Wikifying your Interface: Facilitating Community-Based Interface Translation

M. Cameron Jones **Dinesh Rathi** **Michael B. Twidale**
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 E. Daniel St. Champaign, IL 61820
{mjones2, drathi, twidale}@uiuc.edu

ABSTRACT
We explore the application of a wiki-based technology and style of interaction to enabling the incremental translation of a collaborative application into a number of different languages, including variant English language interfaces better suited to the needs of particular user communities. The development work allows us to explore in more detail the design space of functionality and interfaces relating to tailoring, customization, personalization and localization, and the challenges of designing to support ongoing incremental contributions by members of different use communities.

Author Keywords
Internationalization, customization, personalization, wikis, mash-ups, community-based technology.

ACM Classification Keywords
H5 [Information interfaces and presentation]; User Interfaces. I.3.6 [Methodology and Techniques]; Interaction Techniques

INTRODUCTION
The design process often seems to be most successful when initially focusing on a small clearly defined set of users in a particular use context. Advocates for this approach can be found both in commercial software design, where the use of personas and scenarios have been particularly popular [5] and also within Open Source Software (OSS) development, where the initiation and extension of a successful project is often ascribed to 'scratching an itch' – a volunteer adapting a piece of software to meet a particular need and then sharing the result with others who may happen to have similar needs.

However it is often desirable to extend the use of software beyond the contextual niche of its genesis. This requires extending and adapting both the functionality and particularly the interface to fit more appropriately different users and use contexts. The challenge that this presents can be overwhelming, and in many circumstances it is both desirable and feasible to involve others, including end users in this adaptation process.

Adaptation can involve many different kinds of activities requiring different levels of skill and commitment. At one extreme one can have skilled programmers extending an application either as part of a commercial extension activity or participating in an OSS project. At the other, end users possessing minimal programming skill may select from a range of options to specify which features they want, and something about their arrangement and color. In this paper we examine one kind of adaptation, that of communities of users wishing or needing to use an interface in a particular language, most likely their native language. We explore the challenge of interface translation in the light of existing work on adaptation, tailoring and customization because our proposed solution involves a collective incremental approach whereby an interface is progressively translated by community members themselves, rather than provided as a monolithic solution by developers.

In this paper we look particularly at the process of language access and how community-oriented methods might help. However, the approach carries implications for many other kinds of adaptation activities. Translation is just one component of a larger issue of universal usability [30]. A system is very hard to use if the user is unsure what the words of its interface mean. This obviously applies if the interface is not in the user's native language, but is also true in cases where the language used is esoteric or ambiguous. We hope that the approach to community based translation explored in this work can be extended to other aspects of universal usability.

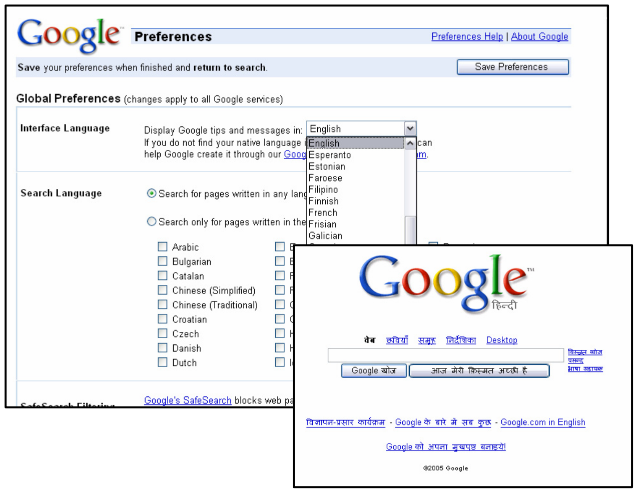
No single interface is usable by all. Interfaces which may serve the needs of the group often are not good for individuals [9] or vice-versa. Currently, interfaces are developed according to the former approach on the principle that "*one-size-fits-all*" [32] and thus often do not meet the needs of diverse individuals or communities. Often this decision is dictated by practical limitations to taking the latter approach (cost, time, etc); however Weld et al. argue that each individual "deserves a personalized interface” [32]. The only viable solution available is to build facilities which empower users themselves to personalize the interface for their particular needs [32]. Our interpretation of personalization is congruous with [3] who defines personalization as “a process that changes the functionality, interface, information content, or distinctiveness of a system to increase its personal relevance to an individual”.

The facilities for supporting personalization are defined within the broader field of tailoring. Tailoring, the adaptation of software to specific work practices, spans a range of activities from customization to integration to extension [22]. We will focus primarily on customization, defined as “any capability that makes generic programs suitable to a specific user need, for example, templates, automated activities like macros, etc.”. Customization researchers have discussed the limitations of the system based on the available items in menus or toolbars or by macros [32]. Tailoring, and likewise customization, is an activity which “takes place after the original design and implementation of the application” [22].

Customization usually concentrates on the user interface, allowing users to adapt the presentation of the system and its data in order to better fit the system to the user’s context. This is commonly achieved by giving users control over the selection of templates, definition of macros, or customization of toolbars. However, these methods are inadequate, not allowing for the degree of personalization prescribed by Weld et al. [32]. The restriction to “predefined configurations” not only limits users in their choices, but also requires designers to envisage the possible use scenarios, something which is difficult, if not impossible given the broad diversity of users of widely disseminated applications.

These mechanisms may well be good enough for applications designed for limited user audiences, but what about applications which are to be used by anyone on the Internet? Enabling personalization of language is fundamental to building applications for a global audience. Hence, the user interface should be flexible enough to “handle text manipulation necessary for translation” [26]. Additionally, interfaces should allow users not only to select from predefined language options, but also to add and modify languages, and create a translation of the interface which is truly personalized. Our current work is focused on developing methods and practices for supporting this degree of openness and flexibility in the translation and personalization of web-based collaborative tools.

The language of an interface is important for cultivating an expanding international user base, creating a higher comfort level among non-native speakers of the original language (typically English) [27]. Generally speaking, there are three methods for providing interface translations: automated machine translation, human translation, and community-based translation [27, 21]. Automated machine translation



Save your preferences when finished and return to search.

Global Preferences (changes apply to all Google services)

Display Google tips and messages in: English

If you do not find your native language … can help Google create it through our Goog… m.

