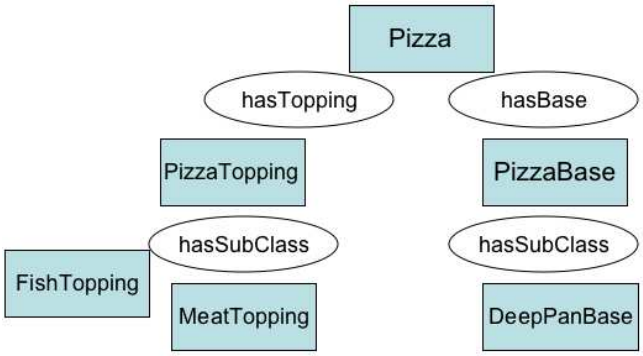

Fig. 2. Concepts map with properties

Not many general purpose approaches have been invented for building ontologies. The available methodologies are generally frameworks with descriptions and outlines on how to build ontologies [12, 13]. Often, such methodologies focus on the ontology engineering steps without much support for the very early stages of KE. The most recent methodology DILIGENT [14] supports the dynamic nature of ontologies and also includes support for argumentation between different actors during the whole process.

The challenges tackled in this paper, therefore, are how to implement a knowledge elicitation methodology which extends existing ontology engineering methodologies and at the same time focuses on the development of a usable tool that supports graphically-oriented building of ontologies, using card sorting and laddering tools respectively, and which allows people to further refine their ontology in a (broader) ontology editor like Protégé.

Our technique and system aims to reduce the work-load for knowledge engineers and domain experts; increase the reusability of laddering and card sorting processes; effectively manage the KE tasks; and seamlessly integrate with an existing software system for ontology engineering.

3 KE Integration Technique

The traditional card sorting method generally consists of a pile of cards with the approximate size of a credit card, created by the researchers, who write or print the domain concepts on cards. A video tape recorder captures both the acts and voices of the entire procedure for future analysis. We can therefore find out that traditional card sorting has three major drawbacks, which are, easy to be destroyed, difficulty to be managed, and not practical to be transferred to computer files. For example, a blast of wind or a cup of coffee can easily disrupt the process, this manual process also cannot be shared on the internet, and the video information is also a bottle-neck while people have many tapes to manage or deliver via the internet.

The solution is straightforward. To avoid the fragility of actions, we can transplant the entire procedure into the computer, and by using a preliminary version-control

mechanism, the user can overall control their milestones when they sort cards and structure concepts. Obviously, this plug-in does not record the activities by capturing the screen just like a screen recording software, contrarily, it logs the activities performed by the user in a text file which is essentially easy to be transferred and managed.

Redo and undo mechanisms in text editors give us a hint to solve this problem by automating the tasks. The whole procedure and all its related matters – we call it version control manager – need to be temporarily stored in the memory and saved to the permanent storage devices if necessary. By doing this, the domain experts and developers are able to go back to anywhere if they want, all they need to do is to store the different versions of the tasks while they are standing at a milestone or a trap point.

Laddering techniques play an important role in discovering the potential relationships between the domain concepts. The laddering method is usually used combining with other KE methods such as card sorting. The subjects and objects within the ontology are inter-connected with several kinds of relationships elicited from the domain experts via laddering, and the structural source of subjects and objects are built via card sorting. As ontology is the structuralized domain knowledge base generated from experts, we can realise that laddering is undoubtedly essential while developing ontologies. So we are also going to implement the laddering technique as part of this plug-in, as well.

