The Software for Cultures and the Cultures in Software *

Gregory E. Kersten[1], Stan Matwin[2], Sunil J. Noronha[3] and Mik A. Kersten[4]

[1] DSMIS, Concordia University, Montreal, Canada (gregory@mercato.concordia.ca).
[2] SITE, University of Ottawa, Canada (stan@site.uottawa.ca).
[3] IBM T. J. Watson Research Center, Yorktown Heights, USA (noronha@watson.ibm.com).
[4] Xerox PARC, Palo Alto, USA (mkersten@parc.xerox.com).

***Abstract*-Software is viewed as an artifact which interacts with cultures of societies in which it functions. Software manufacturers make efforts to adapt the appearance of their products to aesthetic and historical values of the markets in which they are sold ("software for cultures"). It is well known that software embeds behavioral and organizational principles that are culture-determined ("cultures in software"). Internet and e-commerce bring these phenomena into the fore of the debate on societal implications of Information Technology. The paper argues for a research agenda on the multifaceted interactions between software and culture.**

I. INTRODUCTION

Software, like any other product family, contains embedded cultural values and objectives. Some of the embedding occurs unconsciously, inherited via the cultural programming of its human creators; other parts of it are intentional via design requirements explicitly obtained by researching its target markets. A third form of embedding emerges through group and organizational cultures, often reflecting the organizational structures and incentives that control the creation of large pieces of software. The Linux and Windows operating systems are striking examples of the latter, reflecting the importance given to cultural values such as openness and flexibility versus ease of use and stability.

Explicit consideration of cultural factors does not play a significant role in current software engineering practice. Market demand and competition leads firms to concentrate on application features and the reduction of time to market, at the expense of enabling the functionality to work well in the different environments in which it will get used.

Organizations are unwilling to invest the resources to adapt and differentiate their products to multiple cultural markets, unless adequate returns are posited. Some markets are ahead of the others, and therefore products (or product interfaces, e.g., shopping metaphors in e-commerce servers) are designed to address the needs of the dominant culture. Features supporting other cultural needs are added only if they distinguish the product sufficiently to succeed in attracting another niche market. Often the cost of redesign is considered too high, and as a result only the immediately visible layer of the software is considered.

In this paper we discuss the relationships between culture and software in the framework of the critical theory of technology [1]. The discussion takes several different perspectives with the expectation of providing a justification for a call for theoretical and applied research on the two types of these relationships. The first type (i.e., "the software for cultures") refers mainly to the customization of software and to the development of different software for different markets. For example, the need for user interface customization has widely been recognized. However despite the fact that many applications that have their roots in a particular culture exist, the development of software that differs in its core to meet different cultural values, ideas and procedures has attracted little attention.

This type of relationship (i.e., "the culture in software") only partially reflects the cultural roots of different applications. It is based on the premise that an increasing number of applications represent, and act on behalf of, groups and organizations. This software has some degree of autonomy, can interpret and make decisions, and interact with other applications, people and organizations. The first part reflects the anthropologist's viewpoint of how the culture-specific behaviors and assumptions *get embedded* in software, and what the resulting conse-

* We thank the three reviewers for their comments and suggestions. This work has been partially supported by the Social Science and Humanities Research Council of Canada and the Natural Sciences and Engineering Research Council of Canada.

quences are. The second part posits on what cultural norms and procedures should be embedded in software, and in how they *should be* embedded These two parts lead to the call for an interdisciplinary and cross-cultural discussion and research program on writing software for different cultures and embedding different values and ideas in software.

This paper draws on some observations and concepts developed in studies of technology. There is no solid theory that links software and culture, or the way ideas and values are implemented in software. Such a theory is required and needs to go beyond the consideration of the surface manifestations of culture that have been widely accepted in software internationalization methodologies and address the core components of software that, we believe, influences our ideas and values.

II. Technology and Culture

A popular view of technology is that it is neutral and indifferent to the variety of ends towards which it can be employed[1]. This instrumental position on technology has been characterized as uncritically positive and it has been contrasted with a substantive theory according to which technology constitutes a new type of cultural system that restructures the entire social world. Pacey argues that a technocratic value system is single-minded and insistent on an unambiguous view of progress, collaboration, problem solving, and values [2].

