Beyond Rigid KA Metatools: An Initial Proposal for a KR-Independent , Adaptive, Customizable, Ontology-based KA Metatool

Yuangui lei, Enrico Motta and John Domingue
Knowledge Media Institute
The Open University
Walton Hall, MK7 6AA
{y.lei, e.motta, J.B.Domingue}@open.ac.uk

ABSTRACT This paper proposes a new knowledge-acquisition metatool, KRIACO, a Web-based, Knowledge Representation Independent, Adaptive, Customizable, Ontology-based Knowledge Acquisition Metatool, which aims to overcome sorts of shortcomings of current knowledge acquisition metatools. To achieve its goal, KRIACO will adopt OKBC as an underlying knowledge representation model to overcome the restriction caused by a specific knowledge representation system. It will use an ontology-driven tool specification approach to specify the domain and the task models; an interface ontology and a declarative interface model to drive the interface design and customization process; adaptive approach to provide flexible, adaptive user interfaces. This paper discusses these approaches in detail and presents the framework of KRIACO.

KEYWORDS: knowledge acquisition metatool, interface ontology, declarative interface model, adaptive user interfaces, user profile, interface development environment

1. INTRODUCTION

Metalevel tools can support the software development process by automating the design of task- and application-specific tools [Eriksson et al., 1994a]. The main idea is to generate knowledge acquisition tools from knowledge of either a domain, a problem solving method or a software architecture automatically rather than constructing knowledge acquisition tools from scratch, a costly and time consuming process.

Research in this area is getting mature, and several metatools have been developed and put into use successfully: Protégé-I [Musen, 1989]; SIS [Kawaguchi et al., 1991]; Spark [Marques et al., 1992]; DOTS [Eriksson, 1993]; DASH (PROTÉGÉ II) [Eriksson et al., 1994a]; Protégé2000 [Grosso et al.,1999] and KNOTES [Domingue & Motta, 2000] etc.

[Eriksson et al., 1994b] has identified three approaches in knowledge acquisition tools specification and generation.

- *The Method-oriented approach* uses the specification of a problem-solving method as the basis for the tool generation.
- *The architectural approach* uses an architectural model of the target tool as the basis for the tool specification.
- *The Ontology-based approach* generates a knowledge-acquisition tool from ontology by mapping ontology definitions to user interface elements

Protégé-I [Musen, 1989] and Spark [Marques et al., 1992] fall in the first category. DOTS [Eriksson, 1993] and SIS [Kawaguchi et al., 1991] have used the second approach. Protégé 2000 [Grosso et al., 1999], KNOTES [Domingue & Motta, 2000] and Protégé-II (DASH) [Eriksson et al., 1994a] have adopted the third method. Although these tools have achieved great success of providing effective tool support with a minimal tool development effort to generate knowledge-acquisition tools, there exist sorts of restrictions to limit their usability and reusability:

- *Knowledge Representation Restriction* -- The underlying knowledge representation language normally plays an important role in knowledge acquisition metatools. Its interoperability and compatibility with other modelling framework influence the usability and reusability of such tools greatly. All tools mentioned above suffer the knowledge representation limits except Protégé 2000 which adopts the OKBC [Chaudhri et al., 1998] knowledge model as its underlying knowledge model. Other tools work only with a single specific knowledge representation system. For example, KNOTES [Domingue and Motta, 2000] developed in OCML [Motta, 1999] can't be used to generate knowledge acquisition tools for ontology which is not compatible with OCML.
- *Tool Specification Approach Restriction* – The tool specification approach influences the usability and reusability of metatools greatly. The drawback of traditional method-specific approaches is that the metatool is coupled to a specific problem-solving method [Eriksson et al., 1994c]. Architectural specification approach is

more generic, but usefully in a problem, given that they require detailed information about the target interface. Protégé-I [Musen, 1989] and Spark [Marques et al., 1992] use a method-oriented approach to specify the target tools. DOTS [Eriksson, 1993] and SIS [Kawaguchi et al., 1991] uses the architectural approach.