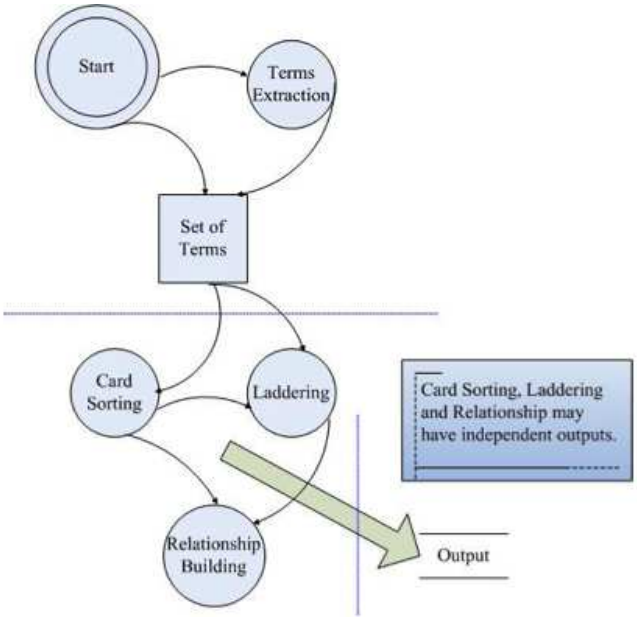

Fig. 3. KE integration technique roadmap

Figure 3 illustrates the process of applying KE techniques as part of ontology building. As indicated in the figure there are multiple ways to apply the various KE tech-

niques. In the following we briefly explain three frequently used ways to apply the process.

1. Start with a [Set of Terms] and perform card sorting and/or laddering
2. Start with a term extraction to retreive a [Set of Terms], then perform card sorting and/or laddering
3. Start with a term extraction to retrieve a [Set of Terms], perform card sorting and/or laddering, then perform relationship building

4 Design and Implementation

While we want to build **real usable** software, rather than a program for demonstration purposes, there are many principles to be followed, especially those related to the design of the user interface. The implementation phase is tightly coordinated with the interface design and there are many rolling procedures to refine the design and implementation respectively.

4.1 User-centred Design

In the Human-Computer Interaction (HCI) research field, user-centred design, also known as usability engineering, is one of the most central methodologies and now widely used in various disciplines, including Software Engineering, Knowledge Management or Information System [15].

One of the objectives of the CO-ODE project is to provide a user-oriented tool set for the Protégé OWL Plug-in, so the user-centred design techniques will be kept in mind throughout the entire plug-in design life cycle.

A key aspect of user-centred design techniques is to make users involved in the software design process, by interviewing various groups of users based on certain requirements, such as age, occupation, gender, culture and so on. The interview results will be gathered and analysed in order to discover the goals and values of the target user group. The techniques of user participatory design are obligatory while designing this KE plug-in, because the target user group is mainly scientific researchers with different disciplines, requirements, personalities, ways of working and thinking.

User participant design includes observing and recording the manual activities, such as using the paper as window frames; cutting the paper into rectangles with difference size as dialogues and menus; choosing difference colours as different selection feedback; drawing, dragging while necessary to modify the interface; taking the picture while performing activities and many other actions. All these are performed by the **real** target group of users. The picture below was taken from the interview activities within the design process of this plug-in.

It is thereby necessary to set a predefined series of interviews in which we invite potential users to participate, and so that we can collect design information. Some interview methods such as unstructured interview and structured interview are going to be employed for different purposes.

An unstructured interview usually tends to be used in the early stages of the interview session, in which the users will be asked some general questions. In this plug-in

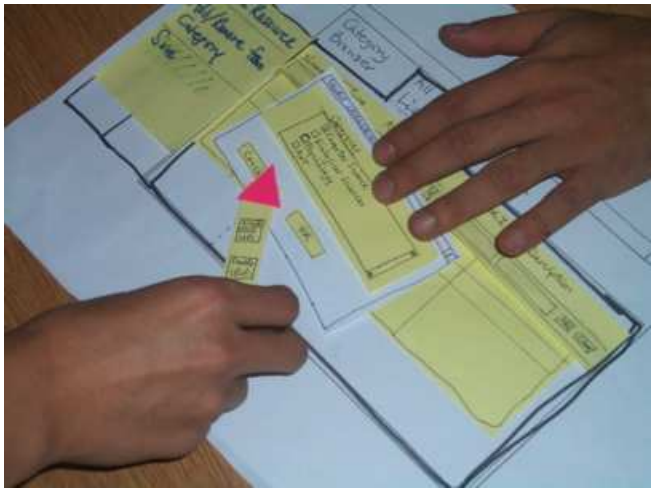

Fig. 4. A user participant design case

design, at first, we need to know the user's general points of view about the card sorting and laddering tools, the user's attitudes towards the perspectives of this plug-in and perhaps, and their personal manners of using computer software. Unstructured interview results will provide the developers with appropriate concepts and sensible ways of thinking, rather than the technical details.