Feenberg advocates a critical theory in which technology is a process of development suspended between different possibilities—a process in which social values and ideas are attributed in the design and development, and not merely the use of technical systems [1, p. 14]. The two tenets of the critical theory are that (1) technology may be used to advance and enrich social objectives, and (2) technology cannot be seen as separate from people.

The critical theory of technology forms the basis for this discussion. It gives grounds for our claim that a theory of software needs to be formulated so that software engineering objectives can reflect social objectives. Software has a unique place in the world of technologies because it often is a particular embodiment of methods, knowledge and even philosophy. The importance of computational methods, information and knowledge has pushed forward the development of Information Technology (IT). IT in turn has provided resources that change the individual, organizational and national environments. By creating and manipulating information and meanings people and organizations can frame the world to reach goals with IT mediating the interpretation of the world [3]. In this context culture affects the information interpretation and framing.

Culture is a concept for which there are as many definitions as there are scholars studying it; Kroeber and Kluckhohn [4] list over 160 definitions and many more have recently been proposed. Culture, according to Hofstede [5], is the *software of the mind*. In explaining culture and its manifestations Hofstede uses the analogy of computers and programming and says that “Culture is always a collective phenomenon, because it is at least partly shared with people who live or lived within the same social environment where it was learned. It is the collective programming of the mind that distinguishes the members of one group or category of people from another." [6, p. 5]. As such, culture is a set of shared and enduring meanings, values, and beliefs that characterize national, ethnic, or other groups and orient their behavior [7]. These and other definitions of culture make us believe that the software is congruent with the culture of individuals, or small groups, that create it.

Culture is an aggregate product of the processes occurring at the social, organizational and individual levels. It includes beliefs, ideas, language, rules, procedures and norms, and it manifests itself in artifacts and objects ranging from art to organizational structures to clothing and cars. Software and hardware companies have realized that in order to sell in the foreign markets they have to adapt their products. However, these modifications are done—as we describe in Section 3—on the user interface level with the core assumed culture neutral.

III. The software for cultures

A. *Embodiment of cultural context in software*

The instrumental position on technology leads to the introduction of cultural aspects in the external layer of software; it is an interface between the hardware, the software engines and data and the user. The development of software for international markets necessitated this external perspective. The specifics of international software represent "slices of the culture for which they are targeted" [8, Chapter 4].

Software needs to fit the cultural context of the user, however, this context has been defined solely in terms of the requirements regarding the user interface. In his answer to the question "What then needs to be encapsulated in this concept of cultural context?" Taylor lists the following locales, i.e., the collections of all the conventions that characterize a particular culture or user community: transliteration, hyphenation, spelling, collation, national conventions (numbers, currency, time and date), and color (op. cit.). Hall and Sauter add such elements as messages, terminology, and positioning of windows, tables and graphs [9, 10].

Methods for translating software from their source market to the target markets have been developed and implemented in many products. These methods are based on three topics: (1) the choice of character codes; (2) the use of locales; and (3) the use of resource files [11].

The premise behind the external perspective is that "all the culturally and linguistically sensitive software components need to be separated from the core of the application" [11, p. 298]. Following this premise software internationalization architectures have been proposed in which the locale sensitive elements are separated from the locale independent core [9]. An application programmer interface (API) is used to implement locale elements.

Many ubiquitous hardware and software systems emerging from a small group of manufacturers in the US (e.g. PC equipped with Microsoft Windows/Office) have imposed standards which other system developers all over the world feel compelled to follow. This has a double effect. On the one hand, prospective users are encouraged by the familiar look of a new software product. For example, an investment advisor system probably has a greater chance becoming commercially successful if it uses a spreadsheet paradigm familiar in the world of finance than some other, potentially better, input/output paradigm. This makes it easier to introduce new functionality with a afamiliar look-and-feel and mode of use.