Search for pages written in any lang…

Search only for pages written in the…

Google's SafeSearch blocks web pa…

Figure 1. Screenshots of the Google language options and an example of the Google search interface in Hindi

has met with limited success; not only is translation quality inferior, but users normally do not have the ability to make changes in the translated text [21]. However, machine translation is far less costly in time and effort than other methods, and cheap, rapid, imperfect translations may well be good enough for certain purposes. McDevitt, et al. note that AltaVista’s Babel Fish is used for translating web page content cheaply and presumably somewhat adequately [21]. The other approaches, human- and community-based translation, require humans to manually translate the interface, producing higher quality translations [27]. It is worth noting that in both methods, translation is done as part of the development process and participation is typically limited to paid human translators working with the developers, or willing volunteers for community-based translation. In the case of Open Source Software Usability, despite its problems [24], software localization has been a notable success, with volunteers creating multiple language versions for a variety of applications, including Firefox, Gnome, and KDE. In all these cases however, the number of volunteers is very small and restricted to the technically adept.

Google™ is an example of a community-based translation system [21]. The Google search engine offers an impressive 150 different language options (see Figure 1). The community-based translation in Google is coordinated through the “Google in Your Language” program where users volunteer to become translators [15]. The set of available languages, large though it is, is still limited in the sense that users wishing to volunteer must select from one of the existing 150 language choices (although 61 are reported as 100% complete and 88 vary from 0-99% complete [15]). Once registered, volunteers are then able to edit the translations made by other users, but must go through an additional authorization before they are able to translate new elements.

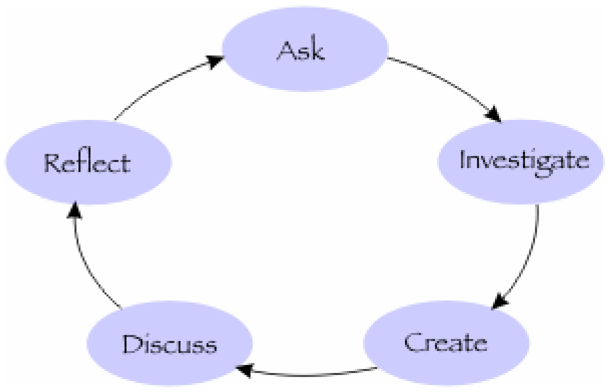


Figure 2. The Inquiry Cycle from the Inquiry Page

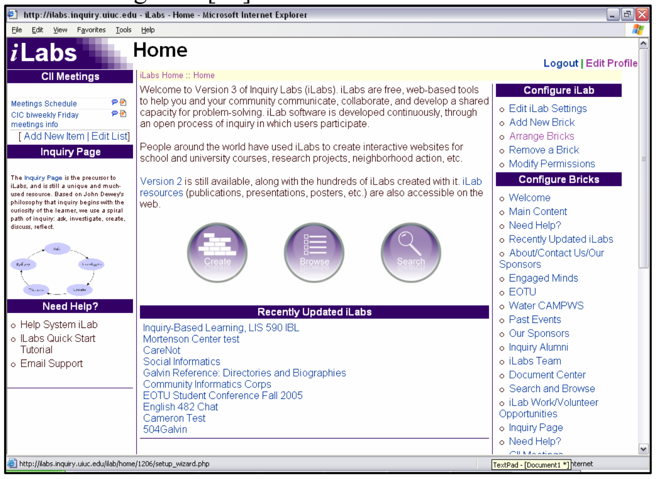
The Google translation system and the other translation methods discussed thus far fail to account for what Fraser calls the "social and cultural information in community languages" [8]. Community languages are minority languages and variants spoken within a majority language context. This idea was originally developed in the multi-lingual society of Great Britain to characterize minority immigrant communities in relation to the majority English speaking culture. However, the concept can be applied more generally to describe variation among communities superficially speaking the "same" language.

Take, for example, the Hindi language translation shown in Figure 1. Google treats Hindi as a single language to be consumed by all Hindi speakers equally. However, consider the more than ten variations of Hindi each with numerous dialectical variations spoken in different regions of India. Each of these communities will be speaking variations of their "native" Hindi language which will vary in terms of vocabulary and meaning and will reflect the local communities' particular culture. Even individuals originating from the same region of India but residing in the US or the UK will have variations in their language which will reflect the English majority culture where they are embedded (not to mention the differences between US and UK English). Google has attempted to account for the most significant variations; adding a Bihari--Angika (a variant of Hindi) translation. However, within Bihari there are several dialects other than Angika including Bhojupuri, Maithili and Magahi. All of these can still be considered "Hindi". Di Biase ([7] as quoted in [8]) emphasizes the importance of recognizing these variations with respect to translation stating that "the translator needs to know the community he or she is translating for and to become acquainted with its linguistic, social and cultural characteristics in terms of both the origins of the community and its new environment".

When language is seen in this hyper-localized context, the community-based translation methods [21, 23, 25, 27, & 29] do not seem adequate. Although purporting to be "community-based" they do not, in reality allow the community to translate the interface. These approaches rely on select volunteers from the "community" to do the translation. The problem is that here, "community" refers to the broad context of national languages, ignoring the intra-community language variations. The Hindi translation Google offers, is Hindi, but whose Hindi? The volunteers performing the translation bring the cultural bias from the community of which they are a member. Inevitably, there will be concepts in the interface which do not translate across the community boundaries. This issue becomes even more acute when considering provisions for minority languages [24].

The work presented in this paper attempts to resolve these issues of community translation by adhering to the principles of community inquiry theory. Community inquiry theory is rooted in the philosophy of the American pragmatists, rising to prominence at the end of the 19th century. Developed most fully in the work of John Dewey, community inquiry is based on the premise that if individuals are to understand and create solutions for problems in complex systems, they need opportunities to engage with challenging problems, to learn through participative investigations, to have supportive, situated experiences, and to articulate their ideas to others [1]. Learning and knowledge construction happens through an iterative process of inquiry as typified by the Inquiry Cycle from the Inquiry Page (Figure 2). The Community Inquiry Labs (iLabs, see Figure 3) tools suite aims to embody and facilitate this philosophical approach to learning.