- *Lack of Interface Adaptability* – Some knowledge acquisition metatools provide customization facilities for users to customize interfaces such as Protégé 2000 [Grosso et al., 1999] and Protégé II [Eriksson et al., 1994a]. But none has been reported to provide support of adapting interfaces to users' tasks and their preferences according to their interaction history with systems. On one hand, different users may need to acquire different knowledge aspects during constructing a knowledge base, especially when the knowledge base is in large scale. So it would be more effective if a system can hide or fold interfaces which are irrelevant with a user's task, and then help the user to concentrate on his or her task. The user's task can be obtained dynamically by analyzing his or her interaction history with the system. On the other hand, the interface customization is a non-trivial task. Users may not want to customize every interface manually. But these knowledge acquisition metatools don't have facilities to help users to relieve the burden.

To overcome above shortcomings, this paper proposes a new kind of knowledge-acquisition metatool KRIACO, a web-based, knowledge representation independent, adaptive ontology-based knowledge acquisition metatool. Comparing to existing metatools, its strength lies in four following aspects:

- ***Web-based*** – KRIACO will be distributed on WWW, so it possesses the cross-platform feature.
- ***Knowledge Representation Independent*** – To overcome the restriction caused by specific knowledge representation systems, KRIACO will adopt OKBC [Chaudhri et al., 1998] as underlying knowledge representation model. OKBC (Open Knowledge Base Connectivity) is an application programming interface for Knowledge Representation Systems. Knowledge Base tools can manipulate most of Knowledge Representation models through OKBC. By using OKBC model, KRIACO can be used on different Knowledge Representation Systems. It will not be limited to a specific Knowledge Representation System.
- ***Adaptive and customizable*** – According to tasks of knowledge acquisition metatools, user-adaptive systems can be much more effective than non-adaptive systems. KRIACO will provide adaptive facilities to help users to tailor interfaces automatically and concentrate their focuses on their specific knowledge acquisition tasks. Customization provides tools for users to set up their preferable interfaces. Adaptive and customizable features will distinguish KRIACO from current knowledge acquisition metatools. It will be more effective and easier to use than current metatools.
- ***Ontology-based*** – KRIACO is a knowledge acquisition metatool based on application ontology and interface ontology. An Application ontology specifies a domain model and a task model. Interface ontology defines conceptions, prototype and guidelines of interface development. The using of both ontologies will enable KRIACO to be much more reusable than any other metatools.

The rest of this paper is organized as follows: Section 2 describes approaches that will be used on developing KRIACO. Section 3 introduces the framework of KRIACO. Section 4 describes key issues in developing KRIACO.

2. APPROACHES

To achieve its goals, KRIACO will use following approaches:

Adopting Open Knowledge Representation model as its underlying knowledge model. OKBC [Chaudhri et al., 1998] is a kind of open knowledge representation model. It serves as a generic access and manipulation layer for knowledge representation systems. Knowledge Base Tools can manipulate different knowledge representation models through OKBC. Using OKBC as its underlying knowledge model, KRIACO will realize its independence of any specific knowledge representation system as much as possible.

Ontology-driven specification approach to mapping the interface from ontology. This approach has been used in Protégé-II [Eriksson et al., 1994a], Protégé-2000 [Grosso et al., 1999] and KNOTES [Domingue & Motta, 2000] successfully. The basic idea for tool generation from ontologies is that a metatool can map the data types of slots in a class

definition to user-interface components of the knowledge-acquisition tool [Eriksson et al., 1994c]. The main advantage of this approach is that the metatool developed for one domain can easily be used on another domain. And domain experts and users can generate knowledge acquisition tools with no demand of knowledge engineers involving. Although the basic idea of this approach is relatively simple, there are many fundamentals and technical problems that must be addressed before generating knowledge-acquisition tools.

- Analysing ontology, identifying class definition and class relationship
- Mapping interface structure from ontology class definition which determines the navigation method of knowledge browsing, editing and maintaining. The metatool must analyse the ontology and then create a high-level design of the target tool.
- Generating details of each graphic form from slots of each class. In DASH [Eriksson et al., 1994a], the first mapping problem is solved by dialogue structure designer which creates an abstract description of the editors in the knowledge-acquisition tool and their relationships. The layout designer solves the second mapping problem.