Comparatively, it is much easier to hold an interview with a list of predefined questions. The structured interview design is more important for the software designers because all the interviewees will be asked a same set of questions related to the software technical details. The analysis of the structured interview results are crucial since the detailed technical issues in the software design phase will be addressed based mainly on these results.

4.2 Case Study of Interviews

The interview results show remarkable differences between people with different academic backgrounds, however, it also shows that the age, gender, and cultural background don't play essential roles. Probably that's because of the statistical analysis requires a much bigger sample, but we have already collected enough information required for the design of this KE plug-in.

Learning from the interview results, this software system should have 1) a input from document and elicitation functionality on user interface; 2) a series of cards generated from the text with round rectangle and the colour style of Protégé, that's because using the colour style of an existing popular base system tends to be more acceptable; 3) a flexible and straightforward user interface with layout of placing the working panel – both the card sorting and laddering tool, at the left as tabbed widgets, and putting the operation results on the right, as well as a number of buttons reasonably arranged; 4) a well-formatted output.

4.3 Implementation

Now we can conclude the procedure of building this plug-in with user-centred design techniques involved, which can be displayed in Figure 5 in a straightforward manner.

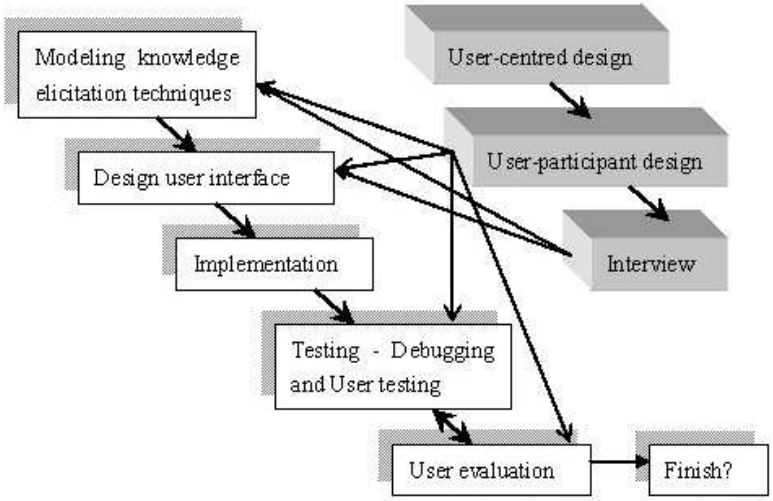

Fig. 5. Whole picture of design and implementation

From Figure 5 we can find that the interview sessions should be held while interviewees are performing and modeling the KE techniques – basically, the card sorting and laddering techniques, and the design of the user interface, i.e. the 3D rectangle objects are issues related to the HCI area. User participant design methods are also applied in the testing and evaluation sessions. We can see from the figure that there might be some loops between the testing/debugging and user evaluation circle, which is because of the fundamental software engineering rule – developers never know when the project will be finished and what kind of extensions should be added. It depends on the evaluation results and up-to-date user requirements.

5 Application of the Plug-in

We first built a prototype with a focus on application input/output aspects, then the prototype was extended to the existing Protégé system. Protégé quite naturally was our first choice as underlying infrastructure due to its widespread adoption and its easy extensibility. There already exists a plethora of freely available plug-ins that extend the basic functionalities.

5.1 Prototype

The KE methodology (cf. Figure 3) requires input and output, in which there are some trade-offs between the simplicity of the user interface and strength of functionality.

For the input, to show the most straightforward idea of this plug-in, we are going to use plain text as the source of concepts. There are many completed projects investigating how to use text mining and natural language processing (NLP) techniques to acquire knowledge from text, for instance, Text2Onto project [16]. So obviously, to search for a possible extension with existing tools is a better choice rather than developing a new one.