Given that the whole underlying ethos of the iLabs tool is to enable communities to investigate and construct their own evolving learning representations, and that this was facilitated by various application tools that could be selected and tailored by members of particular communities, it made sense to investigate whether these same communities could be given tools to enable them to do the translation work themselves. Inherent in the iterative process of inquiry is a view of knowledge artifacts as part of the process of learning; tools to help the community learn and grow [10]. This contrasts the treatment of


http://ilabs.inquiry.uiuc.edu - iLabs - Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
iLabs
Home
Logout | Edit Profile
Create
Browse
Search
http://ilabs.inquiry.uiuc.edu/ilab/home/1206/setup_wizard.php
TextPad - [Document1 *]
Internet

CII Meetings

- Meetings Schedule
- CIC biweekly Friday meetings info

[Add New Item | Edit List]

Inquiry Page

The Inquiry Page is the precursor to iLabs, and is still a unique and much-used resource. Based on John Dewey's philosophy that inquiry begins with the curiosity of the learner, we use a spiral path of inquiry: ask, investigate, create, discuss, reflect.

Need Help?

- Help System iLab
- ILabs Quick Start Tutorial
- Email Support

iLabs Home :: Home

Welcome to Version 3 of Inquiry Labs (iLabs). iLabs are free, web-based tools to help you and your community communicate, collaborate, and develop a shared capacity for problem-solving. iLab software is developed continuously, through an open process of inquiry in which users participate.

People around the world have used iLabs to create interactive websites for school and university courses, research projects, neighborhood action, etc.

Version 2 is still available, along with the hundreds of iLabs created with it. iLab resources (publications, presentations, posters, etc.) are also accessible on the web.

Recently Updated iLabs

Inquiry-Based Learning, LIS 590 IBL
Mortenson Center test
CareNot
Social Informatics
Galvin Reference: Directories and Biographies
Community Informatics Corps
EOTU Student Conference Fall 2005
English 482 Chat
Cameron Test
504Galvin

Configure iLab

- Edit iLab Settings
- Add New Brick
- Arrange Bricks
- Remove a Brick
- Modify Permissions

Configure Bricks

- Welcome
- Main Content
- Need Help?
- Recently Updated iLabs
- About/Contact Us/Our Sponsors
- Engaged Minds
- EOTU
- Water CAMPWS
- Past Events
- Our Sponsors
- Inquiry Alumni
- iLabs Team
- Document Center
- Search and Browse
- iLab Work/Volunteer Opportunities
- Inquiry Page
- Need Help?

Figure 3. A screenshot of the iLabs homepage.

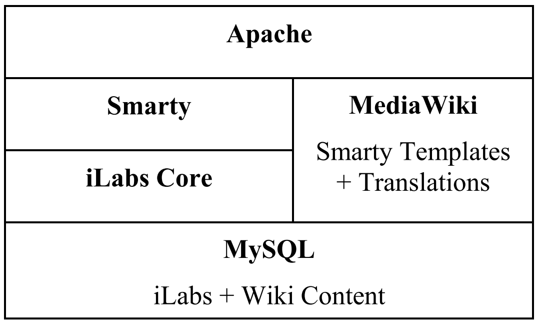


Figure 4. Architectural overview of the iLabs Translation system

translations as "products" by other approaches, and resonates with Fraser's argument that community translation can be understood from a process view [8].

In the following sections we introduce the iLabs translation system and its implementation using an open wiki integrated with the iLabs core. An example is given which illustrates how the translation mechanism works. Following this is a discussion of relevant issues and the limitations of this approach, followed by a sketch of future research to be conducted on the efficacy of our model and conclusions based on the work presented here.

THE DESIGN ENVIRONMENT

The findings presented here are derived from experiences with building the Community Inquiry Labs (iLabs) tools suite. The iLabs system in its entirety comprises three parts: the core iLabs code, a MediaWiki installation, and the Smarty template system (Figure 4). All three components are written in PHP. The wiki and iLabs core components are backed by a MySQL database. The wiki provides editing interfaces and storage for the interface translations and templates.

Requests are sent by the user's web browser to the Apache server which dispatches them to the iLabs system. iLabs then handles the request and gathers the necessary data from the MySQL database and the templates from the wiki. iLabs then bundles these data and passes them to Smarty [16] which renders the HTML output, pulling the translations of interface text dynamically from the wiki. The translated, data-populated HTML is then passed back to the browser by the Apache server.

iLabs

iLabs is a suite of free, open-source, web-based software that is developed, in an open and ongoing fashion, by people from many walks of life, from different countries and a wide range of ages. The iLabs suite has been used to create hundreds of interactive websites that support the communication and collaboration needed to pursue inquiry in classrooms, community centers, libraries, professional associations, research groups, and other settings [2]. iLabs includes software components for library catalogs, syllabi, document sharing, online inquiry units, discussion forums, blogs, calendars, and image galleries, all developed to explicitly support inquiry-based learning activities.

In some ways iLabs is like many other open-source content management systems and can be downloaded and installed on an individual server. However, iLabs has primarily been run on a shared web server where anyone can come and create an iLab for their personal, group, or community needs. This enables communities lacking the financial or technical resources to install and manage their own web server to have access to advanced tools for collaboration, communication and content management. This model of service is similar to Yahoo Groups [13] or Google Groups [12]; however, iLabs differs from these other systems in some fundamental ways.

In addition to the open-use model shared by Yahoo Groups and Google Groups, iLabs uses an open, participatory design model. Given the philosophical aim of supporting inquiry-based learning about any given topic, it is natural to also support inquiry based learning about the use and ongoing development of the iLabs tool itself. The participatory design model engages users in a community of inquiry around the design and development of the iLabs system [4]. This means that the understanding of what the iLabs system is and what it should be are constantly being questioned and evaluated. Unlike Yahoo Groups or Google Groups which are designed to support the developers' interpretation of what such systems should be, iLabs at any moment reflects the collective understanding by its users of the design and implementation of such a system

The tools in iLabs are packaged into units called bricks. Bricks are the components of the interactive website, encapsulating functionality like content management, document sharing, and bulletin boards. A community using the iLabs system can select which bricks they want to use in their community's iLab and can install several instances of each brick. Each brick can be customized in its naming, description, layout, and presentation. Users can also select a default language for the interface, offering a translation of interface elements into a localized lexicon. Furthermore, the content of the iLabs system can be entered in any Unicode encoding supported by the user's web browser. In these respects, the iLabs system is not unlike other content-management systems. The iLabs system has, however, expanded the notion of participation to allow users to take control over the interface elements directly.

