A Community Cloud: Archive Retrieval in Multiple Language Services

Wei-hua Zhao[1] Zhan-wei Liu[2] Zheng-xu Zhao[3]
Shijiazhuang Tiedao University
Hebei, Shijiazhuang, P R China
1. e-mail: zhaoweihua9@hotmail.com
2. e-mail: hoson@126.com
3. e-mail: zhaozhengxu@staff.stdu.edu.cn

Abstract—**This paper presents a common method and practice in implementing a community cloud, where archives have to support the services in multiple languages and cultures. It demonstrates an internationalization process, the development of software, the migration of archive data and the integration of archive retrieval system. The paper shows how such archive retrieval is carried out and how the system is implemented. Although it does not claim to be the best or the only possible solution, the method provides practical and useful tools for establishing and managing digital archives that serve as a community cloud where it essentially requires internationalization or globalization or localization. The main achievement of the paper resides in the presentation of a complete and useful method for archive retrieval in multiple languages.**

Keywords-community cloud; digital archive; globalizaton; internationalization; information organization

I. INTRODUCTION

Cloud computing is considered as web-based processing, whereby shared resources, software, and information are provided to computers and other devices such as smart phones on demand over the Internet [1]. A community cloud may be established where several organizations have similar requirements and seek to share infrastructure so as to realize some of the benefits of cloud computing. Examples of community cloud include Googles "Gov Cloud" [2]. It could be interesting to consider that archive services are evolving from their traditional form toward an emerging digital domain as a community cloud. Literatures [3,4] reveal that a phenomenal proliferation of digital data clearly underscores the ease with which it is being produced and the capacity in which it is to be accessed. While effective archiving and retrieval the information still remain a work in progress, the importance of and the challenge for accessing it in multiple languages with multiple cultures are being recognized and thus there have been a various efforts made to address the needs in specific areas like education services, government administration and enterprise management.

Archive services across difference communities and cultures have evolved into a massive digital media and files in various forms and different languages, yet the current data retrieval practices, due to the limitation of software tools and archive systems, still confine their purposes within specific languages and culture orientations. Those practices in nature are normally carried out in the native or original forms of the archived documents. For example, the archive system deployed in the Shijiazhuang Tiedao University holds the live data records, predominately in Chinese, of staff and students for over 60 years span in hundreds and thousands of individuals. However each year there have been an increasing amount of service demands for retrieval, translation and interpretation of those native documents into various languages to suit for offshore business, overseas collaborations and of course for those whose are to work and study abroad. The workload in transforming those documents into its specific needs and purposes poses a phenomenal challenge both in time and accuracy [5]. This paper reports an investigation into the method that retrieves, translates and interprets the native documentation into multiple languages and cultures to cater for different service needs. The objective is to establish a useful practice that can utilize various tools currently available to transform the archives from their native form into different languages and culture needs. It is expected that this practice will be an essence for effective data translation, index and migration, thereby providing technical solutions to the aforementioned problems.

The text to follow gives a brief description of internationalization in Section II. Section III presents the use of methodology management in the system design and Section IV details the functions of the system. Section V provides the implementation of the system which is followed by conclusions and future works included in Section VI.

II. INTERNATIONALIZATION

Globalization, internationalization and localization are terms referring to the process of developing software systems, media products, documentation and other collateral material for people who speak foreign languages or constitute a specific cultural group with a large language community. They are long words and people took the habit of writing abbreviations instead. As a result, the three terms are abbreviated as g11n, i18n, l10n, with the intervening number referring to the number of letters between the first and last of each word [6].

A. Terminology

By i18n, it refers to the process by which a program or a set of programs turned into a package is made aware of and

 ISBN: 978-1-61208-124-3

able to support multiple languages. This is a generalization process, by which the programs are untied from calling only English strings or other English specific habits, and connected to generic ways of doing the same, instead. Program developers may use various techniques to internationalize their programs. Some of these have been standardized. GNU Gettext offers one of these standards [7] and this paper will be based on GNU Gettext.