Interface ontology or interface model to drive the interface design process [Puerta et al., 1994]. This approach borrows the idea of using a declarative interface model to drive the interface development process from the model-based interface development approach. In which, interfaces are automatically generated from a declarative specification that describes the tasks users need to perform, the content, the structure and layout of displays, and the role that display elements play in user's tasks [Szekely et al., 1995].

Literature review on model-based interface development has been undertaken in [Puerta and Szekely, 1994] and [Szekely, 1996]. Although the research on this area is getting mature, none such software environment has proposed and defined interface ontology explicitly.

Though interface ontology is a relatively new concept, the idea has been used in some model-based interface development environments implicitly. HUMONOID has presentation templates to model the characteristics of display elements [Szekely et al., 1993]. MASTERMIND has a model library which contains a wide variety of presentation [Castells et al., 1997]. MOBI-D uses additional knowledge bases about design guidelines and principles to operate on the interface model or to advise the developer [Puerta, 1997].

Interface ontology defines the basic and sharable conceptions of relevant knowledge of interface design and development. It serves as an interface knowledge base to provides data presentation guidelines for interface generation. For example, Boolean data type can have presentation of check-box widget or radio-button widget. The advantages of using this approach are:

- It conceptualises interface design knowledge, provide explicit interface knowledge base for interface generation.
- It helps to standardise interface generation and customisation process.
- It maximises the reusability of KRIACO.
- The declarative interface model which is generated by instantiating interface ontology gives end users maximum flexibility to customise interface.

Adaptive interface approach to tailoring user interfaces to users' needs and preferences. This approach has been used in intelligent tutoring systems and adaptive education systems successfully. Several web-based adaptive systems have been developed and used, such as AHA [Bra and Calvi, 1998], AHM [Silva et al., 1998], TANGOW [Carro et al., 1999] and ADAPT [Brusilovsky and Cooper, 1999] etc. At the same time numerous tool systems emerged during the last few years that aim at assisting companies in developing and deploying personalized web sites [Fink and Koba, 2000]. The idea of this approach is exploring a user model to adapt interfaces. The user model is generated and updated dynamically during the process of a user's interaction with software systems. It records a user's information including knowledge background, tasks and preferences etc. There are two major benefits of using this approach to develop KRIACO:

(1) This approach will relieve the heavy burden of customizing a number of interfaces of knowledge acquisition tools from the end user. It is a non-trivial task for users to customize every interface of knowledge acquisition tools to their needs and preferences. This approach can record a user profile about a user's preferences and needs during his or her customization processes, and then system can adapt other interfaces automatically based on the user profile.

(2) This approach will help a user to concentrate on his tasks. Besides user's

preferences and needs, the user model can also record user's task during his knowledge acquiring process. After several times interaction, system will assume the user's main tasks, and hiding interfaces which are irrelevant to these tasks.

Although the idea of using a user model to adapt interfaces in KRIACO is the same with existing adaptive systems, the practical problems are quite different. First, the user model is different. Adaptive systems in ITS and AES mainly record a user's knowledge level about concepts or subjects during user's learning process. In KRIACO, the user model will contain a user's preferences recorded during the user's customization processes, and a user's tasks acquired by monitoring his or her knowledge acquisition processes. Second, adaptive technology is quite different. Most of existing web-based adaptive systems narrowly focus on the web-page content adaptation and the navigation adaptation. But KRIACO's approach targets on software application adaptation. Research on software application adaptation and user modelling has been undertaken for many years. UIDE [Sukaviriya and Foley, 1993] supports adaptive interface and adaptive help. MOBI-D [Puerta, 1997] applies an adaptive algorithm to automated user interface design within its framework. User modelling has been reviewed in [Kobsa, 2000]. Thus, KRIACO adaptive interface approach will combine the traditional software application adaptation approach, traditional user modelling approach and current web-based adaptive systems approach to achieve its target.

Component-Based Software Development Approach -- The idea is to create reusable software components which can be plugged in the applications dynamically and flexibly by using Java and JavaBeans OO programming technology.