Many potentially superior solutions, on the other hand, are never introduced because they diverge from the “de facto standards”. It is well known that a mouse uses only a small part of the dexterity of a human hand, but no new input devices have come close to the commercial success of the mouse. The way that Internet search engines present an answer to a query is a case in point: most search engines rank the retrieved documents according to their distance form the query, computed by a function which is internal to the engine. The users could likely benefit from a graphical display of these documents as points on a line of a plane, where this distance would naturally help with the selection of the documents the user wants to see, but the risk of rejection of a novelty is too big for search engine developers.

The cultural and communicational artifacts (e.g. pull-down menus and bullet-style presentation templates) have propagated from software to television, becoming truly global (e.g. the CNN method of displaying text accompanying a news clip). Moreover, certain communicational conventions that emerged on the Internet have made their way to everyday written language, and not only English. “Emoticons” (e.g. -:)) and abbreviations (IMHO, AFAIK) have become communication clichés. The word “bookmark” migrated back from the Internet jargon to spoken English with a different meaning then what was used before the introduction of browsers (e.g., “Let me bookmark this meeting.”). These are just few examples of the influences of software developers’ culture on other cultures and they reflect the “Silicon Valley perspective” that technology, organizational and business models, and market structures originate in, and inevitably spread from, Silicon Valley.

B. Use of software in different cultures

The modification of user interfaces for customers is an obvious attempt at considering cultural needs and is practiced by many businesses. No car manufacturer, for example, attempts to enter a market without translating manuals and positioning dials according to local regulations and customs. Occasionally, local markets require changes in the core of the design, for example moving the steering wheel or adding environmental protection components. There is also another significant difference that distinguishes car manufacturers from each other: by selling cars they also market and sell cultural artifacts. Many customers buy American, German, Italian or Swedish cars because they embody their respective ideas and norms. The differences between these cars go beyond the "interface" and manuals, and into the very core of the engine, chassis and suspension.

Methodologies for “internationalizing” software distinguish between core and customizable components. These components address only surface manifestations of cultural differences. The assumption is that the core of is independent of the user's cultural context. Indeed, this might be the case with respect to systems used for document preparation and management, database operations, OLAP and data visualization. There are many other applications that implement the principles of problem understanding, communication and interpretation. Deeper manifestations of cultural differences that influence ideas, values and beliefs are influenced by the core of systems that implement organizational standard operating principles (SOPs) and methods developed or adopted by manage-

ment consultants are heavily culture-dependent. As an example, consider three schools for decision support: American, English and French, and, respectively, three examples of typical DSSs: Expert Choice, Decision Explorer and Electre. Each of these systems embodies characteristics that are particular to the culture in which they were created.

The American Expert Choice application implements a "measurement philosophy" in which every complex problem can be reduced to a number; hence every alternative course of action can be measured and directly compared with another.

The English Decision Explorer is based on the assumption that the search for the solution of a complex problem is equivalent with the finding its appropriate representation. If one understands the problem and its potential implications, and can view its representations from different perspectives and in varying levels of complexity, often the solution becomes obvious.

The French Electre systems are based on the concept of reduction of incomparability between actions through the use of outranking relations that represent decision-maker's preferences. Effort is made to exploit the relations between actions gaining information allowing for the determination of their partial comparability [12]. These three systems are examples of a larger and very different families of systems that can be used for the same problems but require a very different cultural background and problem solving approach on the part of the user.

Claims have been made that formalized approaches to decision-making, many of which form a core of DSSs, do not differ and are not a function of culture [13, 14]. The reasons for such claims may be due to—as Carmel notes—the global market domination of U.S. based companies in packaged software [15]. The above three examples, as well as many other studies indicate that there are significant differences in the software developed in different national and organizational cultures.

We have conducted a study in which over 2,000 people from 79 countries used INSPIRE, a Web-based negotiation support system, to conduct anonymous bilateral negotiations [16]. It is obvious that approaches to negotiations as individual decision-making are rooted in culture. This is also the case with the INSPIRE users; their use of the system differs with respect to their national, organizational and professional culture. For example, most of the users from India are engineers and they prefer to communicate using offers without supporting arguments. Business students from Ecuador, on the other hand, attempt to establish a friendly relationship with their counterpart that requires informal messages in addition to offer-supporting arguments.