By l10n, it means the operation by which, in a set of programs already internationalized, a program is given all needed information so that it can adapt itself to handle its input and output in a fashion which is correct for some native language and cultural habits. The formal description of specific set of cultural habits for some country, together with all associated translations targeted to the same native language, is called the *locale* for this language or country. Users achieve localization of programs by setting proper values to special environment variables, prior to executing those programs, identifying which locale should be used [7].

For globalization, or g11n, one could consider that it stands for a much broader activities than for some specific technology areas such as software design and system development.

B. Development of i18n

Organizational bodies and commercial companies are always looking for competitive strengths and improved business practices. As Perrow [8] stated, in today's economy, this translates to a demand for better software. The increased demand has to do with a growing recognition that, throughout the new economy, software is the means for conducting business, tapping new markets globally, and connecting suppliers, manufacturers and end users in a worldwide domain. While the Internet serves as a vital tool for transaction and communication, the software systems themselves - often on either end of an Internet connection – do the heavy lifting and represent the greatest challenges and opportunities for business growth.

An i18n process represents the way in developing such modern software systems that enable organizations to exploit strategically localized advantages and economies of scale to leverage a competitive edge world-wide. The concept of i18n describes the establishment of a network of cross-border activities between and within companies in which any or all of the organizations' departments may be involved. The main standard motives for i18n are seeking market, resource, efficiency and strategic asset [9,10,11]. By any measure, i18n is not a solved problem. Although i18n has become a mainstream software development process, achieving it in streamlined, flexible and standardized manner remains a grand challenge [12]. This may be partially due to the misinterpretation of certain problem areas of i18n [13]. Table I lists the key problem areas and the related questions that are extracted from the discussion, with each area being represented by a question to highlight the fundamental problems for the i18n practice.

TABLE I. PROBLEM AREAS PERTAINING TO THE SOFTWARE DEVELOPMENT PROCESS OF I18N

Problem Area	Description
User interface	Does i18n mean that software can be simply translated via externalising its user interface?
Software translation	Can software translation in i18n process always use the best phrase in the target language?
Programming tool	Can programming tools that themselves have i18n always produce i18n code?
Unicode support	Is software that supports Unicode an i18n system or product by itself?
Open source	Does the use of open source in software product mean that i18n requirements are not applicable for the product?
Standard encoding	Is a standard encoding is automatically making a product having i18n?
Internal tool	Is it true that all company employees speak English so only English needs to be supported by internal tools?
Administration interface	Do administration interfaces need i18n?
Product module or component	Does every product, module or component need i18n?
Product version	Does it mean by adding i18n in the last release a job well done?
Customer feedback	If something is wrong, will the customers' feedback to the developers?
Multilingual capability	Can a software product that works in a foreign language be considered to have i18n?
Base code	Is i18n implemented after the base product is written by a separate group of engineers?
Software engineering	Is i18n only needed in the software development department?

III. METHODOLOGY MANAGEMENT

Methodology management, normally called design methodology management (DMM), originated from computer aided design frameworks that are software environment that integrate design tools and programs required for prototyping computer operating systems and managing the data being generated [14]. To attain rapid design processes of high level automation, frameworks have to select and execute tools automatically for lower-level tasks to enable designers to concentrate on higher-level decisions. The sequence of design tools is called methodology and the functionality of selecting and executing the tools is called methodology management. A software environment, often part of a framework for selecting and executing design methodology, is called methodology management system [15].

As the design automation community begins to understand the benefits of the technology, its expectations grow: less error-prone design, rapid prototyping systems,

new and customer-tailored product development, high product quality and improved productivity. A good DMM system can bring these benefits to different categories of users such as product designers, tool developers, system developers, chief designers and company managers [14]. DMM is relevant not only to electrical design, but also to software design and other fields like mechanical design. Nevertheless it has not yet become a subject of much discussion in either research community [16] or i18n process. This article introduces DMM into i18n practice to develop an i18n framework and it does so for three purposes. The first one is to demonstrate how an integrated and streamlined i18n process can be carried out and thus to give an insight into the i18n process from an implementation perspective and with sufficient technical details. The second is to provide a practical approach with useful techniques and tools for packaging of software with complete i18n support. The last one is to help in understanding the i18n process in relation to the questions in Table I.