3. FRAMEWORK OF KRIACO

According to [Puerta et al, 1994], the necessary tasks of knowledge acquisition tools need to possess are:

- Visaulization and browsing of existing knowledge. Users must have some means of browsing knowledge, both ontologies and instances.
- Editing and review of knowledge. The maintenance function of knowledge base is crucial to knowledge acquisition tool. Users can input, edit, delete knowledge (class instantiation) very easily.
- Search and retrieval of specific parts of the knowledge base. Knowledge acquisition systems must allow users to search and retrieve information stored in the knowledge base.

Like other metatools, the aim of KRIACO is to generate knowledge acquisition tools possessing above functions. Its approach outlines the framework of KRIACO (Figure 3.1).

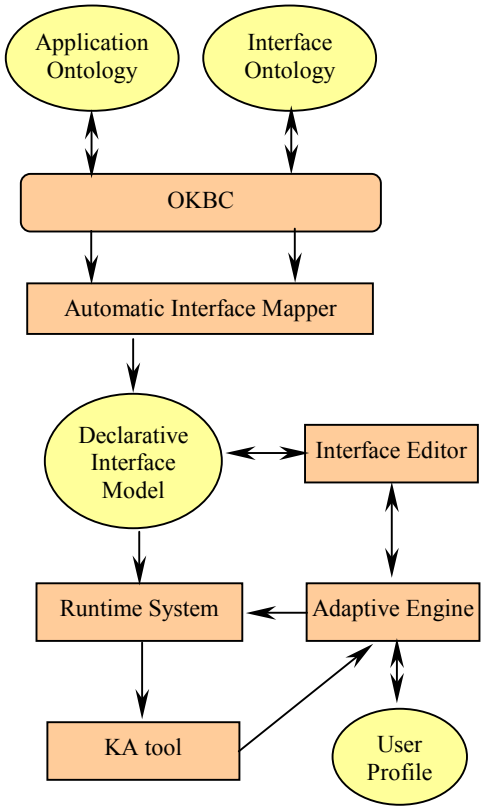

Figure 3.1 Framework of KRIACO

Application Ontology represents domain ontology and problem solving ontology. It serves as input of Automatic Interface Mapper through OKBC to specify domain and task model for KRIACO.

Interface Ontology defines user interface concepts and design guidelines including widget prototypes, form prototypes, navigation structure concepts and data-widgets mapping rules. It performs as an interface mapping knowledge base for Automatic Interface Mapper. The high level of hierarchy of the interface ontology can be shown in Figure 3.2. The top hierarchy of the interface ontology contains navigation class, form class, widget class and widget guidelines:

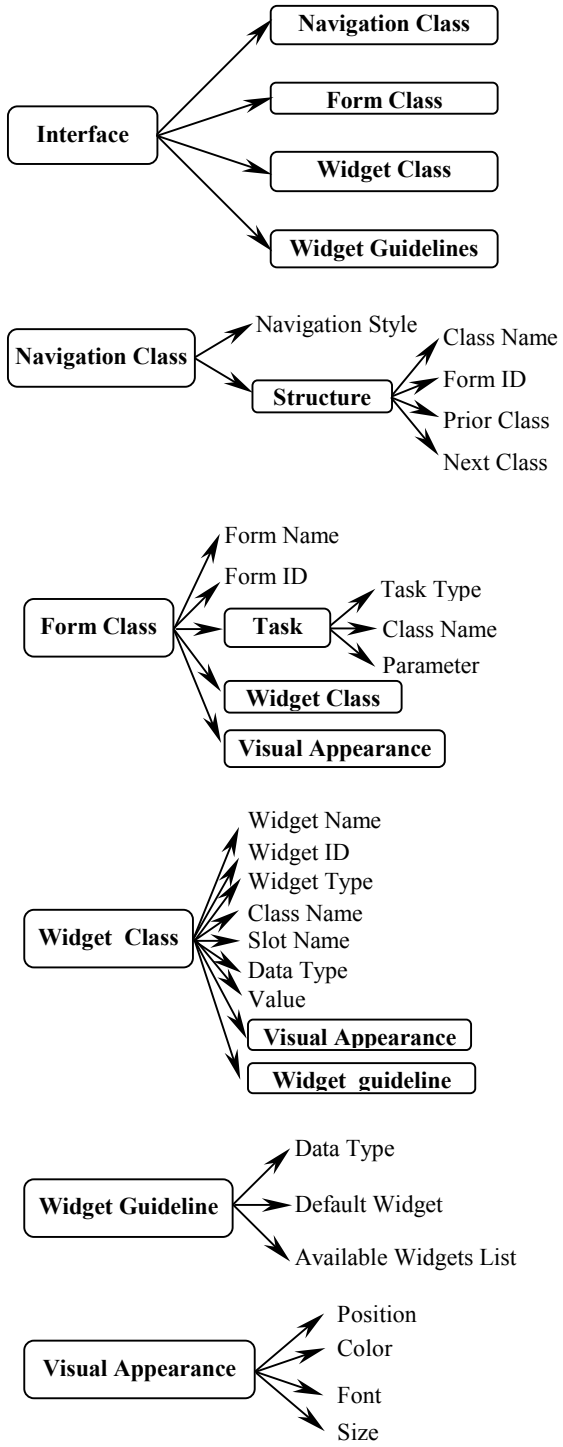

Figure 3.2 High Level Hierarchy of Interface Ontology

- *Navigation class* models knowledge navigation structure in knowledge acquisition tools. Application ontology is made up of lots of classes with different hierarchy structure, users need to navigate from one class to another. So navigation structure is backbone of knowledge acquisition tools.
- *Form class* models form specifications including form task, appearance and its widgets. Form tasks contain input/edit/search knowledge. Form appearance defines the visual appearance.
- *Widget class* defines widget elements in form.
- *Widget guideline* models widget mapping guidelines from data type. It provides information for the system mapping widgets from slots.

A Declarative user interface model is an expressive user interface specification. It is actually an instantiation of the interface ontology. This model works as input of the Runtime System and the Interface Editor. Users can modify this model by manipulating the interface customization tool, the Interface Editor.

A User profile records a user's personal information such as name, password etc., user preferences, tasks and interaction history. It is updated every time the same user interacts with the system. It provides information for Adaptive Engine to perform adaptive analyses and actions.

The Automatic Interface Mapper aims to generate user interface model from application ontology and interface ontology automatically. It has two components: *knowledge navigation mapper* which generates knowledge navigation structure from application ontology, and l*ayout mapper* which is responsible for mapping widgets and calculating layout from application ontology slots and interface ontology.

The Runtime System is responsible for generating a Knowledge Acquisition tool from a declarative interface model. It performs as the entrance of KRIACO. When KRIACO is run for the first time, it will turn to Automatic Interface Mapper automatically. Otherwise the Runtime System will generate a KA tool from the declarative interface model.

The KA tool is the target tool KRIACO need to generate. Besides the knowledge acquisition tasks it should complete, it also need to support following activities: (1) provide facilities to record and update user profile dynamically. (2) receive information from the Adaptive Engine and perform adaptive actions. (3) provide support for users to convert to the Interface Editor.

The Interface Editor provides support for end users to customize the KA tool interfaces. Furthermore, it receives information from the

Adaptive Engine and performs most of customization works for users automatically.

The Adaptive Engine performs to monitor users' interactions with the KRIACO, record information to user profile, analyse user profile and send adaptive information to the KA tool or the Interface Editor. The adaptive information sending to the KA tool includes a user's tasks and relevant information. The adaptive information sending to the Interface Editor mainly contains a user's preferences about interfaces.

KRIACO is an integrated development system. It integrates the Automatic Interface Mapper, the Interface Editor, the Adaptive Engine and the Runtime System together. The whole system including application ontology, interface ontology, declarative user interface model, user profile and all applications are stored on a server.

There are three scenarios for users to start KRIACO:

1. The KRIACO is used for the first time. There doesn't exist any declarative interface model and user profile. When user start KRIACO, the Runtime System will start the Automatic Interface Mapper, generate a default declarative interface model, and then it will generate a KA tool. At the same time, the adaptive engine will create a new user profile for the current user to record user information.
2. The KRIACO is not used for the first time. But the user is new. The KRIACO Runtime System will generate a KA tool automatically from the default interface model which is created at the first time when KRIACO is run. At the same time, the adaptive engine will create a new user profile to record the user information.
3. An old user starts KRIACO. First, the Runtime System will search the user's declarative interface model. At the same time, the Adaptive Engine will analyse the current user's profile and provide adaptive information to the Runtime System. Then the Runtime System will generate a KA tool.

