A human activity approach to software localization

Bao-Tong Gu
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Business and Corporate Communications Commons, and the English Language and Literature Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/16136

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
A human activity approach to software localization

by

Bao-Tong Gu

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ARTS

Department: English
Major: English (Business & Technical Communication)

Approved:

In Charge of Major Work
For the Major Department
For the Graduate College

Iowa State University
Ames, Iowa
1994
# TABLE OF CONTENTS

I. INTRODUCTION...............................................................................................................1

II. PAST PRACTICE AND THE EXISTING LOCALIZATION MODEL OF
DESIGNING, WRITING, AND TRANSLATION.................................................................5

1. Translation: A Misleading Concept.........................................................................5

2. Designing, Writing, and Translation: An Inadequate Model....................................9

III. MY THEORETICAL MODEL OF LOCALIZATION.....................................................22

1. A Broader Perspective—Aspects of Localization.....................................................22

2. Human-Computer Interaction Design: The Key to Successful
Localization...................................................................................................................24

3. Activity Theory as a Foundation for HCI Design..................................................46

4. Activity Theory, HCI Design, and Localization.....................................................58

IV. CHINESE LANGUAGE KIT: A CASE STUDY..........................................................67

1. An Overview..............................................................................................................67

2. WorldScript and Language Kits..............................................................................68

3. Design Rationale and Target Users........................................................................71

4. Chinese Language Kit Examined from the Human Activity Perspective.................75

V. CONCLUSION...............................................................................................................87

BIBLIOGRAPHY.............................................................................................................91

APPENDIX: COMPONENTS OF CHINESE LANGUAGE KIT........................................97
ACKNOWLEDGMENT

I am deeply indebted to Dr. David R. Russell, my adviser, who in the past two years has offered me tremendous and generous help with my program of study, whose advice has made my academic life at Iowa State easier and more entertaining, and who has guided me through almost every step of the way in making this thesis possible. Without him, this thesis would have virtually been impossible.

I also owe my gratitude to my other two thesis committee members, Dr. Rich Freed and Dr. Johnny Wong, for their patience during my making of this thesis and their generous and helpful advice and suggestions, which have rendered this thesis much better than it would have been.

My thanks also go to Mr. Ker Gibbs, Product Manager of AppleSoft Products Group, who helped me with my comprehension of the software localization process and of Chinese Language Kit and whose materials proved to be of great help to my writing of the thesis.

A special word of thanks goes to my dear wife, Li Lin, whose consistent support, support in every sense of the word, has made this thesis and my pursuit of this master's degree and academic goals possible.
1

I. INTRODUCTION

Computer technology has been developed for no more than several decades, yet it is safe to say that it has already spread to every country on the earth. Less than a decade ago, the United States was the dominant market for technology, and today, though still a leading market, it is no longer a dominant one (Uren, et al. viii). This fact, true in most fields of technology, is revealed especially in computer technology, which represents one of the most, if not the most, advanced and modern fields of technology. Digital, one of the leading computer companies, for example, acquires 60% of its annual $12.9 billion sales revenue from countries other than the United States (Jones et al. ix). Apple computer, Inc. sells more than half the Macintosh computers in non-U.S. geographies each year (Apple Computer, Inc. 1992 5). The same is likely to be true, more or less, of IBM and many other computer companies.

Accompanying such a fact is the need to adapt computer products to the needs and cultures of the diversified international market. Computers are becoming daily operational devices of companies worldwide and a necessary part of the daily life of people of varying cultures. However, it is unlikely that a Japanese, or a Chinese, or a Russian user will have identical needs and ways of operation in using a computer with an American user, for whom the computer or the software product is originally designed. It is only logical then that computers and software products should cater to the diverse needs and cultural differences of their users or at least provide the flexibility to do so. Unlike in the past when foreign (non-U.S.) customers would settle for software products in English without even a translated version in their own language, today they are
demanding software products that best support their needs and cultural idiosyncrasies, for only such software products will yield the kind of efficiency for companies that use them to stay on the cutting edge in the increasingly competitive world market.

What such a trend dictates is the urgent and imperative need to internationalize and localize computer software products so that they are readily applicable in today’s global marketplace. Computer companies are already recognizing such a need and are dictating a great deal of effort in localizing their products. Most major software companies have special localization groups. Apple Computer, Inc., for example, has been localizing its products for a number of years, with localization groups in many major countries. Digital Equipment Corporation began its localization efforts over a decade ago and has now localization groups in America, Asia, and every country in Europe (Jones et al. xiii). These companies have been not only localizing their products but summing up their valuable empirical experience along the way, some of which is already in the form of published guidelines (e.g., Digital Equipment Corporation/Corporate User Publication; Microsoft Corporation; Microsoft International Product Group/Lingua Department; Apple Computer, Inc. 1988, 1990, &1992). Meanwhile, individual researchers have also been tackling localization issues (e.g., Jones, et al.; Uren et al.; Merrill & Shanoski; Hussey & Homnack; Blaschke; Spragins; Horton; etc.). Many have tried to come up with an applicable localization model; however, due to the complicated issues involved in localizing software products, the results have been, so to speak, short of being satisfactory, although efforts and attempts in this respect have been substantial.
In this thesis, I will review past research on localization and examine the current localization model of designing, writing, and translation. Limitations of this model will be discussed. I will then propose a broader perspective of localization, with a discussion of the important aspects often ignored by proponents of the conventional three-step designing-writing-translation model. Among them, human-computer interaction (HCI) design will be a major topic because computer design has to be understood in terms of how humans communicate (interact) with the computer and what factors, especially human factors, are involved in this communication (interaction). For computer design to fully incorporate all the complicated factors in human-computer communication and to yield truly user-friendly software products, HCI design is the key. Only when HCI design affords and allows the consideration of all user aspects can the software product acquire necessary qualities for internationalization and, further, allow for smooth localization because when HCI design is based on use practice, it leaves room for culture-specific considerations in software localization. In view of this regard, I will examine the theory base for HCI design: how information-processing psychology as the major theory that has been guiding HCI design for a period of time falls short as the ultimate driving force of this field because it does not take into account the dynamic aspects of HCI design and software localization. As an alternative, I will propose the theory of activity in Soviet psychology as a foundation for HCI design because activity theory will base HCI design on the actual use practice, thus enabling and providing ease for localizers to incorporate cultural considerations, which is likely to lead to successful software localization.
Following this discussion will be a case study of a localized Macintosh program, Chinese Language Kit, which, through an analysis of its design objective, rationale, and process, and an evaluation of its user-friendliness, will illustrate my proposed perspective.
II. PAST PRACTICE AND THE EXISTING LOCALIZATION MODEL OF DESIGNING, WRITING, AND TRANSLATION

In this chapter, I will examine the two localization models used so far: the translation model and the designing-writing-and-translation model. The application of these two localization models, especially the latter, by different companies will be discussed, and their inadequacies for successful software localization will be revealed.

1. Translation: A Misleading Concept

Though, as stated in my introduction, localization efforts began as early as over a decade ago, localization at that time was chiefly understood (or misunderstood) as mere technical translation. At a time when there were “not enough good translators” (Klein 28), especially technical translators, people’s attention was still focused only on translation issues such as manual translation, machine translation, and computer-aided translation, and few people were paying attention to important issues other than translation involved in localizing a product, especially software. It was such not because researchers deliberately ignored aspects of localization other than translation but because there was still short of an awareness of such aspects as cultural considerations, different use practices, and varying individual user needs and idiosyncrasies. It was natural then that whenever it came to the question of technology transfer, such as software localization, translation seemed to be the only logical issue that popped up in people’s minds. However, as is evident in the term itself,
“translation,” when used to refer to localization, has severe limitations in both its denotation and connotation.