IV. INTERNATIONALIZATION OF ARCHIVE RETRIEVEAL

Operations for i18n are often accomplished using software tools, both interactive and automatic, and DMM addresses the need to manage the manner in which these tools are executed to achieve a desired function. According to Fiduk et al [17], this paper adapts the definition of following terms.

1) Execution environment: it is a computing environment to manage the tools, tasks, flows, data and information movement that are essential for an i18n process to be accomplished.

2) Tool: this is a single executable program capable of performing a specific function toward i18n.

3) Task: it is an abstraction of a function toward i18n, for example, translation of an error message or version check of a source code file.

4) Flow: it is the order in which tasks are executed. The definition and manipulation of flows provide a mechanism to describe a sequence of tools that make up a process or task and tasks make up a methodology.

5) Process: this is a specific combination of tools and/or other processes that performs a function toward i18n.

6) Tool invocation: This is the selection of a tool and the use of it to perform what is needed to be done.

7) Operation: it is an atomic action within a process.

8) Methodology: this is a specified sequence of tasks.

A. The Framework

In general, DMM should have an execution environment that is responsible for user interaction, launching tasks, monitoring processes, automatically executing flows, and so on. For this paper, the execution environment has following specifications:

- It is operated under the common MS Windows XP operating system.
- All tools and software are run as Win32 programs (similar tools and software with different compilations should be used for other operating systems).
- The i18n development language is C/C++ under Microsoft Visual Studio .NET 2003.
- The i18n software to be developed is open source, so are all internal tools for the i18n process.
- The execution environment is compliance to open source community's Native Language Support Library and Tools, GNU Gettext Tools, Version 0.14.4 [7].

Based on the above specification, the execution environment is set up and its flow graph is shown in Fig.1. The graph formalism represents individual methodologies. It is actually a bipartite cyclic directed graph that has two types of nodes wherein all edges connect one type to the other and there are no paths from a node back to itself. The two types of nodes are task nodes and specification nodes. Each task node is labelled with a task description. There are two types of task nodes: terminal and non-terminal. A terminal task node represents a run of an i18n tool or programme and is called a tool invocation; it is drawn with single circle.

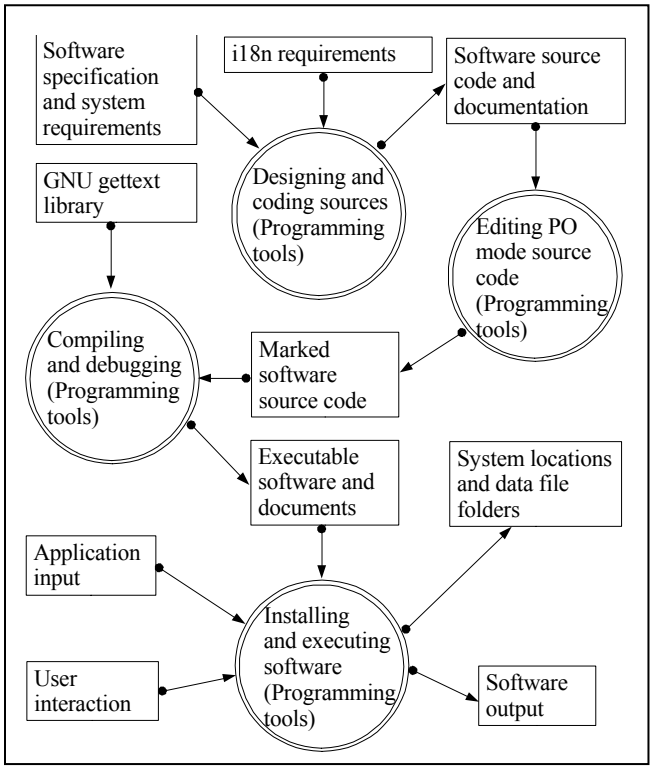

Figure 1. DMM information flow graph of the execution environment for i18n of softwar.