MediaWiki System

Leuf and Cunningham define a wiki as "a freely expandable collection of interlinked Web 'pages', a hypertext system for storing and modifying information—a database, where each page is easily editable by any user with a forms-capable Web browser client" [19, page 14]. The first wiki system, the WikiWikiWeb, was developed by Cunningham as an add-on to the Portland Pattern Repository with the intention of allowing a community of programmers and designers to easily modify and expand the database of patterns.

Wikis are designed to support democratic, open participation where every user has the same capabilities to create and edit pages, engaging users in “an ongoing process of creation and collaboration that constantly changes the web site landscape” [19, page 16]. The wiki maintains a complete history of revisions for each article in the wiki database, allowing users to browse and undo changes. Editing wikis is easy, usually using plain text, HTML, or a simple wiki markup.

By some counts there are over 130 different wiki engines written in dozens of programming languages [11]. One very popular wiki system is the MediaWiki software used to host the Wikipedia (www.wikipedia.org). It is a robust, stable system which has a proven record of use. The MediaWiki system, having been built for the Wikipedia project, is designed for openness and ease of use; it is relatively unencumbered by complex permissions and user management overhead and it uses a simple markup language which is easy to learn.

Wikipedia itself has a strong tradition of supporting different languages. Building on the open source model of community participation, but drawing on a much wider constituency by having very low technical barriers to entry, it has grown very rapidly. Not only are there currently (April, 2006) over one million articles in English, but almost three million additional articles in 228 other languages [18].

Participation in any language version of wikipedia is easy. Volunteers may choose to translate an article from an existing language, or create a new article from scratch. Various mechanisms are evolving to address consistency issues between the same entries in different languages.

Creating a new language version of Wikipedia is more complex and requires a much larger commitment of time, effort and some technical skill. To create a translation of Wikipedia, it requires setting up a separate MediaWiki installation. Translations of the content are not accommodated within a single wiki, but are spread across language-specific wikis. Consider also that although there are 200 different language installations of Wikipedia, the MediaWiki software itself only supports 90 different language translations of the interface. For some reason, what has been considered a significant enough difference to warrant a separate language content installation fails to translate into additional language translations of the interface messages. Perhaps the reason has to do with the difference in barriers to using Wikipedia versus creating translations. While Wikipedia has very low barriers to participation, creating new translations of the MediaWiki interface is difficult, requiring users to write a PHP script – something which is easy to skilled programmers, even those unfamiliar with PHP, but is a significant barrier to participation for non-programmers or non-technical users. By contrast, we wanted to achieve the speed, flexibility, granularity and incrementality of Wikipedia’s approach to *content* translation, but applicable to *interface* translation.

MASHING UP ILABS WITH MEDIAWIKI

The reasons why we chose to use MediaWiki in the iLabs system can be understood by comparing both systems in terms of their philosophical similarities as well as their technological compatibility. In the context of the iLabs project it was very important that the system chosen for managing the translations and templates allow for easy, open participation in creation and editing of content.

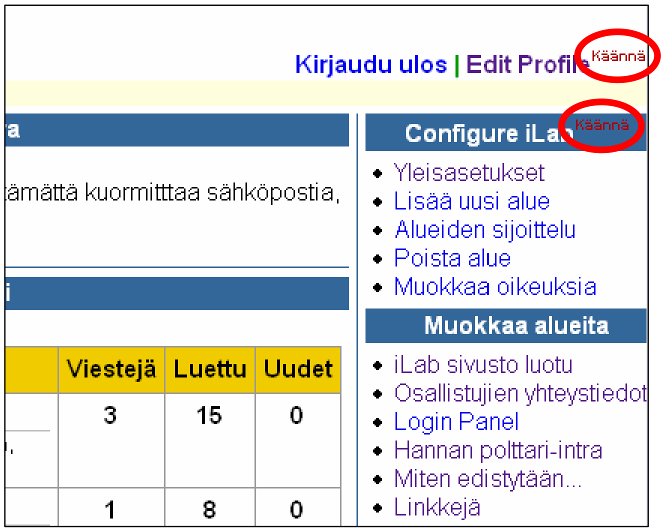
The design and development philosophy of iLabs has adhered to the core values of John Dewey’s theory of Inquiry as described in [4]. There are many similarities between these principles and those of wiki design. Principally wikis and iLabs both employ open participation models. Wikis allow anyone to participate and contribute to the creation and organization of content and in the case of MediaWiki, which is open-source, to the design and development of the system as well. Likewise, the iLabs system has used open, participatory design and development practices which have been guided by the Inquiry theory principle of empowering users to participate actively in the design, development, and use of resources. The core values of Inquiry also stress the community orientation in the process of knowledge construction whereby members share and build upon each other’s work. This is also one of the defining principles of wikis, where individuals are invited to edit each other’s postings and build on the work of others. Diversity of opinion is something which has been respected as a valuable resource in the iLabs design and development. Wikis, by nature of their open access and easy use, encourage users from all backgrounds and technological abilities to participate.

Integrating Technology

Backing the iLabs system is a MediaWiki installation. The wiki provides an easy to use and manage system for organizing and editing the interface templates and translations. iLabs interfaces are defined in an html-like language called Smarty – a PHP-based templating language. Each webpage a user sees in the iLabs system is dynamically compiled from a set of Smarty templates. Each template is stored as a separate page in the wiki. Likewise the translations for the interface are stored as wiki pages.

The Smarty template language was chosen because it uses an intermediate HTML language for the source templates. These source templates are then compiled into PHP scripts which render the proper HTML document.