A. What Is Translation?

Translation is “a rendering from one language into another” (Webster’s Ninth New Collegiate Dictionary). When applied in software production, it refers to the conversion of information, both on-line and document, from one language into another, and in the case of localizing American software, from English into a target language. The term may seem too simple to be even worth studying; however, it is the very sense of this self-evident term that is pointing to the limitations in its application to software localization. Translation at most is only part of the whole localization process. It refers to the actual act of rendering the documentation and the verbal part of on-line information into the target language, which is often the last of several stages involved in the localization process. The other stages include at least the designing and writing of on-line information and documentation. To be localized, a software product must have the flexibility to cater to the needs of its foreign users, and a consideration of such needs has to be incorporated in the early preparation, designing, and writing stages of software development. A localization process that begins as late as the translation stage will inevitably fail to incorporate such a consideration and therefore fail to provide full national language and cultural support for users of a specific foreign locale.
B. How Translation Is Misused in Place of Localization

Unfortunately, even as late as a few years ago, some people were still perceiving localization as a mere translation process. Gertrude Witter, for example, although recognizing the critical importance of “the strategic planning and designing of software technical information” (162), provides little clue as to how such strategic planning and designing should be incorporated into the localization process to make translation easier. Instead, Witter discusses the localization of a software product in terms of three types of translation, namely, machine displayable information (on-line information), documentation, and nomenclature.

This misconception of localization as translation is also reflected in people’s misuse of the term. Hussey and Homnack, for example, acknowledge the broad use of localization to refer to “the process of preparing and adapting software for foreign users” (RT-44); nevertheless, they “use the term ‘localization’ to describe the translation of software and documentation into the ‘local’ languages of foreign countries” (RT-44). They describe the localization process as containing four phases, namely, glossary development, software (screen message) translation, documentation translation, and the production of camera-ready documentation (RT-45-6). Although the authors claim that they are presenting the localization process from the perspective of the project manager, their definition of the process as one containing the above-mentioned four phases reveals their perception of the localization process as nothing more than translation as none of the four stages go before the translation stage, which reflects no involvement of translators in the designing stages, not to mention the planning stage, and of designers in the translation stage. It should be noted,
however, that Hussey and Homnack do give consideration to the need to involve software engineers and technical writers as well as professional translators in the localization process as it is an “interactive” process, although their description does not adequately support this notion.

Perhaps, the least successful form of localization is machine translation, which has been a much researched topic in the past and still sparks the interest of many people today, for machine translation has some obvious advantages: it is productive, can translate lower priority documents, can provide “emergency” translations, and can enforce terminological consistency (Grasmick & Elliott 102). However, even though a machine can be intelligently designed to translate a German sentence like “Das Gras frisst die Kuh” into “The cow eats the grass” rather than “The grass eats the cow” as the sentence was originally structured (Grasmick & Elliot 101), its limitations and undesirability are obvious especially when it comes to software localization because “word substitution alone will not yield a satisfactory product” (Spragins 28). Localizing a software product is more than a mere rendering from one language into the corresponding structure of another; it requires the thorough understanding on the part of the translator of the design rationale of the product, the inner workings of the system, and the different ways of perception by users of the target locale for localization, of which a machine, however intelligent, is obviously incapable. Translation even in its narrowest sense calls for an accurate interpretation of all the aspects of the product in the original and an anticipation and full consideration of the factors likely to be involved in using the product by its target users. Then in actually putting down the information in the target language, the translator has to create, upon such an understanding and interpretation, what linguistically and
culturally makes sense to the target users. In this sense, interpretation and creation seem to better capture the essence of translation.

2. Designing, Writing, and Translation: An Inadequate Model

A. What Is Localization?

Loosely speaking, localization refers to “the process of preparing and adapting software for foreign users” (Hussey & Homnack RT-44), or, in other words, “the process of modifying a product so that it is readily accepted in a different country, culture, or region of the world” (Apple Computer, Inc. 1992 xv). Uren, Howard, & Perinotti have put it in different terms with yet similar meanings: “Localization is the formal process that makes a program written for one language freely usable by people who speak a different language. This means adapting, translating, and adding international features to U.S.-based programs” (x). Jones et al. have defined localization in more specific and comprehensive terms:

Product localization is the process of adapting a product to suit the language, conventions, and requirements of a locale other than the one for which the product was originally developed. *Localization does not necessarily include translation.* (2; emphasis added)

An obvious notion embedded in these definitions is that localization means more than translation (and may not even include translation as Jones et al. have claimed). Such a notion brings us away from the old concept of localization as translation or translation as localization. As we can see, translation as the rendering of information from one language into another accounts chiefly “for linguistic differences between natural languages” (Jones et al. 3) and a limited
amount of cultural differences. The discrepancy in cultural considerations often requires the software localizer to adapt and add information for the needs of the target user, and translation alone is incapable of accommodating all the cultural differences. Therefore, aspects other than translation need to be incorporated into the localization process, namely, planning, designing, and writing, or to be more exact, every phase of the development of a software product.


Aware of the limitations of the translation model, researchers and practitioners in recent years have begun to take a more dynamic perspective toward translation and localization. Rosseel and Roll regard the translation process as “the communication act between the translator, the technical writer, and product developers starting at the preliminary phase and going on as a continuum until recomposition of the target language is complete” (RT-99). Though the authors here are not directly addressing the issue of software localization, we can see their new approach toward translation: they are extending the translation process back to the writing stage and further back to the design stage as well.

A similar notion is found in Fox and Swain’s discussion of “technical translation for multiple-language support,” in which they realize that “the essence of translating software comes down to the product’s design” and claim software design to be “an integral part of the translation process” (156). Therefore, according to Fox and Swain, the localization process begins with the product design, not with the technical rendering of information from one
language into another (156). Thus, a common problem with most software products today is that, often, when localization begins, the product to be localized has already been designed, produced, and in some cases, even shipped to the target locale. Localization in such cases can be no more than a compensation measure to ease the difficulty of the target users, to some degree at best but never completely.

Therefore, people are recognizing the need for localizers' early involvement in the development process of a software product. At Digital, planning is considered the first step of localization, at which localizers are required to

- Gain a general understanding of the localization process
- Gain an understanding of the roles [they] and others play in preparing and localizing user information
- Address the following localization requirements of the user information to be produced:
  — Availability
  — Short time to market
  — Flexibility
  — Consistency
- Include in all project documentation or training plans factors that influence the localization process. (Jones et al. 13)

Such a planning phase clearly precedes the translation phase. It should be noted, however, that this planning does not refer to the planning stage of the software development process. Nevertheless, to make such planning possible, it requires localizers' involvement in the planning and designing stages of software
Figure 1. Key Players in Digital's Software Localization (reprinted from Jones et al. 14)

development. If this is still inadequate in illustrating such an early involvement, a look at the key players in Digital's localization effort will give us a better clue (see Figure 1). These players can be divided into two groups: developers and localizers, who are coordinated by the translation coordinator and the translation team leader. Such a coordinating nature of the relationship between developers and translators (or localizers to be more exact) itself suggests that translators are involved in the early stages of software development and the localization process goes through the designing and writing phases before it comes to translation.

This need to start localization as early as possible is also reflected in the typical localization process of Apple computer software, which proceeds through the following phases:
• From the earliest stages of conceptualization and design, programmers and software designers identify later localization needs and take them into account. Nothing is done that will preclude easy and widespread localization of the final product.

• Early in the design process, the product’s target cultures are defined. Each target culture is an area of the world—a region, country, or culture—where a localized version of the software product might sell. If possible, the product is targeted for worldwide distribution.

• When a product has been designed and programmed for maximum ease of localization, and its target cultures defined, the actual process of localizing each version takes place.