The graph representation of the execution environment is arranged in a top-down manner by applying transformations to nodes that represent non-terminal tasks. Fig. 2 shows the next level graph (or sub-graph) that describes a process where the *Marked software source code* is inputted to the *Extracting translatable code* task by which a portable object

template (POT) file named as *PACKAGE.pot* is generated through the *xgettext* tool. The *Comparing and refreshing* task is to obtain updated *Target language PO files*. Note that a target language PO file is a PO file that has the translation of a target language for all original program strings (in file *PACKAGE.pot*); it is generally named as *LANG.po* where *LANG* will be replaced by an ISO639-1 language code [18] when it must refer to the name a specific target language PO file. For example, *ru.po* is the name of a Russian language PO file.

Then the *Comparing and refreshing* task executes the *msgmerge* tool to refresh an already existing *Target language PO file* by comparing it with the up-to-date *PACKAGE.pot* file. The *Source code translating* task is similar to the *Editing PO mode source code* task in the flow diagram shown in Fig. 1, but it is a task of translating the PO files into the target language PO files using *poEdit* tool (see more description about the *poEdit* tool below in Section C). The *Generating MO files* task turns the target language PO files into files of machine-oriented (MO) format.

Finally, the marked software sources code is compiled and linked with the GNU Gettext libraries. This task is automated with Perl [19] scripts managed by the *Perl* tools. This will result in an executable software installed somewhere that users will find it.

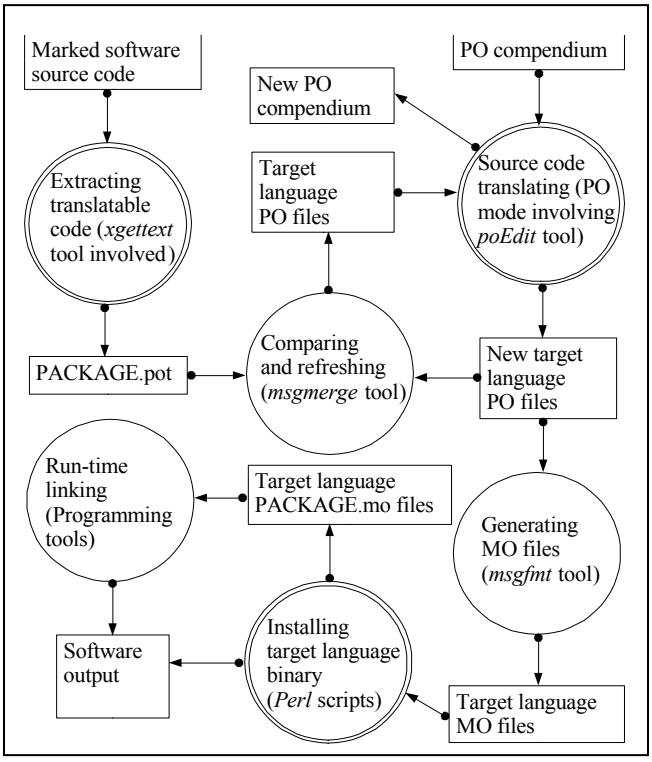

Figure 2. DMM information flow sub-graph of the execution environment for i18n of software.

B. The Archive Retrieval Tools

The archive retrieval a tool is the building block of the framework. With the tools, the framework must define how to use them and in what order to use them. The first refers to task and the second to flow. The tools in each category are described as follows.

- Programming Tools: These are found with Microsoft Visual Studio .NET 2003, mainly the C/C++ compiler and linker. Programming tools includes creating and compiling source code and debugging and converting the source code into libraries, components and executables.
- Internal Tools: These are mainly the GNU Gettext utilities. Once the GNU Gettext is installed in the execution environment, these tools are available for the i18n framework to use. The i18n framework reported in this paper only uses following internal tools and invoke each tool with standard commands.
  a) xgettext tool creates the PO template file *PACKAGE.pot*.
  b) msginit tool creates a new target language PO file *LANG.po* from the PO template file *PACKAGE.pot*.
  c) msgmerge tool updates an existing target language PO file *LANG.po* based on a newer version of the PO template file *PACKAGE.pot*.
  d) msgfmt tool generates binary MO files.
- Manipulation Tools: GNU Gettext also provides a range of manipulation tools to manipulate PO files in a way that is better performed automatically than by hand. The i18n framework does not use these PO file manipulation tools. Instead, it uses following two open source tools: (1) *poEdit* is a cross platform message catalogues editor which is software for manipulation and translation of PO files. (2) *SciTE* is a text editor that is specialized in source code manipulation and it is used in the i18n framework for text file editing and for manipulating source code and Perl scripts. (3) Another set of tools used is from Perl [19] which creates and runs Perl scripts to automate the i18n tasks such as *Generating MO files* and *Installing target language binary*.