The MediaWiki software allows developers to define customized extensions. To create a custom extension, first a developer must define a tag which MediaWiki will use to call the extension. Everything within the tag is passed to the extension handler, bypassing the regular MediaWiki engine. We use the tag `<smarty>` to denote that everything


Kirjaudu ulos | Edit Profile Käännä

a

ämättä kuormitttaa sähköpostia,

i

	Viestejä	Luettu	Uudet
	3	15	0
	1	8	0

Configure iLab Käännä

- Yleisasetukset
- Lisää uusi alue
- Alueiden sijoittelu
- Poista alue
- Muokkaa oikeuksia

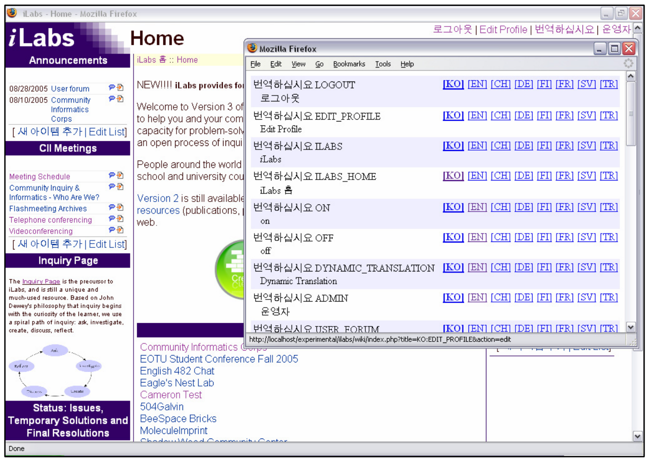
Muokkaa alueita

- iLab sivusto luotu
- Osallistujien yhteystiedot
- Login Panel
- Hannan polttari-intra
- Miten edistytään...
- Linkkejä

Figure 5. A Finnish iLab with Finnish interface translation. The untranslated text is flagged with the red link: Käännä (Finnish for "translate").

between the open and closing tags should be passed to our Smarty extension. The Smarty extension takes the content passed it by the MediaWiki engine and uses the Smarty library to compile it into a PHP script which is then saved in a location where iLabs can access it. Meanwhile, the Smarty source template is saved into the wiki database. When a user views the template page in the wiki, they are given a view of the template's source code. The wiki system also provides an archived history of revisions to each template that users can browse through and roll-back to at any time if the template is ever broken.

In addition to the templates, the wiki is also used to manage the translations of the interface. The list of languages supported by the iLabs system is stored on a wiki page. Adding support for translation into a new language or lexicon is as easy as adding an appropriate language abbreviation to this wiki page. Translation of the interface



iLabs - Home - Mozilla Firefox
iLabs
Home
Announcements
CII Meetings
Inquiry Page
Status: Issues, Temporary Solutions and Final Resolutions
Done
File Edit View Go Bookmarks Tools Help

로그아웃 | Edit Profile | 번역하십시오 | 운영자

iLabs 홈 :: Home

08/28/2005 User forum
08/10/2005 Community Informatics Corps
[새 아이템 추가 | Edit List]

Meeting Schedule
Community Inquiry & Informatics - Who Are We?
Flashmeeting Archives
Telephone conferencing
Videoconferencing
[새 아이템 추가 | Edit List]

The Inquiry Page is the precursor to iLabs, and is still a unique and much-used resource. Based on John Dewey's philosophy that inquiry begins with the curiosity of the learner, we use a spiral path of inquiry: ask, investigate, create, discuss, reflect.

NEW!!!! iLabs provides fo

Welcome to Version 3 of
to help you and your com
capacity for problem-solv
an open process of inqui

People around the world
school and university cou

Version 2 is still available
resources (publications,
web.

Community Informatics Corps
EOTU Student Conference Fall 2005
English 482 Chat
Eagle's Nest Lab
Cameron Test
504Galvin
BeeSpace Bricks
MoleculeImprint

Mozilla Firefox

번역하십시오 LOGOUT [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
로그아웃
번역하십시오 EDIT_PROFILE [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
Edit Profile
번역하십시오 ILABS [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
iLabs
번역하십시오 ILABS_HOME [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
iLabs 홈
번역하십시오 ON [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
on
번역하십시오 OFF [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
off
번역하십시오 DYNAMIC_TRANSLATION [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
Dynamic Translation
번역하십시오 ADMIN [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]
운영자
번역하십시오 USER_FORUM [KO] [EN] [CH] [DE] [FI] [FR] [SV] [TR]

http://localhost/experimental/ilabs/wiki/index.php?title=KO:EDIT_PROFILE&action=edit

Figure 6. Popup listing of translatable text on the iLabs interface with links to the translation wiki.

is done using a simple key-value dictionary. Chunks of text on the interface are identified by a key which is combined with the abbreviated name of the language to create a wiki page title. The content of the page is the value for that particular key translated into the specified language. For example, the key "WELCOME_END" (Figure 8) is translated into English (EN) and Korean (KO) with page titles EN:WELCOME_END and KO:WELCOME_END respectively.

When elements of the interface have not been translated into the user's language, they appear with a red translation hyperlink to the wiki (Figure 5). In English, these hyperlinks read "Translate", Figure 5 shows the hyperlinks translated into the Finnish word "Käännä", meaning "translate". By clicking on these flags, users can input translations as they encounter aspects of the interface. Alternatively, the user can click on a translation link at the top of every page which will pop up a window containing all of the interface text that can be translated, the keys for those elements, their current translations in the user's language, and links to the wiki for translating the text into the available languages with the user's language as the first option (Figure 6). The wiki-based system allows a community to collaboratively define and refine the terminology used to name and describe the system, allowing everyone to participate in the process. The process is lightweight and incremental. Participation can be as little as adding or revising a single command. Of course this does lead to users interacting with partially translated interfaces, but we believe that this is a reasonable design trade-off.

Community-based Translation in iLabs

The community-based translation feature was developed in response to interest expressed in using iLabs by colleagues from Finland and Chinese users at our University, beginning in September, 2004. The mechanism for creating and editing translations has gone through several iterations in the past 14 months as we have gained deeper understanding of the translation process. The original design of the translation system was much like the traditional, product-based approaches. We generated complete lexicons of all the iLabs interface messages which were then distributed to the Finnish and Chinese volunteers. These volunteers then sent back a translated lexicon which we installed in the iLabs system. This process, however, was cumbersome and involved a lot of human intervention as translators tended to work in fits and spurts, incrementally building the complete lexicon over time rather than in a single shot. Also, the translators found it challenging to immediately translate all aspects of the interface from the lexicon file as it did not give the context in which the word or phrase is used. As a result, translations were often changing as translators saw where and how the translations were applied. Furthermore, managing the translations through text-files was challenging as developers here in the US lacked adequate font-support and

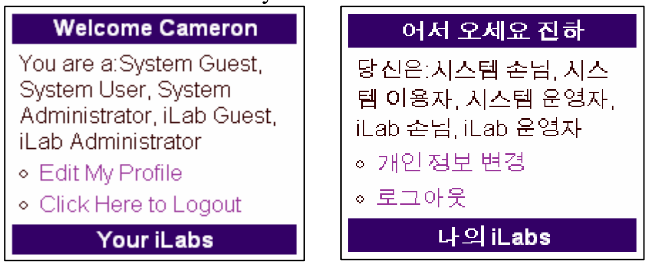
Language	# Messages Translated	% Translated
English	496	100.0%
Chinese	444	89.52%
Finnish	293	59.07%
Turkish	78	15.73%
French	49	9.88%
Korean	10	2.02%
Swedish	8	1.61%
Italian	3	0.60%

Table 1. Breakdown of the 8 different translations of the iLabs interface, showing the percentage of the interface messages translated in each.