4. CONCLUSIONS

KRIACO is a new knowledge acquisition metatool proposed to address shortcomings in current knowledge acquisition metatools. It will explore following approaches to achieve its goal:

- Adopting OKBC as its underlying knowledge model will enable KRIACO to be independent of any specific knowledge representation system.
- Ontology-driven tool specification approach gives KRIACO maximum independence from application ontology. KRIACO can be used on different domain easily.
- Sharing interface ontology approach gives end user maximum flexibility to customize user interfaces.
- Adaptive interface approach will make KRIACO much easier to use than current knowledge acquisition tools. First it helps users to customize interfaces automatically during knowledge acquisition tools construction process. Second, it will help users to concentrate their focuses on their main tasks by hiding or folding irrelevant interfaces.

Above all, KRIACO is proposed to be much easier to use, more generic and provide more support for end users than current knowledge acquisition metatools. It will involve study on area of software specification, interface development, open knowledge representation model, adaptive interfaces and user profile. The key issues in developing KRIACO involves:

- Developing user interface ontology and guidelines from literature.
- Developing automatic interface mapper tool for mapping user interface from interface ontology and application ontology
- Developing interface editor tool for editing user interface by end users.
- Developing efficient adaptive engine to adapt user interfaces.
- Developing runtime system for generating knowledge acquisition tools

REFERENCES

[Bra and Calvi, 1998] Paul De Bra, Licia Calvi, AHA! An open Adaptive Hypermedia Architecture, *The New Review of Hypermedia and Multimedia, Vol 4, pp 115-139, 1998.*

[Brusilovsky et al., 1996] Brusilovsky, P., Schwarz, E., and Weber, G, A Tool for Developing Adaptive Electronic Textbooks on WWW. *In Proceedings of WebNet'96 - World Conference of the Web Society, October 16-19, 1996. San Francisco, CA, AACE. - pp. 64-69.*

[Brusilovsky & Cooper, 1999] Peter Brusilovsky, and David W. Cooper, ADAPTS: Adaptive hypermedia for a Web-based performance support system, *In Proceedings of the 2nd Workshop on Adaptive Systems and User Modellng on the WWW, 1999*.

[Carro et al., 1999] Rosa María Carro, Estrella Pulido, and Pilar Rodríguez, TANGOW: Task-based Adaptive learNer

Guidance On the WWW, *In Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW.*

[Castells et al., 1997] Pablo Castells, Pedro Szekely, and Ewald Salcher, Declarative Models of Presentation, *In proceedings of the 1997 International Conference on Intelligent User Interfaces, January 6-9, 1997, Orlando, FL USA, PP 137-144.*

[Chaudhri et al., 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. R ice, OKBC: A Programmatic Foundation for Knowledge Base Interoperability, *In Proceedings of AAAI-98, July 26-30,1998, pp. 600-607*

[Domingue and Motta, 2000] John Domingue and Enrico Motta, PlanetOnto: From News Publishing to Integrated Knowledge Management Support, *Intelligent system, MAY/JUNE, 2000, P 26-32*

[Eriksson, 1993] H.Eriksson, Specification and Generation of Custom-Tailored Knowledge-Acquisition Tools. July 1993, *http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-93-0490.html*

[Eriksson et al., 1994a] Henrik Eriksson, Angel R. Puerta, and Mark A. Musen, Generation of Knowledge-Acquisition Tools from Domain Ontologies. *Int. J. Human-Computer Studies (1994) 41, 425-453*

[Eriksson et al., 1994b] Henrik Eriksson, Angel R. Puerta, Joh H. Gennari, Thomas E.Rothenfluh, Samson W. Tu, and Mark A. Musen, Custom-Tailored Development Tool for Knowledge-Based Systems, *Knowledge Systems Laboratory, Medical Computer Science, January 1994.*

[Eriksson et al., 1994c] Henrik Eriksson, Angel R. Puerta, and Mark A. Musen, Generation of Knowledge-Acquisition Tools from Domain Ontologies. *Int. J. Human-Computer Studies (1994) 41, 425-453*