C. Archive Translation and Interpretation

The definition and manipulation of tasks enable a framework to carry out planning without users knowing the details of operations and how they are implemented. Tasks are performed by invoking specific processes. A task could be, for example, encoding Unicode to transform character sets for supporting multilingual languages. For the i18n framework as illustrated in Fig.1 and Fig. 2, the tasks to be performed are non-terminal and terminal. The non-terminal tasks include:

1) Designing and coding sources is a combination of tasks and processes for creating and programming source code. Tools used in this task are programming tools.

2) Editing PO mode source code marks the source code according to the GNU Gettext convention toward i18n.

3) Compiling and debugging is normally to generate the source code into executable software and components which do not necessarily have i18n.

 ISBN: 978-1-61208-124-3

4) Installing and executing software is to arrange the software, data files, linked libraries, the source files and documentation in organized filing structures so that they can be accessed by all i18n tools.

5) Extracting translatable code is to find translatable code or strings from all source files and then to generate the PO template file. This task mainly uses the xgettext tool.

6) Source code translating performs all translation of the marked code and strings in the source code from the common language (English) to the target languages. It is carried out using poEdit tool.

7) Installing target language binary is automated with Perl scripts. This task compiles each target language source code and then links with a MO file into a linked library (dynamic linked library) and installs the results in a specific filing location.

The i18n framework has following three terminal tasks:

1) Comparing and refreshing uses the tool *msgmerge* to update PO files whenever the source code is changed.

2) Generating MO files uses *msgfmt* tool to generate MO files from target language PO files. This task can be integrated with the *Installing target language binary* task using Perl scripting to form an automated process.

3) Run-time linking is carried out during testing and debugging phase of the final software. It will execute the software by run-time linking the library generated by the *Installing target language binary* task for a specific target language. For CJK (Chinese, Japanese and Korean) and other Unicode languages, appropriate font files must be made available for this task to accomplish i18n (generating font file is beyond scope of this paper).

D. The Retrieval Process

There are three general processes involved in the i18n framework. The first one is for preparing source code; the second is for translating the source code; the third is for run-time generating linked library. Each process is of course a combination of tasks and sub-processes.

The first process includes creating and marking the source code. This process is formed by the tasks in Fig. 1. The second process involves in creating and updating the template PO file and editing, updating and manipulating PO files. The third process generates for each target language PO file a target language MO file and then creates a run-time link library for that target language. After the source code is compiled into executable software, this run-time link library enables the software with i18n. This process is the combination of tasks *Installing target language binary*, *Generating MO files* and *Run-time linking* as shown in Fig. 2.

V. SYSTEM IMPLEMENTATION

The framework shows how the complete i18n processes could be carried out. It is certainly not the only possible solution, but it provides a skeleton for real i18n packaging, i.e. the key tasks that have to be accomplished.

A. System Requirements

The key requirement is a version of GNU Gettext Tools (Version 0.14.4 for this paper) which is essential for i18n. All other internal tools (see the above Section IV) must be installed as part of the execution environment. It must be noted that *Perl* tools should be installed and program compiler and linker tools should also be made accessible in command line mode if *Perl* tools and Perl scripts are used to automate some of the i18n tasks in the i18n framework.