• At all stages, each localized version of the product is tested and evaluated by users familiar with its target culture. (Apple Computer, Inc. 1992 7)

It can be seen that such a dividing of phases places quite some emphasis on the planning and designing stages. Another element also reflected in this process is the coordination and cooperation between the development and localization groups. In order to illustrate this, and also for the sake of my later discussion in the case study, I will go to lengths here to describe the detailed procedures of the technological process of Apple’s software development and localization. The example that follows is the process of creating electronic instructional products at Apple (see Table 1). Notice how the development process and the localization process go hand in hand.
Table 1. Apple's Procedures for Creating Electronic Instructional Products

<table>
<thead>
<tr>
<th>Instructional Products</th>
<th>Software Support (ISS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DESIGN</strong></td>
<td></td>
</tr>
<tr>
<td>Delphi—Weekly project report for instructional products. Includes information about major milestones and team members</td>
<td>ISS creates localization status report from Delphi project report. Includes information about current and upcoming projects (Countries receive localization status report).*</td>
</tr>
<tr>
<td>Suite Plan—Defines a plan for teaching customers how to use Apple products. Describes the product, identifies the target audience, outlines the learning materials (print and electronic) and learning path. Created by one or more instructional designers with input from writing managers, developmental editors, and product marketing managers.</td>
<td></td>
</tr>
<tr>
<td>Design Plan—Comprises two plans for disk. Instructional Design Plan describes product, target audience, goals and instructional approach, structure and interface components, content outline, and technical considerations. Production Plan describes tools and methods, resource plan, and rough schedule.</td>
<td></td>
</tr>
<tr>
<td>Design Plan Review</td>
<td>ISS receives final design plan from production supervisor and distributes design plan to countries. (Countries may review design plan and decide whether to localize the product. System software and on-line help must be localized.)</td>
</tr>
<tr>
<td>Script—Textual content and description of user interface complete. Can be script, storyboard, or rough prototype.</td>
<td>ISS receives final script from production supervisor and distributes script to countries on request. (Countries may review script.)</td>
</tr>
</tbody>
</table>

* Information in parentheses refers to localization actions at the country-specific level.
Table 1 (continued)

<table>
<thead>
<tr>
<th>DEVELOPMENT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rough Disk</strong>—Comprises edited text, prototype of all interface components, and complete rough graphics, sound, video, and animation.</td>
<td><strong>ISS receives final rough disk from production supervisor.</strong></td>
</tr>
<tr>
<td><strong>Rough Disk Review</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Pre-Alpha Disk</strong>—Produced in final delivery tool. comprises final integration of content and interface; final text, graphics, sound, video, and animation; and complete interface design.</td>
<td><strong>ISS receives final pre-alpha disk from production supervisor.</strong></td>
</tr>
<tr>
<td><strong>Pre-Alpha Review</strong></td>
<td></td>
</tr>
</tbody>
</table>

**PRODUCTION**

| Alpha Disk—Software Configuration Management (SCM) builds disk. Macintosh CPU Software Quality (MCSQ) tests disk and creates bug report. IP fixes bugs. Process is repeated until disk meets alpha requirements. MCSQ accepts disk as alpha when preliminary Engineering Requirement Specification (ERS) defines critical features of software; interface is reviewed by human Interface team member; software modules are integrated, preliminary test is complete; and all features are functional. | **ISS receives build instructions for alpha disk and produces localization kit for disk, including disk configuration information, instructions for translating files, special development tools, testing guidelines, and text to be translated. (Countries receive localization kits for disk. Each country forecasts the likelihood of translating the disk.)** |
| Beta Disk—SCM builds disk. MCSQ tests disk and creates bug report. IP fixes bugs. Process is repeated until disk meets beta requirements. MCSQ accepts disk as beta when  
  - ERS is updated  
  - User interface is frozen  
  - All critical features are functional. | **ISS receives build instructions for beta disk. (Countries receive localization kit for disk and begin translating stable components, such as frozen text or graphics.)** |
| Final Disk—SCM builds disk. MCSQ tests disk and creates bug report. IP fixes bugs. Process is repeated until disk meets final requirements. MCSQ accepts disk as final when final ERS is complete and all features are functional. | **ISS receives build instructions for final disk and change history for final candidate, writes international test plan for disk, and updates localization kit for disk.** |
| Golden Master Candidate—SCM builds disk. MCSQ tests disk and creates bug report. IP fixes bugs. Process is repeated until disk meets golden master requirements. MCSQ accepts disk as golden master when full testing is complete. | **ISS receives build instructions and change history for golden master candidate and updates localization kit for disk.** |
Table 1 (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golden Master</td>
<td>SCM builds golden master disk.</td>
</tr>
<tr>
<td>Sign-off</td>
<td>Golden master release binder for disk is signed off by team members (engineers, animator, manager, production, supervisor, tester, SCM). Release binder contains original code and build instructions. Review Change order Board (RCOB) confirms part number, signs off, archives copy of release binder, and delivers disk to manufacturing.</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Disk is duplicated, labeled, and packed out with product. Copies are sent to countries that will ship U.S. versions of disk.</td>
</tr>
</tbody>
</table>

ISS receives golden master disk image from SCM and completes localization kit for disk. (Countries receive final localization kit, begins translation, and send translated files back to ISS. International SCM builds testing disk. Countries test disk. Countries and ISS fix bugs. Process is repeated until disk meets golden master requirements.)

(International SCM builds golden master disk.)

(Golden master release binder for disk is signed off by team members: localizer, ISS engineer, SCM, tester, manager. Release binder contains any new or modified code and build instructions.)

Disk is duplicated, labeled, and packed out with product.

(Adapted from "Creating Electronic Instructional Products" by the Instructional Products Department, Apple Computer, Inc.)

Two processes are described here: one is the software development process represented by the procedures at the Instructional Products level and the other the localization process represented by the procedures at the International Software Support level and the Country-Specific level. What this figure tells us is that the localization process begins at the very first stage—Delphi—of the software development process. The two processes develop simultaneously side by side throughout the design, development, and production phases. The localizers are kept informed of every step of the way of software development. At most stages, actions on the development side are followed or matched up by corresponding actions on the localization side. What this suggests is that some form of communication is going on between developers and localizers, although
judging from the figure, this communication seems largely unidirectional, i.e., from developers to localizers but not vice versa. This communication is a sign of an awareness of the need for localizers to gain an understanding of the design rationale on the one hand and for designers to possibly take into account cultural considerations of foreign users such as their values, learning styles, beliefs, habits, etc. (Spencer and Yates RT-149).

If we examine these studies and practices, whether by Rosseel and Roll or by Fox and Swain, whether by Digital or Apple, or by other researchers and practitioners (e.g., Spencer and Yates, Merill and Shanoski, etc.) and other companies (e.g., Microsoft, IBM, etc.), we find the same trend, i.e., the localization process extends much further back than the translation stage, often to the very beginning of the development process. Such a trend has resulted in a localization model that is prevailing in the software development industry today—the three-step localization model of designing, writing, and translation (the term "designing" here is used to include both the planning and designing phases as it is used by many people today).

This three-step model of designing, writing, and translation represents a positive progress from the old localization-as-translation model because, as the localization process takes an early-bird start at the designing stage of software development, it is effecting a communication act between localizers and software designers, through which cultural considerations can be incorporated into the design to ease for later translation while the design rationale can be explicated to guide localizers. Such a communication act is activated through some kind of coordination between developers and translators by the translation coordinator.
and the translation team leader in Digital's case and through some kind of information transmittal from designers to localizers in Apple's case.

No doubt, this new concept of localization deserves acclaim as it points to a new, encouraging phenomenon—people are updating their viewers in their perception of localization issues and endeavoring to perfect their localization practice to keep up with the rapidly changing situations of the global marketplace and make their software products better address the needs and requirements of their foreign customers. This represents a positive effort on the part of companies to integrate translators, though to different degrees, into the production process, thus enabling them to be business communicators rather than isolated translators (Rosseel and Roll RT-99). However, although it represents a break from the old notion of localization as translation, this three-step localization model of designing, writing and translation still leaves much to be desired, at least in two aspects.