The format of the output is one of the most important design issues, because a primary consideration of this plug-in system is extendibility, which emphasises globally unified input/output. This software system might have many possibilities of input, thus we are going to discuss the output format here.

Basically, the proposed output file formats are: pure text, HTML and XML/RDF. Pure text is the most common way to store information, however, it may have different default format like ASCII or Unicode, while they are processed in different operating systems. HTML is a well-defined syntax-based mark-up language and easy to be parsed, but an HTML file is not easly machine-understandable. The Resource Description Framework (RDF) [17] is based on XML technology with machine readable format, and the processing of RDF is well-implemented by many third party programming language APIs.

As matters stand above, the most sensible choice is to use RDF for information storage in this software because of the consideration of feasibility, portability and acceptability. Another possible output is to use the existing Protégé components such as Protégé OWL Plug-in to directly transfer the output to the ontology tree for future development.

After making the decision about the input/output issues, we have the prototype for this plug-in as in Figure 6.

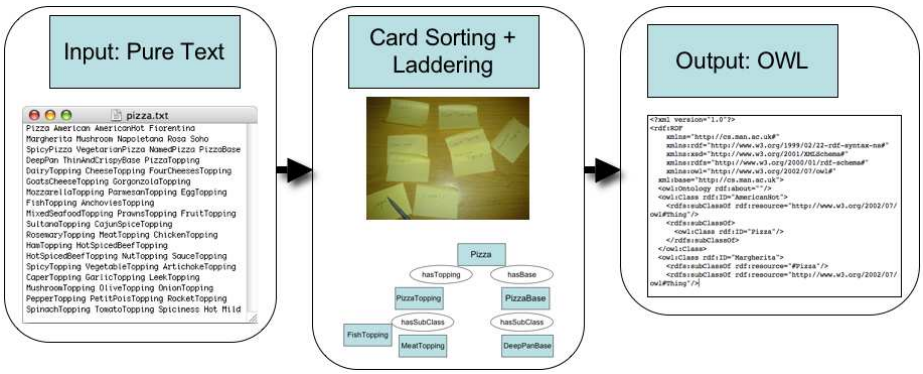

Fig. 6. The plug-in prototype

The starting point is we have a series of terms in a text format file. The users can initialise their ontologies by card sorting and laddering as tool tabs in the software interface at the middle, and then at last the users get the output as an RDF document. Thereafter by applying the prototype demonstrated, the plug-in can be developed in reality.

5.2 The Reality

Assume that users have a source of texts, in which there are a list of terms, this plug-in allows a user to elicit terms from pure text as Figure 7 shows. After the elicitation procedure, the users can build a subsumption relationship conceptual tree from the working panel, by adding, deleting and editing the cards, which is controlled overall by the "Version Manage", marked in Figure 7.

Essentially, this is a typical **card sorting** session. The two processes mentioned are combined together as the conceptual modeling for generating the subsumption relationship, depicted in Figure 7, respectively. The black colored numbers is to indicate the steps of operations. The blue arrows will show as dark gray if the paper is not colour-printed.

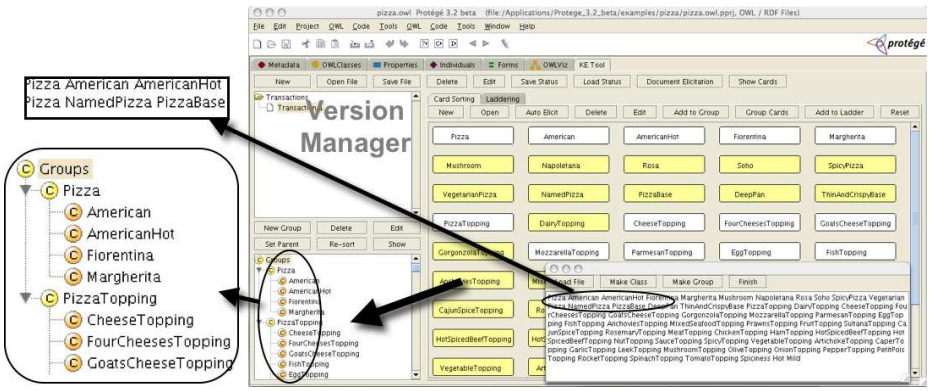

Fig. 7. Building subsumption tree

The prototype figure 6 has displayed the steps to elicit knowledge from text. A pure text window which locates at the right bottom of the figure is providing the knowledge resource. The users can load the text to the working panel that consists the cards to be sorted. Then the users can begin to sort the cards into different piles or groups that are displayed as the tree showed at the left of Figure 7 to get the subsumption tree as the very first ontology structure. Different levels of colors indicate the status of each card, illustrating whether the cards have been sorted (white) or not.