B. Source Preparation

This is about programming or creation of software source code. In this paper, it is C/C++ source. Bringing GNU Gettext convention into software package is to identify in the sources those strings which are meant to be translatable and those which are untranslatable. Beside this, some simple and standard changes are needed to initialize the GNU Gettext library. Changes to the source code fall into three categories. First, the programmer has to make the localization functions known to all modules needing message translation. Second, the programmer should properly trigger the operation of GNU Gettext library when the program initializes, usually from the main function. Last, the programmer should identify and especially mark all translatable strings in the source code [7].

As illustrated in Fig. 1, from the *Designing and coding sources* task to the *Marked software source code* specification, all work should be achieved by programming tools. From this point, the source code is marked according to the GNU Gettext programming convention and can be either compiled into package without i18n or brought to the next stage for translation toward i18n (see Fig. 2).

C. Generating PO template File

Once the source code has been modified and marked, the *Extracting translatable code* task makes invocation of the *xgettext* tool. This tool will find and extract all translatable strings from the source code files and then create the PO template file *PACKAGE.pot*. Below is a typical example of the invocation of the *xgettext* tool:

xgettext -a --files-from=POTFILES.in -o PACKAGE.pot

where *-a* is an optional command to instruct the *xgettext* tool to extract all marked strings from the source code files that are listed in the input file *POTFILES.in*. The *--files-from=POTFILES.in* lets the *xgettext* tool to read the names of the input source code files from the *POTFILES.in* file. The *-o* optional command makes the *xgettext* tool to write output to the file *PACKAGE.pot*.

D. Using PO Files

PO files store the translations; every target language should have a single PO file. For instance, a Chinese translation process will translate a PO file into a Chinese (target language) PO file [18]. Normally, a translated PO file contains previous translations provided by the translators. The update of a PO file is done by the *msgmerge* tool. Following is an example of the invocation of the *msgmerge* tool to update the *zh.po* file:

msgmerge -u zh0.po PACKAGE.pot –output-file=zh.po

 ISBN: 978-1-61208-124-3

where *-u* is to instruct the *msgmerge* tool to update the old version of PO file, *zh0.po*, into a new version *zh.po* in reference to the new version of file *PACKAGE.pot*.

E. Generating MO files

For each PO file (in corresponding to each language), the i18n framework will generate one MO file via the *msgfmt* tool. Below is an example that shows how the *msgfmt* tool is invoked:

msgfmt -c zh.po --output-file=zh.mo

where *-c* instructs the *msgfmt* tool to perform a check on language dependent format strings, contents of the source file header entry and conflicts between domain directives and the *--output-file* option. As the generated *zh.mo* file is a binary file, it is ready for the programming tools (compiler and linker) to use in generating a run-time link library such as the dynamic link library *zh.dll*. It should be pointed out that the i18n framework described above is only necessary for software package maintainers or developers. End users do not have to perform any of the tasks showed in Fig. 1 and Fig. 2.

VI. CONCLUSION AND FUTURE WORKS

The i18n in an archive retrieval system does not mean that information is simply translated via externalizing its user interface. It is a process that demands for a collaborative effort from managers, designers, programmers, translators and it depends on users' feedback for further development. Every software, module or component needs i18n, but i18n is not a simply a software translation. Translators may not be able to choose the best phrase in the target language for any text that may possibly be seen by an external user, that is, error messages, help messages and the like. Archive translation software should be concurrently carried out within the whole software development cycle so that translators are able to work within the context about the software and the project. Software that supports Unicode is not necessarily an i18n system or product. Unicode is a coded character set. Only characters or parts of characters are encoded, but there is no information about language, locale and font. If a software product can support Unicode, it only recognizes single characters. Unicode is for supporting languages around the world, but it is not a panacea for i18n.

Further works are needed to (1) implement a language code interpreter, (2) increate the vocabulary in the indexing database and (3) test and validate the reliability of the system tools.

ACKNOWLEDGMENT

The work carried out in this paper is partially funded by the China National Science Funding Council No. 60873208.

REFERENCES

[1] Wikipedia, "Cloud Computing". The free encyclopedia http://en.wikipedia.org/wiki/Cloud_computing. Retrieved November 2010.