It is true that this model has extended localization from a single-stage process into a three-stage one; it is also true that developers have recognized the need to incorporate translation considerations into product design from its very conception (Merrill and Shanoski); it is even true that companies are making special efforts to internationalize their software before localizing it (e.g., Apple and IBM); however, a closer look at their actual localization procedures suggests that most people in actual practice are looking at the three-step model of designing, writing, and translation as a linear, sequential process. Though to a certain extent this process is diachronic, to a larger extent, it is synchronic, synchronic in the sense that these three stages interpermeate and intertwine with one another. For example, when product design incorporates translation
considerations and acts toward translation, it is already performing part of the translation job. On the other hand, in successfully translating (or rather recreating) information for users of a foreign locale, the translator (or rather the localizer) takes on the task of interpreting and adding a new meaning to the original design rationale, in which manner he is performing part of the design act. This relationship is more clearly reflected in that between the writing and translation stages. As long as we acknowledge that translation is more than mere word substitution, we can see that translation is not readily separable from writing because translation itself means recreation, and thus writing. Therefore, successful localization means that whenever the localizer performs at one of the three stages, he is performing part of the actions at the other two stages as well because localization is basically an integrated process.

At this point, one may argue that such a three-step localization model does seem workable as long as we give due consideration to all the aspects at each phase. Such an argument is exactly right in that all aspects involved in each phase should be given due consideration, but this is also exactly where problem arises. When we look at the localization process as a linear, sequential one, putting distinct dividing lines between the three phases and thus inevitably denying their integrative nature, we are bound to fail to give due consideration to all the aspects involved. As a result, in the actual localization practice, the translation phase is being overemphasized, with designing and writing being assigned a subordinating, less important role.

Another problem with this three-step localization model of designing, writing, and translation is that it has given rise to two parallel processes in the actual practice of software production, namely, the processes of development and
localization. At first sight, the coexistence of these two processes seems natural and even necessary. However, what is problematic is that the localization process is put in a position secondary to the development process, rendering localizers in an inferior, passive, and subordinating position to developers. This results in the localization process being nondynamic and noninteractive.

Take Digital’s localization process for example. It puts much responsibility on the localizers (Jones et al. 13; also see discussion at the beginning of this section). Although such responsibility is reasonable and justifiable, we see no responsibility, however, on developers, who definitely need to gain an understanding of localization considerations just as localizers need to understand the design rationale. As we can see in Figure 1, translators (localizers) are put in a separate group from developers. Thus, communication between the two can only be carried out through coordination by one or two persons. Though this coordination appears to be bridging the gap between developers and localizers and indeed it does to some extent, at the same time it is isolating the two from each other and creating a gap, rendering them unable to interact directly with each other and thus reducing communication to a nondynamic and less interactive form. A likely result of such cooperation will be a localized software product the design of which is short of full consideration of all the possible factors involved in its actual use in the foreign locale: individual, corporate, and cultural.

The same problem exists with Apple’s localization process, though perhaps to a lesser degree. Localizers are informed of the progress of development at almost every stage and have the chance to review the plan or the outcome at each stage (see Table 1). However, as stated earlier, the
communication between the developers (the Instructional Product group) and localizers (the International Software Support group and the country-specific localization group) appear to be mainly unidirectional, with localizers in a subordinating position, adapting and adjusting their localization actions to the progress of the development group at each stage. Though we can assume that localizers provide feedback to developers at each review, the localization group is not shown to have active participation in the development process. Rather, the localizers are shown to be in a passive position (partly reflected in the repeated use of the work “receive” in Table 1). Thus, two distinctive processes are revealed in the procedures, with the localization process dependent on the development process but not vice versa. I’m not arguing here that the development process should be dependent or fully based on localization, but I do argue that in the production of a software product, all parties involved—designers, engineers, writers, localizers, etc.—should be active participants in the development process. Only in this way can localization considerations be fully incorporated into the design (this point will be elaborated in the next section). This is especially important at a time when software products localized from English make up about 75% of all the software used in the world (Fox and Swain 156) and when many companies such as Apple are deriving over half of their revenues from localized products (Uren et al. 106).
III. MY THEORETICAL MODEL OF LOCALIZATION

In this chapter, I will first of all specify the factors involved in localization. I will then argue how software localization should be viewed as a dynamic, interactive process that involves various aspects and expertise in many different fields. A good portion of this chapter is devoted to the discussion of human-computer interaction design as it is the focal point of computer design and is thus crucial to and lays the foundation for localization. One reason for laying such an emphasis on HCI design is that the localization process begins as early as the very conception of the product and thus includes as an essential part of the process the design of HCI. Another reason is that how successful HCI design is determines how successful localization will be because HCI design is the stage where designers incorporate or provide the flexibility to incorporate culture-specific considerations of users of the target locale for the localization product. I will thus examine what researchers are doing with regard to establishing a theory for HCI design.

1. A Broader Perspective—Aspects of Localization

As stated in the last chapter, the current localization model of designing, writing, and translation regards the localization process as a linear, three-step sequential process, thus denying the intertwining, interpermeating, and interactive nature of the three phases, resulting in an overemphasis on the translation phase and the emergence of two distinctive processes of development and localization, with localizers being in a subordinating position to developers. Such a model basically defines localization as a non-integrative and
noninteractive process, inevitably stopping short of an adequate consideration of all the factors involved.

What exactly then are involved in localization?

Broadly speaking, there are two major fields involved in localization: computer sciences and human sciences. Everything involved in localization falls into either of the two categories. Technical design, including systems functions, languages, graphics, etc., goes without saying, but it is not the first and the only step. Strategic planning precedes technical design and is critical to the whole localization process. It is especially at this stage that all the factors should be duly and fully considered. Since the product to be localized is produced for foreign users, their customs, values, beliefs, working conditions, and cultural psychology should be taken into account, not only in the planning stage, but in designing, writing, and translation stages as well. Then, there are also technical documentation and translation. Even marketing consideration should be incorporated sometimes.

Wulff, Evenson, and Rheinfrank have provided a fairly comprehensive list of the areas of expertise required on designers' side in developing a software product:

- Systems design
- Graphic design
- Product design
- Psychology
- Human factors
- Linguistics
- Communication
Moreover, they also include users on their development team. This unique idea highlights the importance of designing software that addresses user needs. Though Wulff, Evenson, and Rheinfrank are not specifically addressing the issue of localization, their ideas are certainly applicable.

Correspondingly, to produce a localized software product, we would need system designers, graphic artists, software engineers, cultural psychologists, professional technical writers, professional technical translators, communication experts, marketing personnel, and so on. An ideally successful localized software product would be designed by someone who has the expertise of all the above people; however, such a person is impossible to find. Therefore, to develop and localize a software product, it requires that all these people with different expertise in different fields actively participate in the process and communicate and interact with one another in a dynamic way. Therefore, in view of these considerations, I define localization as a dynamic, integrative, interactive, and communicative process of developing (not just adapting) software for users of a foreign locale which involves knowledge of both computer and human sciences.

2. Human-Computer Interaction Design: The Key to Successful Localization

When we speak of the knowledge of human sciences required in software design and localization, we are talking about human-computer interaction (HCI) design. The extent to which software design can successfully accommodate the ways how humans communicate and interact with computers decides how successful the design and product will be, in other words, how user friendly the
system or a program will be. Likewise, how well the system design can accommodate various kinds of circumstances and situations in its use by its foreign users with regard to their unique customs, habits, beliefs, values, etc. is the key to successful localization. Unfortunately, people have not always agreed on the importance of HCI design in software development, and especially in localization, and when they do agree, they have not been unanimous on their views about methods in HCI design. However, before we discuss the current state of HCI research and its impact on localization, let me first of all define HCI and some of its key concepts.

**A. What Is HCI?**

Loosely defined, human-computer interaction (HCI) refers to the way humans and the computer act upon each other and meet each other’s capabilities. This field has been variously termed as human factors, user interface, man-machine communication and variations of these (Jones 6). Jones considers human-computer interaction as “the most broadly encompassing term” and defines it in terms of interactive systems:

Guedi (1980, p109) defines an interactive system as one in which “man and machine play a symmetrical role, in the sense that they exchange initiative, they share responsibility for system control and task performance, and they both can show complex behavior.” Human-computer interaction is thus used to refer to all the factors which influence and affect design of interactive systems.