Card sorting panel gives users a clear and straightforward illustration of how the concepts are arranged and which concept has or has not been sorted. By using this component, the users are not going to be confused by the texts listed on the documents but sorting fast and correctly.

While the users have the skeleton of the ontology and they want to add properties between the concepts, the **laddering** tool provides a smooth route for this task. The detailed procedures are indicated at Figure 8. The card sorting window is not flowing at the right bottom, showing the procedure of how to add concepts from a series of cards to the ladder. The conceptual ladder with magnified view locates at the centre, and we can observe from the magnified ladder edition windows at the bottom that the relation between the current concepts "Pizza" and "PizzaTopping" is "hasTopping". The direction of the current ladder is "ladder down", therefore the domain of object property "hasTopping" is "Pizza", while class "PizzaTopping" is the range. In our integrated KE technique, we build object properties in this way.

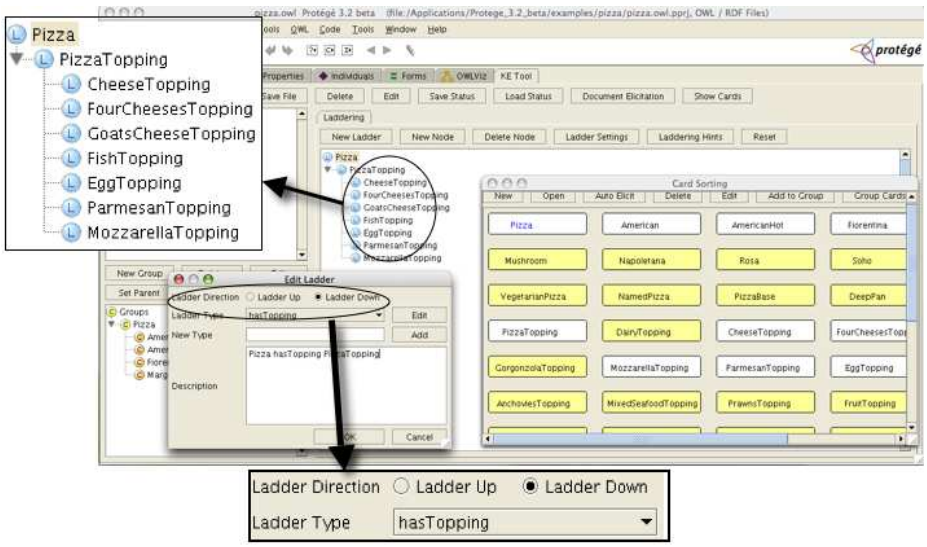

Fig. 8. Laddering

To control the card sorting and laddering tools overall, and to enable users to easily manage their milestones, a version manager function is implemented by saving and loading the runtime status of the plug-in, both the information of the laddering and subsumption structure, as well as the contents in card sorting tab, to and from the main memory.

We make use of this manager to organise the global actions performed by the users so that users are able to track back to their previous task runtime status by simply choosing and loading the different versions that are created and saved before. The users just need to choose a target version, press a "Save Status" button, and then the system will give a message to tell the users whether the version is successfully saved or not. Once the users come back from other versions, they can simply load this version into the working tabs and trees immediately by pressing the "Load Status" button. This

component lays on the upper right side of the interface, which can be seen from the screen shots in Figure 7 with label “Version Manager”.

6 User Evaluation

User evaluation shows the user’s attitudes towards the quality of this software. Based on the requirements of user-centred design, the feedbacks from user evaluation will be treated as an essential guideline for software testing and debugging.