Unicode-compatible tools for merging the Finnish and Chinese revisions.

Over the course of several intermediate implementations we moved the translation feature from a back-end function to a front-end system. Eventually, we settled on the wiki-based system for managing the translations as the wiki offered a robust interface for editing the translations, a history of revisions to explicitly support iterative change, and a discussion space useful for translators to interact.

In all, since the first system, we have seen interest in having translations in 7 additional languages other than English (a complete list can be seen in Table 1. Beyond English, Chinese is the language with the most complete translation, 89% of the interface has been translated. One reason why Chinese has such a complete translation is that the University hosted a visiting Chinese scholar program over the summer of 2005. Prior to the arrival of the foreign visitors Chinese scholars on campus set about translating the interface for the workshop. More typically, translations remain a work in progress with 0% to 60% of the interface translated. It seems that a partially translated interface is not too bewildering for potential users as much of the content of the iLabs system is input by the users directly in their own language resulting in pages which can contain extensive amounts of accessible content information. Also, it is the more commonly used interface elements that tend



Welcome Cameron

You are a:System Guest, System User, System Administrator, iLab Guest, iLab Administrator

- Edit My Profile
- Click Here to Logout

Your iLabs

어서 오세요 진하

당신은:시스템 손님, 시스템 이용자, 시스템 운영자, iLab 손님, iLab 운영자

- 개인 정보 변경
- 로그아웃

나의 iLabs

Figure 7. A portion of the iLabs interface in the default English and naively translated into Korean.

to get translated first, such as those on an iLabs home page, and the more obscure power-features that tend to languish in translation limbo. The common features are also those most used by typical (rather than power) users. Consequently, even when only a relatively low percentage of all the language in the interface is translated, this can still meant that a large proportion of all uses are now performed in the native language.

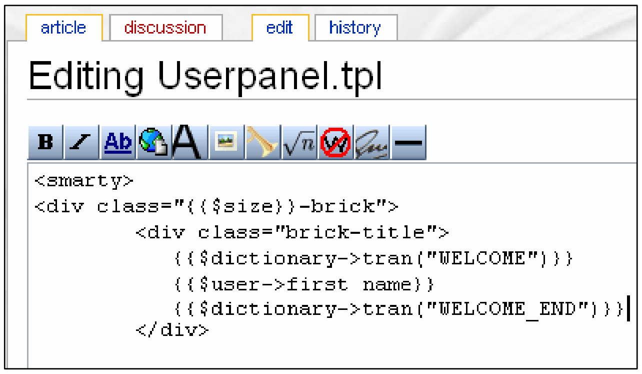
Example: Modifying and translating the interface

This sequence of screenshots shows users being able to change the interface to better accommodate translation into the Korean language. Figure 7 shows a section of the interface as designed for an English speaking audience with the same region as seen by a Korean user (or someone who has selected Korean as their preferred language).

In some cases a direct word-for-word translation will suffice. The translator just has to enter the equivalent word in the other language. Even in this trivial case, it is highly unlikely that the translation will be exactly the same length as the source text. Fortunately, iLabs is implemented using Cascading Style Sheets as a way to enable greater customization of both brick selection and overall visual appearance of a given iLab for a particular user community. As a consequence of this prior design decision to support a different kind of tailoring, translations are able to exploit the relative interface positioning of HTML and CSS and so in most cases are able to cope with variants of length. A more complex issue is that many translations can not be done directly but require more nuanced additional steps, and often additional or replacement text. For example, the Korean translation contains both translations of language, but also culture. The 'Your iLabs' text has been translated into the Korean equivalent of 'My iLabs' – reflecting a cultural difference in how this kind of information is typically presented on Korean websites.

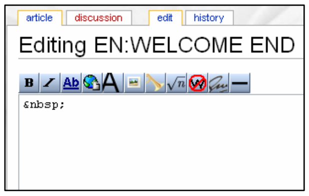
The welcome message has also been translated (in this case, for a different user); however, the translation is not entirely correct. It is not grammatically incorrect, but to a native Korean speaker the translation is awkward; containing a mixture of formal and informal language. To correct for this, the suffix 님 needs to be added to the end of the user's name (진하) in order to make it agree with the formality of the greeting (어서 오세요). The suffix cannot be accommodated in the current translation scheme which was based on the English grammar depicted in Figure 7 (where the name lacks a suffix).

Given the mutability of the interface, however, the user is able to modify the template for the page and define a suffix field (Figure 8: 'WELCOME_END'). This translatable suffix is then translated in Figure 8 into the Korean 님, and to a HTML non-breaking space entity in English (Figure 8). The final result can be seen in Figure 9.



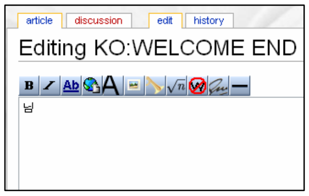
Editing Userpanel.tpl

```
<smarty>
<div class="{($size})-brick">
        <div class="brick-title">
           {{$dictionary->tran("WELCOME")}}
           {{$user->first name}}
           {{$dictionary->tran("WELCOME_END")}}
        </div>
```



Editing EN:WELCOME END

```
&nbsp;
```



Editing KO:WELCOME END

```
님
```

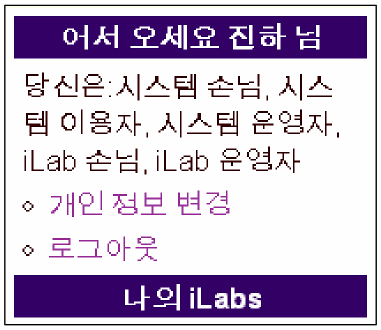
Figure 8. Screenshots of the translation wiki showing the user modifying the interface template and inputting translations in English and Korean.