The analysis of the INSPIRE negotiation transcripts also shows that different cultures require different types of support for preference elicitation, offer selection, analysis and comparison, and concession-making. They have different expectations, needs for communication and collaboration, risk attitudes, and assertiveness. These characteristics have been thoroughly studied, and it seems only natural that the software, which supports processes of a psychological and social nature, takes them into account. The fact that software of which the core is also culture dependent is rarely available does not diminish the need but it is—we believe—an example of the immaturity of the software industry and the dominance of three elements particular to “culture software” in the US, namely: the individualistic dimension, the entrepreneurial dimension and the risk-taking ethos that embraces ad-hoc, innovation-driven development [15].

IV. THE CULTURES IN SOFTWARE

C. Presentation and customization

As a result of its global access and Silicon Valley roots, e-commerce magnifies the phenomenon of North-American cultural clichés inserting themselves in other cultures. E-commerce sites push artifacts to cultures in which they may be foreign or exotic (e.g. shopping cart and auction metaphors for online purchasing). While many artifacts will permeate to other cultures, we argue that e-merchants will have to develop conceptual interfaces fitting other cultures.

The current situation, in which the most popular software products are a result of a top-down, hierarchical corporate development, is likely to change in the light of the recent Microsoft ruling and other developments. It is quite likely that a more competitive software development market will emerge in the near to middle-term future. Moreover, growing popularity of competing modes of software development, testing and maintenance witnessed by the GNU Open Software Foundation and Linux may influence how software is created. It is possible that the

future model will evolve, in which technically competing proposals are put forward and unified by open, international standardization bodies which include representatives from several major cultural areas of the globe.

E-commerce and globalization of business allows businesses to sell, and customers to buy, on any market. The markets become borderless, but this does not imply that the sellers' and the buyers' cultural traits disappear. The "store windows" of e-commerce firms reflect their national and organizational cultures. Subsidiaries of decentralized firms create their Web-sites independently of each other and with different visual presentation and content. Centralized companies, on the other hand, attempt to present a standardized image.

An instance of this is demonstrated in the fact that home pages of Sony Corporation are different in Europe, Latin America, North America and Japan (compare http://www.latin.sony.com, http://www.sony.co.jp, http://www.sony.com and http://www.sony-europe.com) [1]. The difference is not only in language, color and graphics but also in the navigation model, architecture, and marketing strategy employed for the particular market. On the surface "Sony-on-Line" have a very different look-and-feel. For example, for the homepage for Latin America countries has several active graphic components, and the first link is to the company history and profile page in a given country. Pages for European countries are static or contain minimal components, and provide no information about the company.

E-commerce grows rapidly and companies make significant efforts to attract new customers. At present, e-commerce market penetration is very small and most of the customers come from the "Internet community". It is possible that they represent an emerging culture but there is a lack of evidence pointing to the fact that everybody will embrace this culture and that local organizational and national cultures will disappear. In order to attract new customers firms have to take into account their individual values and needs, including those rooted in culture. The customization and personalization that has attracted so much attention needs to address these issues.

At present customization is oriented mostly towards a small segment of the U.S. market. U.S. firms use software developed by U.S. companies to attract U.S. customers, and therefore the specificity of culture can be, and is ignored. Marketing, sales and other business transactions are done differently in different places requiring that companies, which want to enter new markets, adapt to local customs, values and procedures of doing business.

There are many stories about business failures caused by the lack of understanding of the local cultures. It is now common knowledge that a company, in order to establish presence on a local market and relationships with local firms, has to learn the local culture and use it in its own business practices. E-commerce technologies allow the penetration of local markets but do not guarantee business.