We evaluated our approach by performing a user evaluation based on the requirements of the user-centred design. The user evaluation has two different parts. One is the interface evaluation which concerns the GUI, including ease of use, look and feel, and so on. And the other one is functional evaluation whose emphasises are the background functionalities. This evaluation methodology aims to detect the user’s comments on two basic aspects in the domain of user-centred design – the software should be powerful, flexible and robust.

6.1 Evaluation Result

There were eleven people involved in the user evaluation activities, and they are diverse in academic and cultural backgrounds. In order to quantify the result, a grading system similar to the university examination was borrowed (0-10 scoring scale), that is, 5 is a pass, 6 is a good pass, and above 7 is a distinction. In the arrays of the scores introduced below, the first five scores in each array come from the experts or frequent users of knowledge systems. The participants are marked with “E” for expert and “N” for “Non-expert” plus the reference number.

In terms of the user interface design, the grading result will be given to four different aspects as **interface evaluation**. The users were asked for the grades of the four points, and their grading results are listed in Table 1. To be statistically accurate, the average score was calculated by eliminating the highest and the lowest scores in each array.

Participant	E1	E2	E3	E4	E5	E6	N1	N2	N3	N4	N5	Average
Look and feel	9	7	7	8	9	7	7	6	7	8	6	7.3
Interface layout	7	7	9	6	8	6	9	5	6	5	6	7.3
Ease of use	7	7	6	7	8	7	6	6	6	6	5	6.4
Flexibility	7	8	8	6	5	6	6	6	9	6	8	6.8

Table 1. Interface evaluation results

The overall score was calculated by formula using standard deviation so we get **6.7** points here.

The **functional evaluation** involved the grading of each basic component, including card sorting, laddering, relationship setting and version manager. They are four major components provided by this plug-in and users are easily getting familiar with them, so the grading of these components is direct.

Participant	E1	E2	E3	E4	E5	E6	N1	N2	N3	N4	N5	Average
Card sorting	9	8	7	9	9	7	8	8	7	7	8	7.9
Laddering	7	6	7	7	7	7	6	7	7	9	8	7.0
Relationship setting	6	7	6	6	6	8	7	9	7	8	8	7.0
Version manager	9	9	8	8	9	8	7	8	9	8	9	8.4

Table 2. Functional evaluation results

We can see that the overall is **7.6** points. After taking the scores, we now analyse the results and make a conclusion.

6.2 Result Analysis

From the scores, we can simply find out that the users are mostly satisfied with the functionalities of the plug-in, which stands that the primary user-centred design procedure has been well-established. With respect to the interface of this plug-in, although the score is comparatively moderate, the users also generally have given positive comments.

To discover more from the evaluation results, we find that the interface look and feel, card sorting and version management components have the highest ratings and are thought to be the best implemented. Meanwhile, the elements related to the ease of use principle and interface layout arrangement require much future improvement.

If we go further, we may find that the plug-in interface are more appreciated by the experts rather than the amateurs, because the knowledge engineering experts are more familiar with the existing Protégé system, card sorting and laddering approaches. They find that this software have a unified style with the Protégé system, which doesn't quite make sense to the non-experts, though. Otherwise, contrarily the experts are not fully satisfied with the laddering tool and relationship setting component. Their feedback express the way of their working is somewhat different from how this plug-in does. That's because, people from different disciplines are likely to use laddering tool in many different ways for different purposes, and the plug-in is developed according to the design principles of CO-ODE project with strong emphasis in the medical and biological domain.

It is worthwhile to mention that the evaluation of the software from the wholly independent UK Freshwater Life Biological Association, and their comment on this software is:

"It was good to see what he has been doing and looks like a potentially very useful tool. We really liked to get our hands on a copy to play around with. Even in its current state it could save us considerable time."

In a nutshell, this plug-in is commonly considered to be a well-implemented and powerful tool in **real** use, whereas the interface is possibly only recognised by the knowledge system experts. All the evidences in this user evaluation procedure show that people are very eager to see the future development of this plug-in.