[2] T. Claburn, "Google's "Gov Cloud" Wins $7.2 Million Los Angeles Contract". The Internet On-line Resources: Informationweek.com http://www.informationweek.com/news/services/saas/showArticle.jhtml?articleID=221100129. Retrieved August 2010.

[3] NIST, "Long Term Knowledge Retention (LTKR): Archival and Representation Standards", Gaithersburg, MD 20899, March 2006.

[4] B. Swanson, "The Coming Exaflood", *The Wall Street Journal*, http://online.wsj.com/article/SB116925820512582318.html. Retrieved August 2010.

[5] Z. X. Zhao and L. Z. Zhao, 2008, "Small-world phenomenon: toward an analytical model for data exchange in Product Lifecycle Management", International Journal of Internet Manufacturing and Services, Vol. 1, No. 3, pp. 213-230.

[6] The ACC Localization Advisory Board, "ACC Globalization Internationalization Localization", Austin Community College, http://www.austincc.edu/techcert/accGIL.html. Retrieved March 2009.

[7] U. Drepper, J. Meyering, F. Pinard, and B. Haible,, "Native Language Support Library and Tools, GNU Gettext Tools, Version 0.14.4 (Edition 0.14.4, 8 March 2005", GNU Project - Free Software Foundation (FSF), Free Software Foundation, Inc., http://www.gnu.org/software/gettext/. Retrieved August 2009.

[8] M. Perrow, "Editor's Notes: Better Software Require A Better Process", The Rational Edge, January, 2001, pp. 1-2.

[9] J. H. Dunning, "Recent developments in research on multinational enterprises: an economist's view". In Lars-Gunnar Mattson and Finn Wiedersheim-Paul (eds), Recent Research on the Internationalization of Business. Proceedings from the Annual Meeting of the European International Business Association, Uppsala, Sweden, 14-17 December 1977. Uppsala: Uppsala University, pp. 1-16.

[10] J. H. Dunning, "Location and the multinational enterprise. A neglected factor", Journal of International Business Studies, Vol.29, No.1, 1998, pp. 45-66.

[11] A. Heinrich, "Internationalisation, market structures and enterprise behaviour. The Janus-faced Russian gas monopoly Gazprom". In Kari Liuhto (ed.), East Goes West. The Internationalization of Eastern Enterprises. Lappeenranta: University of Technology, 2001, pp. 51-87.

[12] J. L. Nhampossa, and P. Nielsen, "Experiences of internationalizing Information Systems: The challenge of standardization", Presentation at the Oslo/Cambridge IS Workshop at University of Cambridge, UK. http://www.hisp.info/confluence/download/attachments/2374/cambridge_2004_nielsen_nhampossa.pdf? Retrieved December 2007.

[13] A. Vine, "All Things International, only Some of Them Software", The I18n G. A. L, Sun Microsystems, Inc., http://blogs.sun.com/i18ngal/. Retrieved September 2008.

[14] S. Kleinfeldt, M. Guiney, J. K. Miller, and M. Barnes, "Design methodology management", Proceedings of the IEEE, Vol.82, No.2, 1994, pp. 231-250.

[15] R. A. Baldwin, and M. J. Chung, "A formal approach to managing design processes", Computer, IEEE Computer Society, February, 1995, pp. 54-63.

[16] Z. X. Zhao, "A methodology management approach to computerised process planning", International Journal of Computer Integrated Manufacturing. Vol.10, Nos1-4, 1997, pp. 83-91.

[17] K. W. Fiduk, S. Kleinfeldt,, M. Kosarchyn, and E. B. Perez, "Design Methodology Management - A CAD Framework Initiative Perspective", Proceedings of the 27th ACM/IEEE conference on Design automation, 1991, pp. 278-283.

[18] ISO639 Joint Advisory Committee, "Codes for the Representation of Names of Languages", ISO639.2, November 2006, http://www.loc.gov/standards/iso639-2/php/English_list.php. Retrieved March 2009.

[19] ActivePerl, "ActivePerl: Version5.8.8.817", ActiveState Software Inc. http://www.activestate.com/Products/ActivePerl/. Retrieved October 2010.

 ISBN: 978-1-61208-124-3