Hoffman and Novak postulate that new technologies and a new business paradigm have to be constructed freeing customers from their traditionally passive role as receivers of communication and allowing them to become active participants in the marketing process [17]. Customers need to have control over the search and acquisition of information relevant for consumer decision-making. While this postulate refers only to the marketing function, the implication is that firms must understand their customers. Where the customers come from is as important as what they want. Understanding of the customers will then lead to the selection of software components that reflects business practices in the local marketplace.

Depending on the type of product and service the role of the local culture differs. As a result, ales of software and hardware can be done without a significant cultural component. Internet sales of clothing, home furniture, [MIK: something is missing in this sentence..] provision of consulting or training services have to take into account the culture of individual and organizational customers that goes beyond the interface and virtual store window. This leads to the development of software components and methodologies for putting these components together for a given market leading to *culture aware software*. These applications will support such cultural characteristics as shopping patterns, expectations about prices and bargaining, and values that affect customer satisfaction. E-commerce applications are but one example of the need for software exhibiting culture-specific behaviors and assumptions.

D. Representation and support

IT in any organization embeds behaviors that constitute organizational culture. Business processes need to be implemented and many applications provide facilities to adapt specific accounting, financial, and logistic and operations practices. Emergence of the active DSSs, software agents, meeting systems and brainstorming tools,

[1] These sites were compared in December 1999.

knowledge management systems and other programs that contribute to the behavioral patterns of individuals and organizations changes the scene in that these systems change the individual and collective "software of the mind".

The cultural aspects of IT have been discussed in research on computer-mediated processes, including group support, collaborative work and negotiations [3, 16, 18]. However, the relation between the cultural assumptions embedded in technology and national and organizational cultures have not been studied.

Consider the case of the negotiation software agents developed at the MIT Media Lab [19, 20]. These agents negotiate on behalf of their principals: people who want to buy or sell goods. It is well known that negotiation processes differ and are culture-dependent, and that negotiators have to be sensitive to the cultures of their counterparts. The agents however, do not exhibit any of these sensitivities nor are able to engage in different negotiations.

The use of software agents will increase; they will represent firms and engage with other agents representing people and other firms in business transactions. They will be used in teaching, construction of knowledge bases and knowledge management, problem solving and decision-making. If there are significant discrepancies between the cultural assumptions that are (consciously or subconsciously) built into systems and cultural influences on power, politics, structures and information network of organizations the systems will often be rejected.

The issues of the relationships between this culture and the user culture have to be considered. IT is not a static, neutral and objective entity, but it is interpreted, used and shaped in a context [21]. Software developers need to take the context in which the software will be used into account and build systems that either have -context orientation or can be adopted to different contexts.

V. RESEARCH DIRECTIONS

E. Design issues

Software design is a process that involves the composition of many interoperating pieces, called modules, into a functioning system. When these individual modules are engineered, culture-specific concerns become embedded within them. For example, the graphical user interface in the Windows NT operating system relies on the cultural convention that text is written and read from left to right. If this is not the case, as is true in Hebrew writing, functionality such as displaying text, scrolling, and highlighting no longer behaves appropriately.

Internationalization software, such as the Java™ Internationalization API facilitates the adoption of some cultural concerns such as text formatting. However, cultural factors affect much more than the system's user interface, and cultural adaptability of software needs to go beyond the presentation layer and impact its functional design as well. For example, the different Sony sites mentioned above not only present a user with completely different interfaces that employ different languages, layouts and effects, but also vary their e-commerce model in both the policy that it uses to log users in and the method in which it presents material to them: the American version employs the cultural archetype of a shopping, and uses a catalogue layout and navigation that is drastically different from that employed by the Japanese version. Many different parts of the software, such as the user interfaces, the e-commerce transaction processing systems, and the databases are involved in describing the cultural conventions that the system adheres to.

Evolving a software system to be deployable in different cultural domains is challenging due to the fact the cultural concerns are embedded throughout the software and are not captured in a single changeable place. As a result, in order to change a cultural factor a large number of interrelated parts of the system must be altered—this happens because the cultural convention concern crosscuts the system structure.