The community-based translation and mutable interface make a powerful combination. The change made to accommodate a suffix in Korean can benefit other users of other languages which use suffixes (e.g. in Japanese websites often greet visitors and address them with a –san or –sama suffix). Here we can see the way in which the expertise of a member of one community can benefit many users in many different communities through a process of community development and sharing. The easier, more basic translations can still usefully be provided by a larger group of volunteers with less technical expertise and effort.

DISCUSSION

Many interfaces allow some level of customization. For example Google "personalized home" or Yahoo's portal features in "my Yahoo" allow users to select from many options and have some (but not complete) control over the arrangement of those options on the page. However we claim that our system is beyond customization. There is no fixed or predefine language for users to choose -- rather they can make change they want and so evolve the interface.

By allowing for the open creation of new translation lexicons, the iLabs system is able to support increasing numbers of users. Instead of having to seek out translators for new languages or rely on users to demand new language support, the collection of supported languages expands to fit the needs of those who are currently using the iLabs system. Furthermore, our approach to community-based translation allows *any* individual within the community to participate in the translation instead of the volunteer-based system described by McDevitt et al. and Perugini et al. [21, 27]. Community-based translation has the added benefit of



어서 오세요 진하 님

당신은:시스템 손님, 시스템 이용자, 시스템 운영자, iLab 손님, iLab 운영자

- 개인 정보 변경
- 로그아웃

나의 iLabs

Figure 9. The completed Korean translation of the interface.

creating a sense of membership and ownership towards the interface among community members [27].

The quality of the translation created by the community reflects the community's understanding of the system. This might mean that there is no "wrong translation" [21] in the sense that the lexicon has not be imposed upon the users by an external designer—the quality of the translation no longer depends on the designer's skill. This issue is not limited to cross-language translation, but also resolves problems where the designer uses technical language or jargon which is foreign to the user. However, just as with Wikipedia, collaborative translation is vulnerable to error, neglect, vandalism and wild variations in the quality of its components [31]. Worse still, as with OSS usability, collaborative translation efforts are unlikely to maintain internal and external consistency, unless explicit efforts are taken to address this and other issues of quality and adherence to usability standards [24].

Fortunately, the wiki's vulnerabilities are also its strengths. With many people using it, as errors are spotted they are easy to fix, as well as to introduce. This enables user to effectively maintain their lexicon. Other approaches have battled with control issues, trying to maintain the integrity of the translation while accommodating change. The wiki provides an archived history of the various iterations of the translation, allowing users to undo changes as well as a discussion space for users to resolve disputes over translations. Here the community-based approach can be fully exploited similarly to how Wikipedia relies on the community of contributors to correct errors in articles and resolve differences over content.

When combined with the wiki-editable templates, every aspect of the interface can be "translated" into a culturally-specific version. Russo and Boor [28] listed seven aspects of the interface that must be considered in translation; of which text is just one. The other elements of images, color, flow, symbols, and date format can all be translated; the only aspect not translated is the functionality.

This does not just apply to translating to native languages. An English language application may still be confusing or inappropriate in its language use to native English speakers. For example, iLabs was developed to support a particular activity (inquiry based learning) that can be applied in a large variety of contexts. These include formal learning at all levels through high school, undergraduate, and graduate education, as well as research projects. Terminology appropriate for scholarly communities may not be appropriate for elementary school students (e.g. the concept of "bricks" is one used by researchers, but "tools" might be more suitable for other audiences). Additionally, some contexts of learning and inquiry supported by iLabs may be explicitly outside systems of formal education. For example, a group of black women used the tool to explore issues of personal physical and spiritual health. Terms in the interface such as 'student' and 'lesson' are not only inappropriate in that context but may serve to deter actual use.

FUTURE WORK

The system as presented here has been in use for several months. The maturity of the system in terms of stability, integration and community contributed languages is reaching the point where we can begin thinking about conducting evaluative studies. The evidence from actual use through the various iterations to date, gives us confidence that despite various teething problems and the peculiarity of the concept of 'translate as you go', at least some people are willing and able to participate. The first study we plan to execute addresses the overall usability of these translation features. How easy is it for communities with little technical skill to create and manage translations of the interface? Can users recognize and understand the mechanisms we have devised for selecting a translation to use and managing language inheritance rules? How difficult is it for users to translate untranslated interface messages and to edit existing translations? Within this study we also seek an understanding of the dynamic community processes involved in translation task division and conflict management/resolution. What additional tools, if any, are needed for communities to manage the articulation work involved in carrying out the translation? How static are the translations, do they change over time? Does the language of the interface reflect an increasing understanding of and technical sophistication with the system?

Another study is geared more at understanding the effects of community languages on usability. We envisage a study where we can take a translation of the interface done by one community and evaluate its usability using members of another community which speaks the same national language. For example, we have translations of the interface done in Chinese done by a community of Chinese graduate students in the University. How usable is this interface by a first-generation Chinese-American or a mainland Chinese? What are the significant variations or hurdles? Which, if any, concepts or interface elements are not translatable?

We also see room for greater improvements in integrating the two systems. The interlinking described in this paper is mostly one way: users can easily navigate from the system interface to the wiki to input translation information. What about the reverse? How can the wiki be more meaningfully integrated with the iLabs core? Currently, the wiki user account data is controlled by the wiki and user customizations for features like changing the interface of the wiki are unfortunately limited to the models we have sought to undo. Can we create a system where the wiki interface can itself, be translated by the wiki? Or, can the interface for the wiki be removed entirely? We believe that exploring this latter idea might prove fruitful utilizing the concept of direct manipulation.

Finally, we need to explore whether in larger scale use the community-based control mechanisms provided by the wiki are enough to counteract spam and vandalism.

CONCLUSION

Widening access to computer technology is a core feature of why people have chosen to get involved in research areas such as universal usability and participatory design. In addition to the social desirability of wider access, there are often sound commercial incentives as well. Enabling people to overcome individual and collective barriers to access is often time-consuming and expensive due to the many forms that these barriers can take. Fortunately, a collaborative approach can be helpful. This is inspired by the successes of open source collaborative software development by technically skilled programmers and wiki-based collaborative content creation by less technically skilled end-users. With care, putting the two approaches together makes it possible to develop interfaces that are individually and collectively tailorable. This care is necessary since it inevitably involves the development and use of a meta-interface to manipulate the interface to be improved.