(7)

Carroll regards human-computer interaction as “an interdisciplinary area of applied research and design practice [whose] key concern is to understand and
facilitate the creation of ‘user interfaces,’ that is, of computers as experienced and manipulated by human users” (1). He further specifies that HCI draws on knowledge from various areas such as psychology, computer science, anthropology, management science, industrial design, etc. (1). More accurately speaking, Carroll’s is a definition of human-computer interaction design, but it does give us a more specific idea of what this field is about, especially from the designer’s point of view. Moreover, an important observation in Carroll’s definition is that of the interdisciplinary nature of HCI, which is considered “probably the most salient feature” of HCI by Brusilovsky, Burmistrov, and Kaptelinin (11).

A key term in HCI design is “user interface,” which is sometimes used as a substitute term for human-computer interaction. User interface is the essence of HCI because most considerations in HCI design come down to the design of user interfaces. To understand HCI design, you must first of all understand user interface. For the sake of my later discussion, therefore, I will explain here what interfaces are, according to how these terms are scientifically and technically defined.

According to Webster’s Ninth New collegiate Dictionary, an interface is

a. the place at which independent systems meet and act on or communicate with each other (man-machine interface);

b. the means by which interaction or communication is effected at an interface.

Accurate as it can be, this technical definition may still be too abstract for some people to conceptualize the term “interface.” In an effort to enable people to visualize the term, Sayeki has provided us with an excellent illustration with the
example of a blind man with a cane. The handle of the cane where the blind man grips and thus comes into contact with the tool—the cane—is one interface, the interface to the tool. The point of the cane that comes in contact with the world is another interface, the interface to the world (Gradin 272). The difference between the two will be discussed later in this section.

Norman explains interface in terms of its function in relation to the cognitive artifact, "an artificial device designed to maintain, display, or operate upon information in order to serve a representational function":

We can conceptualize the artifact and its interface in this way. A person is a system with an active, internal representation. For an artifact to be usable, the surface representation must correspond to something that is interpretable by the person, and the operations required to modify the information with the artifact must be performable by the user. The interface serves to transform the properties of the artifact's representational system to those that match the properties of the person. (26)

Jones, on the other hand, defines interface more specifically as

the boundary between one part of a computing system and another, or between the computer system and the human user. The specification of how and in what format information is transferred across the boundary comprises the interface specification. (118)

What these definitions suggest is that there are several kinds of interfaces, the terms of which are often misused interchangeably. In an effort to eliminate confusion and misuse, Grudin distinguishes between "the user interface" to a computer, "the computer interface" to a user, "the user's interface" to a computer, and "the computer's interface" to a user (271). According to Grudin,
"the user's interface" to a computer refers to all the artifacts, processes, and people involved in using a computer, including

- the software-controlled dialogue, which is the major part
- any documentation and training that are part of using the computer
- colleagues, consultants, system administrators, customer support, field service representatives. (See Figure 2)

Figure 2. A User's Interface to a Computer (reprinted from Grudin 272)
Grudin explains that this figure illustrates a user’s interface to a computer. The user consults documentation, is trained, and solicits advice from colleagues, system administrators, and others. The system administrator may modify the system on behalf of the user. Following a hardware failure (a less frequent part of the user computer interaction as hardware reliability improves), a field service engineer may directly modify the system. These activities shape the nature of the interaction that takes place through the input and output devices. (272)

On the other hand, “the computer’s interface” to a user, from an engineering perspective, is traditionally defined as “being the software controlling the dialogue” (Grudin 272). This comes down to mainly the input/output devices (see Figure 3).

Thus, Grudin states, “‘user interface’ denotes the computer’s interface to the user, not the user’s interface to a computer” (272). Accordingly then, we can assume that “computer interface” refers to the user’s interface to a computer though this is not expressly stated in Grudin’s discussion. Such a conception of user interface has been dominating in the field of computer design. It has led to a neglect of the user aspects as we can see that no human factors are included in this perspective, thus excluding any aspect of the computer interface. Let’s go back for a brief moment to Sayeki’s example of the blind man and the cane. The blind man’s interface to the tool, the cane, represents the interface between the human and the computer while the cane’s interface to the world stands for the interface between the human working with a computer and the world. Through the blind man’s execution, the cane as a plain, simple tool connects and
combines the two interfaces, effecting an interaction between the human and the world.

Unfortunately, the computer is no plain, simple tool. Its interfaces involve many complex aspects and factors, both technical and ergonomical. The user interface from the engineering perspective discussed above is much too narrow to accommodate all these aspects and factors. A consideration of all the interfaces between the human working with a computer and the world thus calls
for a broader perspective of user interface. Grudin envisions a merge of the computer interface and the user interface in the future as the computer interface expands to include on-line documentation, on-line help, and on-line training (some of which have already been realized), thus reducing the need for mediators (Grudin 273; see Figure 4).

Figure 4. A User’s Interface to a Hypothetical Future Computer (reprinted from Grudin 273)
B. Understanding Human-Computer Communication as an Interactive Process

The design of human-sciences-related aspects in software design and localization requires that the designer take a proper perspective toward human-computer communication because perspectives structure cognitive processes and are indicative of how designers look at the different aspects of HCI, the importance of user interests, and the nature of human beings (Maass and Oberquelle 234-5).

It follows then that the design process will reveal about the designer’s perspective and priorities: the degree of user involvement, “designers’ openness for other people’s ideas, their willingness to admit crucial gaps in their own knowledge or education,” etc. (Maass and Oberquelle 235).

The reason I am taking a moment here to review some designer perspectives is that different designer perspectives reflect designers’ different perceptions of human factors in HCI design, which are crucial to the incorporation of user perspectives. Since user perspectives reflect considerations of specific cultures, designer perspectives are thus directly related to the incorporation of culture-specific concerns in localizing the software product. However, not all designer perspectives take into account the social nature of human-computer interaction and thus the human factors involved in it. It is therefore necessary to review these perspectives and see how they affect HCI design and, ultimately, localization.

Maass and Oberquelle have specified five perspectives in system design and use. The first is the “machine perspective,” which regards the computer as a complex machine and humans as servants to the machine, with the ultimate
goal being “to substitute human labor as far as possible” (237). The second is the “systems perspective,” which sees humans and the computer as having basically the same properties and thus HCI as data transmission only. This perspective emphasizes the efficiency of data transmission and basically ignores the ergonomical aspects of HCI. The third is the “communication perspective,” which looks at humans and computers as agents engaged in an interaction similar to a human-to-human dialog. This perspective has two derivatives, the “partner perspective” and the “formal communication perspective.” The “partner perspective” aims at systems designed to act as humanly as possible, often resulting in overestimating computer intelligence. The “formal communication perspective,” however, acknowledges the crucial differences between human-computer communication and human-to-human communication and sees the computer and the user as interacting via some kind of shared language. This approach does have a problem, though: the interactive system is often designed by several people, but the user often experiences it as a whole and will therefore encounter inconsistent and unpredictable behavior from the system. The fourth is the “workshop perspective,” which regards the computer as an embodiment of a workshop-like work environment providing tools, materials, and space. According to this perspective, the individual is viewed as “similar to a craftsman who has to make a complete product according to the rules of the craft” (240). The last is the “media perspective,” from which the computer is seen basically as a human aid, a means to support human communication.

The way these perspectives are presented here represents the historical sequence of their emergence. Each perspective reveals designers’ implicit
assumptions about humans and human work and about a particular human-computer relationship. As is reflected in some of these perspectives, the dominating view of humans among designers has been pessimistic as designers view humans as passive, irresponsible, and non-creative (Maass and Oberquelle 248). As a result, their design rationales are characterized by an attempt to control user behavior and to develop prescriptive software. However, Maass and Oberquelle propose that designers take the role of “a real, cooperative partner for users” instead of “a generous god” or “invisible friend,” adopt “humanistic,” “work-oriented” design approaches, and look at humans as active, creative participants in interaction (248-9). In view of this regard, the “machine perspective” and the “system perspective” are obsolete because they “neglect or underestimate users’ needs and capabilities” (Maass and Oberquelle 250). In contrast, Maass and Oberquelle argue, the “formal communication perspective,” “the workshop perspective” and the “media perspective” are acceptable because they are based on “a view of computers as complementary and supportive to humans” (250).