Current software development paradigms such as procedural and object-oriented programming fail at capturing units of software modularity that crosscut module boundaries [22]. However, a promising technology called aspect-oriented programming is being developed at Xerox PARC, and has been demonstrated to capture crosscutting concerns in new units of software modularity called aspects [23]. Since cultural concerns crosscut system architectures, aspect-oriented programming may prove to be a valuable method of capturing these concerns. Aspect selection and configuration can be controlled with a rule-based infrastructure by employing languages like AspectJ (see http://aspectj.com), an aspect-oriented programming extension to Java.

F. Organizational issues

Software development technologies that allow for high level modularization, reuse and reconfiguration are necessary for the development of culturally sensitive applications. Production of such applications requires cultural sensitivity of the part of the software firms and their employees. In present situation highly trained computer scientists regularly underestimate the importance of user-centered design. It is then not surprising that concepts further afar, such as cultural factors from the social sciences, are ignored.

Organizational culture, for example, is a key determinant of success in software reuse. While object-oriented software practitioners have described how their practices enable reuse and the significant benefits that result, it has been observed that the behaviors of many software organizations indicate a low valuation of reuse.

The problem is that reuse is predominantly a socio-political issue, but the prevailing development culture focuses on technical solutions to problems rather than cultural ones. This is bound to be unsuccessful since incentive structures often reward fast independent development and ownership of intellectual property, rather than investment in the extra design efforts needed to support reuse. Indeed, reuse may actually have a negative image. Increasing the amount of reuse in an organization requires resetting cultural values all the way from executive management to the ground staff.

G. Research Agenda

The relationship between software and culture cuts deeper than the interface. Therefore a research program into this important issue should be developed. Such a program should must be interdisciplinary: IT, ergonomics, philosophy, sociology, anthropology, and, yes, art. This program would identify the factors and phenomena that are involved in the influence of software on culture, and in the influence of culture on software, e.g. what cultural factors are likely to influence software. The research should go beyond the surface manifestations of culture, and focus on cultural factors that influence values, beliefs and behaviors, and which are influenced by the core components of software. This could result in “cultural testing tools”, i.e. rule sets characterizing specific cultures against which some types of software (e.g. groupware) could be run to detect potential conflicts and inconsistencies.

One of the outcomes of such a program will be a framework in which culturally sensitive features are organized into a "best practices" collection of guidelines/recommendations for software design. Such a framework would be useful from both a research and from a business viewpoint if it increased the speed of of internationalizing products (or more generally, expanding from one cultural market to another, even within the same country). A deepening of the "locale" concept is one way to proceed, capturing those cultural attributes that can be precisely measured and represented. Such an extended locale can be treated synonymously with "cultural profile", a subcomponent of the "user profile".

On one hand, the requirement is to provide the "intelligence" that diagnoses a user as belonging to a given culture (profiling), and the business knowledge that determines how what actions should be taken by the software in order to adapt to the culture. From a software implementation perspective there appears to be a heavy intersection with personalization technology. Another requirement is to strengthen the pluralistic and culturally sensitive software development efforts.

The participants of the 1999 E-conomy conference call for plural views on e-commerce as opposed to the Silicon Valley or US-centric view. Software and hardware technologies developed for e-commerce already have other applications, including communication and discussions, voting and other forms of citizens’ participation in municipal affairs. One may argue that these other activities are even more culturally rooted than business. Presently they are largely driven by the U.S. technologists, or—as they now call themselves—software evangelists. The plural view on e-commerce can be achieved, according to some conference participants, through, government regulation and laws. Such solutions are, at the very least, inadequate; they will freeze the country that adopts them if not accompanied with technologies that allow individuals, businesses and other organizations undertake activities on their own terms and according to their own values.

As the critical theory of technology posits, individual, organizational and national values and ideas are embedded in software. We call here for: (1) the identification of the current values embedded in software, (2) the discussion on the relationship between software and values, and (3) the development of tools that allow the manipulation and testing of the embedded values and ideas in software.

REFERENCES

[1] Feenberg, A., *Critical Theory of Technology*, New York: Oxford, 1991.

[2] Pacey, A., *The Culture of Technology*, Cambridge: MIT Press, 1992.