ACKNOWLEDGEMENTS

The authors wish the express thanks and gratitude for the efforts of all the volunteers who have participated in iLabs development and translation efforts. We are particularly indebted to the efforts of Sami Serola and Xiuling Nie for motivating and organizing translation efforts in Finnish and Chinese, respectively.

This work was partially supported by The WaterCAMPWS, a Science and Technology Center of Advanced Materials for the Purification of Water with Systems under the National Science Foundation agreement number CTS-0120978.

REFERENCES

1. Bishop, A.P., Bruce, B.C., and Jones, M.C. (2006). Community Inquiry and Informatics: Collaborative Learning and Action through ICT. *International Journal of Computer-Supported Collaborative Learning (In Press)*.
2. Bishop, A.P., Bruce, B.C., Lunsford, K.J., Jones, M.C., Nazarova, M., Linderman, D., Won, M., Heidorn, P. B., Ramprakash, R., and Brock, A. (2004). Supporting Community Inquiry with Digital Resources. *Journal of Digital Information, 5(3)*, Article #308, http://jodi.ecs.soton.ac.uk/Articles/v05/i03/Bishop/.

3. Blom, J. (2000). Personalization – A Taxonomy. *Conference on Human Factors in Computing Systems, Extended abstracts on Human factors in computing systems CHI 2000*, ACM Press, 313-314.
4. Bruce, B.C. and Jones, M.C. (2005). Evolution of the Inquiry Page: How Participation in Inquiry Leads Technology. http://people.lis.uiuc.edu/~haythorn/ElearningWorkshop.htm
5. Cooper, A. (1999). *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity.* Macmillan Publishing Co., Inc. Indianapolis, IN, USA.
6. Cristiano, L.K. (1989). Methodology for Comparative Selection of Interactive Database Interface Types. *ACM SIGHI Bulletin 21(1)*, ACM Press, 29-36.
7. Di Biase, B. (1987). Translating for the community. Australian Review of Applied Linguistics supplement 4: 52-65.
8. Fraser, J. (1993). Public Accounts: Using Verbal Protocols to Investigate Community Translation. *Applied Linguistics 10(2)*, Oxford University Press, 325-343.
9. Gutwin, C. and Greenberg, S. (1998). Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness. *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, ACM Press, 207-216.
10. Hickman, L.A. (1990). *John Dewey's Pragmatic Technology*. Indiana University Press, Bloomington, IN, USA.
11. http://c2.com/cgi/wiki?WikiEngines
12. http://groups.google.com/
13. http://groups.yahoo.com/
14. http://ilabs.inquiry.uiuc.edu/
15. http://services.google.com/tcbin/tc.py
16. http://smarty.php.net/
17. http://www.google.com
18. http://www.wikipedia.org
19. Leuf, B. and Cunningham, W. (2001). *The Wiki way: Quick collaboration on the Web*. Upper Saddle River, NJ, USA: Addison Wesley.
20. Leventhnal, L., Teasley, B. and Stone, D. (1994). Designing for Diverse Users: Will Just a Better Interface Do? In *Proceedings of CH1 1994: Human Factors in Computing Systems,* ACM Press/Addison-Wesley, 191-192.
21. McDevitt, K., Pérez-Quiñones, M.A., and Padilla-Falto, O.I. (2004). Design of a Community-based Translation Center. *Technical Report cs.HC/0401007, Computing Research Repository (CoRR)*, http://arxiv.org/abs/cs/0401007
22. Mørch, A. (1997). Three Levels of End-User Tailoring: Customization, Integration and Extension. *In Kyng, M. and Mathiassen, L. Computer and Design in Context*. MIT Press, Cambridge, MA, USA, 61-76.
23. Murata, T., Kitamura, M., Fukui, T., and Sukehiro, T. (2003). Implementation of collaborative translation environment 'Yakushite Net'. *MT Summit IX*, 479-482.
24. Nichols, D.M. & Twidale, M.B. (2003). The Usability of Open Source Software. First Monday 8(1) - January 6th 2003.
25. Nichols, D.M., Witten, I.H., Keegan, T.T., Bainbridge, D., & Dewsnip, M. (2005) Digital libraries and minority languages. *New Review of Hypermedia and Multimedia* 11(2), 139-155.
26. Nielsen, J. (1990). Designing for International Use (panel*). In Proc. of the SIGCHI conference on Human factors in computing systems: Empowering people 1990*, ACM Press, 291-294.
27. Perugini, S., McDevitt, K., Richardson, R., Perez-Quiñones, M., Shen, R., Ramakrishnan, N., Williams, C. and Fox, E.A. (2004). Enhancing Usability in CITIDEL: Multimodal, Multilingual, and Interactive Visualization Interfaces. *In Proc. of the 4th ACM/IEEE-CS joint conference on Digital libraries* 2004, ACM Press, 315-324.
28. Russo, P. and Boor, S. (1993). How Fluent is Your Interface? Designing for International Users. *In Proc. of the SIGCHI conference on Human factors in computing systems 1993*, ACM Press, 342-347.
29. Shimohata, S., Kitamura, M., Sukehiro, T. and Murata, T. (2001). Collaborative Translation Environment on the Web. P*roceedings of the MT Summit VIII*, , 331-334.
30. Shneiderman, B. (2000). "Universal Usability," Communications of the ACM 43, 5, 85-91.
31. Stvilia, B., Twidale, M.B. Gasser, L. & Smith, L.C. (2005). Information quality in a community-based encyclopedia. In: S. Hawamdeh (Ed.), Knowledge Management: Nurturing Culture, Innovation, and Technology - Proceedings of the 2005 International Conference on Knowledge Management. Charlotte, NC: World Scientific Publishing Company 101-113.
32. Weld, D.S., Anderson, C., Domingos, P., Etzioni, O., Gajos, K., Lau, T., and Wolfman, S. (2003). Automatically Personalizing User Interfaces. In Proc of the International Joint Conference on Artificial Intelligence (IJCAI03), Acapulco, Mexico.