The evolution of these perspectives, I think, has suggested a shift of focus from the computer to the humans in people’s perception of human-computer communication. Instead of viewing the computer as dominating over the user, people are now looking at the computer as complementary and supporting to humans. Though such a change of views may seem to be going a little bit too far and may result in a tendency to underestimate the role of the computer in human-computer communication, it does, however, afford ample consideration of the ergonomical aspects of HCI, which in turn will lead to the ample consideration of culture-specific factors of different use practices in the
localization of the product. What is important and seems appropriate to do is to look at human-computer communication as an interactive process, of which the complexity of human working conditions, be it of any culture or any use practice, as well as the complexity of the software systems should be given adequate considerations.

C. Research on HCI Design: Much Ado about Nothing?

Accompanying the change in people's views about human-computer communication is an increasing research interest in HCI. As HCI design has become a critical component in software development and exerts a direct influence on localization (and determines to a considerable extent the degree of success in localizing a software product), it is worthwhile to review how much has been done in research in this field and how it has affected software design and, consequently, localization.

Thus, in this section, I will review current literature on HCI design, discuss the inadequacies of information-processing psychology as the guiding theory for the field, and summarize three directions in which current research in HCI design is headed for in search for the best alternative that will lead to an appropriate guiding theory for the field. With the discussion in this section, I am also hoping to provide my readers with a rationale for the theme of my next chapter, why human activity theory is a better alternative than information-processing psychology in guiding HCI design and, further, software localization.
a. History

There is a prevailing view among researchers that HCI research has not come to much so far and has exerted little impact on HCI design. To understand such a view, it is necessary and worthwhile for us to review the brief history of HCI research so as to understand that it has gone through a difficult path and that HCI research is still at an exploration stage.

The first computers were designed for and used by computer specialists or people specially trained in using computers. At a time when computers were big, complex, and expensive machines and small personal computers that could be used by non-specialists were not even in prospect, human factors were not an issue. Later, even after considerable decreases in the size and cost, accompanied by enormous increases in the numbers of computers and users and in the ease of computer use, such a view of computer users as trained specialists capable of coping with the complexities of computers still prevailed for quite a period of time (Jones 8).

The first ergonomical (human factors) concerns with computer design and use probably started in the late 40's to the early 50's. When Mauchly in the late 40's and Shackle, Turoff, and others in the late 50's voiced their concern for human factors problem of non-specialist users, their voices were much too weak to capture the attention of the majority. Such a situation continued into the 60's even though there was an increasing interest in ergonomical issues. Serious research interest in this field did not emerge until the late 60's when in 1969 Ergonomics ran a special issue on human factors issues in computer use, the papers of which were reprinted in IEEE Transactions on Man-Machine Systems. The birth of the International Journal of Man-Machine Studies in the same year
also marked the beginning of an era of research studies in computer ergonomics (Gaines 3).

Publications in the early 70's by, for example, Sackman; Weinberg; Sime, Green, and Guest; Martin; Wasserman; and so on dealt chiefly with the psychological aspects in computer design and use. This is no surprising phenomenon as the mid 70's was witnessing the beginning of an era of small-size, low-cost personal computers. It was around this same period that we saw a boom in the interest and literature in psychological issues in computer use and the emergence of such buzz words as "HCI" and "user friendliness" (Gaines 4). Coming into the 80's, especially the late 80's and the early 90's, we have witnessed a flourish in research interest and literature in HCI, so that today, perhaps no one would deny the importance of human factors considerations in computer design and use. How much this research in HCI has come to in terms of theories and to what extent it is directing software design, however, still remain a question.

b. The Status Quo

Coming into recent years, HCI has developed to the stage where its major concern is to explore the psychological factors involved in the creation of user interfaces and where researchers are endeavoring to come up with an applicable theory to guide, if not direct, HCI design. Among the literature in this field, Carroll's *Designing Interaction: Psychology at the Human-Computer Interface* has been a representative and influential work. This book pools the ideas and research results of nineteen prominent researchers in the field of HCI from different parts of the world. These scientists, like many others, are devoted to the
plain, yet no simple, objective of establishing a theoretical base, of which HCI design is still in lack.

Due to this lack of an applicable theory to guide HCI design, there has been a corresponding lack of systematic guidelines for software localization. As long as no working theory is developed for HCI design, it is impossible to work out a guiding theory for localization. To develop a theory for localization before we do one for HCI design is to put the cart before the horse because HCI design is an essential step that precedes localization and because unsystematic treatment of HCI design will resist and hinder any attempt at a systematic approach to localization, which has HCI design as its preceding step. Therefore, before we attempt to develop a systematic approach to software localization, we need to sort out the complexities in the research in HCI design.

A fundamental question asked about HCI is “What role basic science plays in the application” (Carroll 1). In the case of HCI, the basic science refers to the modern information-processing psychology. However, “past research in academic psychology has had no impact on HCI design practice while some of the most seminal and momentous user interface design work of the past 25 years made no explicit use of psychology at all” (Carroll 1). The early approach in the applied psychology of HCI was direct empirical contrast, but such an approach is not likely to bring us too far forward. Therefore, it remains the task of scientists to provide scientific explanations for the past and present HCI design practice and theoretical guidance for future progress. This is as yet a difficult task, but scientists are coming to agreement on many issues, one of which being that for science to be applicable, it must achieve both specificity and applicability (Carroll 2), which is true not only of HCI but also of all other fields.
The present issue in HCI psychology research is this: shall we go beyond information-processing psychology to social and behavioral sciences or shall we endeavor to enrich the information-processing psychology? In an attempt to find a solution to this issue, several alternative directions have been sought. I will discuss here the following three alternative directions in the research of HCI psychology:

1) To make better use of information-processing psychology
2) To extend the scope of information-processing psychology
3) To broaden the range of psychology we seek to apply.

The first alternative direction for HCI research is to make better use of information-processing psychology. To use a more simple and familiar term, information-processing psychology means cognition, or cognitive psychology. To view HCI psychology in terms of cognition has been a prevailing trend as “the dominant view of human nature portrayed in psychology and allied disciplines...has been a cognitivist...one” over the past 30 years (Bannon and Bodker 228). Such a view tends to treat intelligent agents as information-processing systems, which logically leads to the hypothesis that intelligent systems can be described in computational or information-processing terms. It follows then that human and artificial reasoning processes must bear some essential similarity. As a result, researchers who favor the cognitivist view and information-processing psychology have tried “to build computer models of human thought processes that are taken to be strongly equivalent to the actual processes that are used by people in their comprehension and understanding of the real world” (Bannon and Bodker 229).
The influence of such an approach, of course, can be felt even today. Barnard, for example, in searching for ways to bridge the gap between basic theories and the artifacts of HCI, thinks that science provides a representation of the real world. Therefore, a science of HCI will serve as an intermediary representation of the reality, a “discovery” representation. The reason, he argues, is that in both applied and basic research, we construct our science not from the real world itself but via a bridging representation whose purpose is to support and elaborate the process of scientific discovery” (105). And such a representation and the construction of it can only be realized by means of human cognition to mediate between science and the real world of computers, software, manuals, tasks, users, environment, etc.

DiSessa’s mental models as principles for programming comprehensibility also support the cognitivist view. She specifies three types of mental models: structural, functional, and distributed. A structural model dictates “a context-invariant specification of the set of possible configurations of the computer” (174); a functional model refers to the contextualized description of programming tasks and strategies; and a distributed model is more of an intuitive type that accommodates fragmentary, spontaneous descriptions to address learners’ specific concerns on the spot. Her view bears a clear cognitivist mark as she claims that “design can serve not only as an application of emerging cognitive principles but, more radically, that design can serve as a central part of the scientific work that generates those principles” (162).