[3] Poole, M.S. and G. DeSanctis, "Understanding the Use of Group Decision Support Systems: The Theory of Adaptive Structuration", in *Organizations and Communication Technology*, C. Steinfield and J. Fulk, Editors, Sage: Newbury Park. 173-193, 1990

[4] Kroeber, A. and C. Kluckhohn, *Culture: A Critical Review of Concepts and Definitions*, New York: Random House, 1963.

[5] Hofstede, G., "Cultural Predictors of Negotiation Styles", in *Process of International Negotiations*, F. Mautner-Markhof, Editor, Westview Press: Boulder, CO. 193-201, 1989.

[6] Hofstede, G., *Cultures and Organizations: Software of the Mind*, 2nd ed, New York: McGraw-Hill, 1997.

[7] Faure, G.O. and J.Z. Rubin, eds.,*Culture and Negotiation. The Resolution of Water Disputes*., SAGE: Newbury Park, CA., 1993.

[8] Taylor, D., *Global Software. Developing Applications for the International Market*, New York: Springer Verlag, 1992.

[9] Hall, P. and R. Hudson, *Software without Frontiers*, New York: Wiley, 1997.

[10] Sauter, V., *Decision Support Systems*, New York: Wiley, 1997.

[11] Hall, P., "Software Internationalization Architectures", in *Decision Support Systems for Sustainable Development in Developing Countries*, G.E. Kersten, Z. Mikolajuk, and A. Yeh, Editors, Kluwer: Boston. 291-304, 1999.

[12] Martel, J.M., "Aggregating Preference: Utility Function and Outranking Approaches", in *Multicriteria Analysis*, J. Climaco, Editor, Springer: Berlin. 76-85, 1997.

[13] Al-Jafaray, A. and A.T. Hollingsworth, "An Exploratory Study of Managerial Practices in the Arabian Gulf Region", *Journal of International Business Studies*, 14: 143-152, 1983.

[14] Neganshi, A.R., "Convergence in Organizational Practices: An empirical Study in Industrial Enterprises in Developing Countries", in *Organizations Alike and Unlike*, C. Lammers, Editor, Routledge: London, 1979.

[15] Carmel, E., "American Software Hegemony", *The Information Society*, 13(1), 1997.

[16] Kersten, G.E. and S. Noronha, "Negotiations via the World Wide Web: A Cross-cultural Study of Decision Making", *Group Decision and Negotiations*, 8: 251-279, 1999.

[17] Hoffman, D.L. and T.P. Novak, "A New Marketing Paradigm for Electronic Commerce", *The Information Society*, 13: 43-54, 1996.

[18] Juustila, A., "Interaction of culture, power and IT in organizational change", in *Information systems research seminar in Scandinavia*. Gjern, Denmark, 1995.

[19] Guttman, R.H. and P. Maes, "Agent-mediated Integrative Negotiation for Retail Electronic Commerce", in *Workshop on Agent Mediated Electronic Trading*,1998.: (http://ecommerce.media.mit.edu/papers/amet98.pdf).

[20] Maes, P., R.H. Guttman, and A.G. Moukas, "Agents that Buy and Sell", *Communication of the ACM*, 42(3): 81-91, 1999.

[21] Orlikowski, W. and D. Robey, "Information Technology and Structuring of Organizations", *Information systems research*, 2(2): 143-169, 1991.

[22] Kiczales, G., *et al.*, "Aspect-Oriented Programming", in *European Conference on Object Oriented Programming*, Springer: Berlin, 1997.

[23] Kersten, M.A. and G.C. Murphy, "Atlas: A Case Study in Building a Web-based Learning Environment using Aspect-oriented Programming", in *ACM Conference on Object-oriented Programming, Systems, Languages, and Applications*. Denver: ACM Press 340-352, 1999.

[24] Cioffi, J.W., "The Digital Economy in International Perspective: Common Construction or Regional Rivarly", Report, May 1999 Conference of the E-conomy Project, University of California, Berkeley, http://economy.berkeley.edu/events/deip/summary.html, 1999.