[Fink and Koba, 2000] Josef Fink and Alfred Koba, A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web, *User Modeling and User-Adapted Interaction 10: 209-249, 2000.*

[Grosso et al., 1999] William E.Grosso, Henrick Eriksson. Ray W. Fergerson, Johh H. Gennari, Samson W. Tu, and Mark A. Musen, Knowledge Modelling at the Millennium, In *Proc. the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99) Banff, Canada, October 1999, http://smiweb. stanford.edu/pubs/SMI_Abstracts/SMI-1999-0801.html.*

[Kawaguchi et al., 1991] Atuso Kawaguchi, Hiroshi Motoda, and Riichiro Mizoguchi. Interview-based knowledge acquisition using dynamic analysis. IEEE Expert, 6(5) pp. 47-60, 1991

[Kobsa, 2000] Alfred Kobsa, Generic User Modeling Systems, *To be appear in User Modeling and User-Adapted Interaction, Ten Year Anniversary Issue, 2000.*

[Marques et al, 1992] David Marques, Geoffroy Dallemange, Georg Klinker, John McDermott, and David Tung. Easy programming; Empowering people to build their own applications, *IEEE Expert, 7(3) pp 16-29, 1992.*

[Motta, 1999] Motta E., Reusable Components of Knowledge Modelling: Case Studies in Parametric Design Problem Solving, *IOS Press, Amsterdam, 1999.*

[Motta et al., 2000] Enrico Motta, Simon Buchingham Shum, and John Domingue, Ontology-driven document enrichment: princilpes, tools and applications, Int. J. Human-Computer Studies (2000) 52, 107-1109.

[Musen, 1989] Mark A. Musen, Automated support for building and extending expert models. *Machine Learning, 4:349-377, 1989.*

[Puerta et al, 1994] Angel R. Puerta, Robert Neches, Henrik Eriksson, Pedro Szekely, Ping Luo, and Mark A. Musen, Toward Ontology-Based Frameworks for Knowledge-Acquisition Tools, *Proceedings of the 8TH Banff Knowledge Acquisition For Knowledge-Based Systems Workshop, shareable and resuable ontologies, shareable and reusable problem-solving methods,* Banff, Canada, January 30 – February 4, 1994, *pp26-1 – 26-14.*

[Puerta and Szekely, 1994] Angel Puerta and Pedro Szekely, Model-Based Interface Development, *In Proceedings of the CHI'94 Conference Companion on Human factors in Computing Systems, 1994, pp. 389-390.*

[Puerta, 1997] Angel R. Puerta, A model-Based Interface Development Environment, *IEEE Software 14(4): 40-47 (1997).*

[Peuerta & Eisentein, 1999] Angel Peuerta and Jacob Eisenstein, Toward a General Computational Framework for Model-Based Interface Development Systems, *Proceedings of the 1999 International Conference on Intelligent user Interfaces, 1999, pp. 171-178.*

[Silva et al., 1998] Denise Pilar da Silva, Rafaël Van Durm, Erik Duval, and Henk Olivié, Concepts and documents for adaptive educational hypermedia: a model and a prototype, *In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98, Pittsburgh, USA, June 20-24, 1998*

[Sukaviriya and Foley, 1993] Piyawadee "Noi" Sukaviriya, and James D. Foley, Supporting Adaptive Interfaces in a Knowledge-Based User Interface Environment, In P*roceedings of the International Workshop on Intelligent User Interfaces, 1993, pp. 107-113.*

[Szekely et al., 1993] Pedro Szekely, Ping Luo and Robert Neches, Beyond Interface Builders: Model-Based Interface Tools, *Proceedings on Human factors in Computing System, April, 1993, pp. 383-390.*

[Szekely et al., 1995] P.Szekely, P.Sukaviriya, P.Castells, J.Muthukumarasamy,and E.Salcher, Declarative interface models for user interface construction tools: the MASTERMIND approach, *In Proc. EHCI'95, 1995.*

[Szekely, 1996] Pedro Szekely, Retrospective and Challenges for Model-Based Interface Development, *In Proceedings CADUI'96, Computer-Aided Design of User Interfaces, Eurographics, Belgium, June, 1996.*