7 Conclusion and Outlook

We presented a technique for supporting the building of ontologies by using and integrating the knowledge elicitation techniques of card sorting and laddering. We developed a Protégé plug-in by employing user-centered design methods and thoroughly evaluated the research outcome. In the evaluation users performed very well with the plug-in and gave highly valuable feedback for future development. The conventional KE techniques were seamlessly integrated to the ontology building process to close the gap in the traditional manual ontology engineering cycle.

Future work includes the extension of the plug-in with some featured capabilities from the Text2Onto tool [16] to automatically extract terms from large texts using text mining and ontology learning techniques to further heuristically speed up the ontology building process.

Acknowledgements

The research reported in this paper was supported in part by the CO-ODE project (`http://www.co-ode.org/`) funded by the UK Joint Information Services Committee and the HyOntUse Project (GR/S44686) funded by the UK Engineering and Physical Science Research Council and by 21XS067A from the National Cancer Institute. The authors' current research is also supported by the EU-IST-506826 SEKT project (`http://www.sekt-project.com`). This publication only reflects the authors' views. We would like to thank the support from the potential users and our colleagues for fruitful discussions.

References

1. Upchurch, L., Rugg, G., Kitchenham, B.: Using card sorts to elicit web page quality attributes. IEEE Software **18** (2001) 84–89
2. Cooke, N.J.: Varieties of knowledge elicitation techniques. Int. J. Hum.-Comput. Stud. **41** (1994) 801–849
3. Hinkle, D.: The change of personal constructs from the viewpoint of a theory of construct implications. PhD thesis, Ohio State University (1965) Cited in: Bannister, D. and Fransella, F. (1980). Inquiring Man. Penguin, Harmondsworth.
4. Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., Musen, M.A.: Creating semantic web contents with protégé-2000. IEEE Intelligent Systems **16** (2001) 60–71
5. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The protégé owl plugin: An open development environment for semantic web applications. In: International Semantic Web Conference. (2004) 229–243
6. Boose, J.H.: Knowledge acquisition techniques and tools: Current research strategies and approaches. In: Proceedings of Fifth Generation Computer Systems. (1988) 1221–1235
7. Hoffman, R.R.: The problem of extracting the knowledge of experts from the perspective of experimental psychology. AI Magazine **8** (1987) 53–67
8. Shadbolt, N., Hara, K.O., Crow, L.: The experimental evaluation of knowledge acquisition techniques and methods: history, problems and new directions. International Journal of Human-Computer Studies **51** (1999) 729–755

9. Rugg, G., Eva, M., Mahmood, A., Rehman, N., Andrews, S., Davies, S.: Eliciting information about organizational culture via laddering. Journal of Information System **12** (2002) 215–230
10. Milton, N.: PCPACK Toolkit. (2003) www.epistemics.co.uk/Notes/55-0-0.htm.
11. Schreiber, G., Wielinga, B.J., Akkermans, H., de Velde, W.V., Anjewierden, A.: CML: The CommonKADS conceptual modelling language. In: Proceedings of 8th European Knowledge Acquisition Workshop (EKAW). (1994) 1–25
12. López, M.F., Gómez-Pérez, A., Sierra, J.P., Sierra, A.P.: Building a chemical ontology using methontology and the ontology design environment. IEEE Intelligent Systems **14** (1999) 37–46
13. Sure, Y., Staab, S., Studer, R.: On-to-knowledge methodology. In Staab, S., (eds.), R.S., eds.: Handbook on Ontologies. Series on Handbooks in Information Systems. Springer (2003) 117–132
14. Tempich, C., Pinto, H.S., Sure, Y., Staab, S.: An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In Gmez-Prez, A., Euzenat, J., eds.: 2nd European Semantic Web Conference (ESWC 2005. Volume 3532 of LNCS., Heraklion, Crete, Greece, Springer (2005) 241–256
15. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997)
16. Cimiano, P., Völker, J.: Text2onto – a framework for ontology learning and data-driven change discovery. In: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05). (2005)
17. Lassila, O., Swick, R.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation,World Wide Web Consortium, Boston. (1999) www.w3.org/TR/REC-rdf-syntax (current 6 Dec. 2000).