Lewis’s classification of psychological theories into outer and inner theories is also rooted in cognitive psychology. An outer theory is one that characterizes psychological phenomenon, i.e., what mental processes do and how
they are influenced by external factors, without explaining how these processes are formed. An inner theory, on the other hand, is one that describes these processes, i.e., the mechanisms underlying processes. According to Lewis, what is needed for HCI research is inner theories because "mental processes are fundamental to computer use," and understanding how they work will lead to new discoveries of ways to exploit mental processes (155). He believes that learning about the implementation of mental processes might improve user interface design in a unique way (154).

However, as Bannon and Bodker have pointed out, the model human information processor developed more or less as a result of such mental process models or prototypes "that lies at the heart of the cognitive science tradition" is bound to be very different from the natural human information processor because the former is short of the consideration of the rich context of human interaction (229). Because of this, some researchers are already exploring into other possible theories, which brings us to my next point, the second alternative direction for HCI research.

The second alternative direction for research in the psychology of HCI is to extend the scope of information-processing psychology. Norman, Payne, and Carroll et al., for example, look for factors outside humans that affect human action. More or less, they have argued that "the structures of the external world—including the structures designed by humans—are critical determinants of activity and experience" (Carroll 8).

In describing "cognitive artifacts," devices that enhance human thought, Norman observes that traditional psychology has aimed at the understanding of the unaided mind and ignored the information-processing role of artifacts
although they are affecting human cognition in a tremendous way. He proposes, as an important task for psychology, developing a typology of artifacts and an understanding of some of the advantages and disadvantages in using artifacts. He distinguishes between the system view and the personal view of artifacts. When a person uses an artifact to accomplish a task, the person and the outside observers see two different views. From the system view, the outside observer sees the total structure of person plus artifact in accomplishing the task whereas from the personal view the person himself sees how the artifact has affected the task to be performed. From the system view, the artifact is seen as expanding some functional capacity of the task performer while from the personal view, the artifact has changed the nature of the task, actually, replacing one with another, thus possibly inflicting radically different cognitive requirements and requiring radically different cognitive capacities from what the original task requires.

Payne has adopted a similar view in specifying three aspects of artifact mediation between the task and the performer. He argues that artifacts restructure tasks for people by providing a representation of the task domain on which the user operates, an artificial language mapping operators onto actions, and “a set of responses whose content and timing guide the interpretation of user interactions” (Carroll 8). Carroll et al. have expressed the same point that “the study of artifacts is an important area of concern for psychology in arguing that artifacts in use can be understood as implicitly embodying specific claims about their users, claims about what would have to be the case in order for the artifact to be usable” (Carroll 8-9).

A third alternative direction for HCI research has been to broaden the range of psychology we seek to apply. Barnard, though acknowledging the
impact of cognitive science on design by means of direct application of empirical methodologies and data and of psychological reasoning in the creation of design concepts, recognizes in the meantime the inadequacy of information-processing psychology in addressing extended sequences of behavior and the information-rich environments or circumstances in which people have broader task goals.

DiSessa, Greif, and Bannon and Bodker propose incorporating concepts and methods from developmental approaches to psychology and are concerned with how environment impacts the human mind and their activities. For the evaluation of software usability, diSessa advocates a focus on skill development in a prolonged period rather than only the early stages, which may not be representative of the full course of skill development.

Going further in broadening the range of psychology to be applied in HCI, Greif introduces the German “action theory” and work psychology. This work psychological action theory, sometimes referred to as action regulation theory, has as its central goal of work design the promotion of human growth by enlarging the scope of action for people. The core of work activity design is the task and artifact design; the basic methods of designers are instruments of work and task analysis; and the basic criteria for task and artifact design are “the enlargement of control, complexity, and completeness (or meaningfulness) of the task supporting human growth and personality development” (Greif 209). Bannon and Bodker, on the other hand, explores the relevance of the Soviet “activity theory” to the psychology of HCI from a cultural historical perspective. Activity theory, often cited as the general philosophical background for action theory, emphasizes the analysis of persons acting in context and a higher level of description of extended sequences of action and the long-term
motive behind the action. I will come back to activity theory for a more detailed discussion later in this paper.

Other researchers, Brooks, Pylyshyn, and Landauer, also advocate a broad-based "descriptive" psychology of HCI. Brooks introduces comparative task analysis as an alternative direction for HCI. He proposes a taxonomy of tasks based on the methods required to solve problems and favors description systems that are hierarchical, that are operational on new tasks, and that can assimilate and differentiate tasks in regard to their interface requirements. Both Pylyshyn and Landauer recognize the high complexity of HCI and are far from optimistic about establishing a theoretical base of HCI, especially on cognitive psychology.

In searching for an appropriate theory base for HCI, the above-mentioned researchers, together with others like Henderson, Karat and Bennett, Tetslaff and Mack, and many others I believe, have been endeavoring to make what is currently available in science applicable to the design practice, though without much success. Researchers are rather conservative about the role of current cognitive science in HCI, seeing what appears at present to be an unbridgeable gap between basic science and social practice. Reiterating Card, Moran, and Newell's conclusion, Carroll claims that "the most sustained, focused, and sophisticated attempts to develop explicit extensions of academic psychology for HCI have had no impact on design practice" (1). Pylyshyn also acknowledges that scientific theories, especially in social sciences, have rarely been responsible for practical improvements, which have been the result largely of general wisdom, occasional innovative ideas, and new techniques and methods, but seldom of deduction from scientific principles. One of the reasons that have contributed to such a fact, according to Pylyshyn, is that scientists are generally problem focused
whereas engineers are often solution focused. It seems to have been a tradition that science plays a minor role in the development of technology (Carroll et al.), which makes it easy for us to understand why Carroll et al. have argued against the conventional view of the deductive application of science in the invention, design, and development of practical artifacts. Barnard describes the effects of current theories on HCI design as at best marginal because of their restricted scope and their failure to accommodate "the real context of work" and the "fine details of implementations and interactions that may crucially influence the use of a system" (106). Lewis acknowledges the valuable methodology and a useful empirical orientation that psychology has offered to design but comments that it contributes "little more."

I could go on with quotations from many other researchers. A consistent voice among researchers is obviously that we are still far away from being able to establish an applicable science base for HCI design and even if we could do so, its impact on HCI design would still be questionable. The inadequacy of current theory-based research is obvious. What theories then are needed and can be considered adequate for HCI? A useful theory, according to Landauer, is one "that is good enough to dictate system design characteristics that support much better human-computer interactive performance than are now produced by art and emulation alone" (60). Due to the chaotic, highly complex, and unpredictable nature of the behavior of human-computer systems, a useful theory seems impossible at this time, and "the theory of human cognition is now and may forever be too weak to be the main engine driving HCI" (Landauer 61).

In view of these considerations, therefore, there seems to be only one alternative left before us to solve the problem of paucity and inadequacy of
theories for psychology in HCI— to move beyond the information-processing psychology. This seems to be the only choice because information-processing psychology obviously cannot fulfill the task of directing HCI design, not even to mention localization. Therefore, we need to search outside information-processing psychology for a theory that can enable HCI design and software localization to accommodate the complexities of HCI activities in all kinds of environments, be it in any culture or country.

3. Activity Theory as a Foundation for HCI Design

In this section, I will introduce activity theory in Soviet work psychology as a guiding principle for HCI design. My argument in this section is that since activity theory approaches human activity in terms of its goal-directedness and takes into account its social nature and the mediating role of the environment, it can guide HCI design to address the complicated factors involved in human-computer communication. Such a perception of HCI in terms of activity and the software in use will enable the incorporation of culture-specific considerations in software localization, which will be the topic of my next section. However, let's first of all see how activity theory can be applied to HCI design.

As cognitive psychology seeks to understand the human mental process in terms of the "memory, attention, perception, language, and thought in the single, unaided individual," little or no emphasis was given to the overall environment in which people accomplish their tasks, and, unsurprisingly, the role of the environment is much ignored in the study of human cognition (Norman 18; emphasis added). With such a psychology to guide HCI design, we are likely to ignore the interactive aspects of human-computer interfaces and the
interactive nature of the environment in which human-computer interactive activities are carried out. As has been reiterated in this paper, HCI needs a psychology that is "methodologically and conceptually richer, more diverse, and better specialized for application" (Carroll 5). This is also a clear theme developed in Designing Interaction: Psychology at the Human-computer Interface, which can be considered as representing the state-of-the-art research in the psychology of HCI. To find out what is appropriate for HCI, we must first of all understand what HCI wants.

HCI seeks to produce user interfaces that facilitate and enrich human motivation, action, and experience, but to do so deliberately it must also incorporate means of understanding user interfaces in terms of human motivation, action, and experience. (Carroll 1)

Obviously, information-processing psychology is unable to take up such a task. On the other hand, activity theory in Soviet psychology, to my mind, can be introduced into the psychology of HCI as it precisely fulfills the role of the "means of understanding user interfaces in terms of human motivation, action, and experience."

A. What It Is

The theory of activity was first conceived and developed in the 1920's by the Soviet psychologist L. S. Vygotsky, who introduced almost all the issues in this theory. It was then conceptualized and brought to a more full-fledged form by one of Vygotsky's peers, another Soviet psychologist, A. N. Leont'ev, "who has been primarily responsible for creating the comprehensive theoretical framework within which all these issues are interpreted today" (Wertsch 40).
This theoretical framework can be represented by the diagram shown in Figure 5 above.

It is hard to summarize the theory of activity in brief yet informative terms, but Wertsch has done a good job, especially with the following description:

We begin to see how some of the various features of the theory of activity fit together into a coherent framework when we consider that in Vygotsky's school, the notion of internalization is concerned with the ontogenesis of the ability to carry out socially-formulated, goal-directed actions with the help of mediating devices. (32)
To understand the theory of activity well, one must understand the constituents of activity. According to Leont’ev, activity has two groups of components:

1. need motive goal conditions for achieving the goal
   (the unity between the goal and the conditions is a task);

2. activity act operation. (Davydov, Zinchenko, and Talyzina 35)

Put in another way, it can be described as this:

A motive impels a human being to set a task and to define a goal, which, being posed under definite conditions, requires the accomplishment of an act aimed at the creation or the procurement of an object that satisfies the motive and need.

The nature of the act carried out to resolve the task is determined by its goal, and the conditions of the task determine the operations necessary for resolving it. (Davydov, Zinchenko, and Talyzina 36)

These components form the structure of activity as an integral system.

What is complicated and also important about activity is that its constituent parts may interchange and mutually transform, depending on the conditions. For example, an activity may lose its original motive and transform into an action or operation, thus becoming a part of a different activity. Conversely, an action may acquire a motive and become an independent activity in its own right. Leont’ev calls this phenomenon “mobility” (65-6).

Another important point Leont’ev makes about the theory of activity is the objectiveness of activity. Contrary to the old notion of “non-objective activity,” activity theory postulates that any activity has its object. According to Leont’ev, the object of activity appears in two forms: “first and foremost, in its dependent existence as subordinating and transforming the subject’s activity, and
secondly, as the mental image of the object, as the product of the subject's detecting its properties” (48). This notion of objectiveness is the key point of activity theory. It provides us with a new perspective to look at our knowledge acquisition and the relationship between us human beings and the world. According to Leont'ev, human knowledge is mediated by human interaction with the world. This poses criticism of two contrasting notions: the epistemological notion that humans are passive receptors of reality and thus the structure of the external world determines human knowledge and the idealistic notion that humans create knowledge by conscious reflection. Rejecting both notions, Leont'ev and his followers claim that “neither the external world nor the human organism is solely responsible for developing knowledge about the world” and that “the key to the process is the activity in which the human agent engages” (Wertsch 38; emphasis added). This explains why the notion of human activity plays a central role in Soviet psychology.

Activity is considered the most important concept in Soviet psychology by Wertsch, who specifies the following six “central defining features” of the theory of activity:

1) The three-level analysis of activities: activity, actions, and operations
2) The conscious goal-directedness of human actions
3) The mediation of human activity
4) The developmental or genetic explanation of mental processes
5) The social nature of human activity
6) Internalization.

The first feature of activity theory is its varying levels of analysis. According to A. N. Leont'ev, there are three components, and therefore three
levels of analysis, to activity, namely, activities, actions, and operations, which are distinguished by their motive and object, their goals, and conditions of their execution respectively. These levels of analysis are defined by functional criteria, as a result of which, an action involved in an activity in one case may be considered an entire activity in another, depending on its functions. For example, the execution of the movement of the mouse on a computer by a skilled user is constituted by unconscious operations whereas to someone who is using the mouse for the first time, operating the mouse becomes a conscious goal-directed action because he is learning its execution.

The second feature of the theory of activity is its notions of "goal" and "goal-directedness." These concepts are directly involved at the level of analysis of actions, but they have an important place in the theory of activity because, as Marx and Engles have pointed out, conscious goals and conscious goal-directedness are what distinguish human beings from animals, which may exhibit goal-directed behavior but are never guided by conscious goals. The importance of this concept of goal is also reflected in the fact that goal formation is one of the major research topics in Soviet psychology.

The third feature of activity theory is the mediation of activity. It is a known and accepted concept that tools mediate human behavior. Vygotsky, however, brought this point further. He proposed the notion that human behavior is also mediated by sign systems, including, but not limited to, language. He observed that signs are a means of controlling human behavior. Sign systems can mold reality and shape human minds to fit the societal structure. Signs transform reality rather than just passively mirror it.
The fourth feature of the theory of activity is its emphasis on developmental or genetic explanation. Vygotsky and his followers have defined the most important way of explaining human mental processes to be the examination of their origins and development. This developmental approach was considered by Vygotsky to be the central method of psychological science and has resulted in the study of onteogenesis, or, in the Western term, "developmental psychology." For instance, the history of the development of the Chinese language is crucial to the development of some of the artifacts in Chinese Language Kit, such as the different input methods for different people who have learned Chinese in different ways (I will elaborate this point in Chapter IV).

The fifth feature of activity theory is that social interaction makes human activity and its means of mediation possible. This is closely related to the fourth feature, the importance of developmental approaches. It specifies the social nature of human activity. Social interaction provides a reason, a need, for human activities. What makes this feature unique is Vygotsky's claim that in a child's development of the skills and modes of mediation for an activity, there is a "gradual transference of links in the activity's functional systems from the interpsychological to the intrapsychological (i.e., from the social to the individual) plane" (Wertsch 30). For example, a child who grows up in a bilingual environment internalizes bilingual interaction as a natural way of social interaction in the culture(s) in which he lives.

The sixth major feature of the theory of activity is internalization. This feature is closely associated with the last. According to Vygotsky and Leont'ev, at a certain stage during a child's development, there happens a process of
internalization, i.e., a transition from the external to the internal plane and a transition from collective to individual activity. What is important and unique about this notion of internalization is that the process of internalization does not consist in the transition from the external plane to the internal plane but in the very formation of the internal plane (Davydov, Zinchenko, and Talyzina 34).

If we reiterate and imitate Wertsch's summary, we can understand activity theory in the following terms:

With its multi-level analysis of human activity, the theory of activity takes a developmental approach to the study of how humans perform socially-regulated, conscious-goal-directed actions with their internalized knowledge and skills and with the help of mediating devices.

B. Activity Theory: Its Implications for HCI Design

As has been reiterated several times in this paper, HCI design is the foundation, and thus an inseparable part, of software localization. To discuss how activity theory can be applied to software localization, we must first of all examine the applicability of this theory in HCI design. With the comprehensiveness of its perspective of the HCI context including activity, human intentionality, individuals, social group, artifacts, etc., the theory of activity provides some useful implications for HCI design.

First, human-computer interaction is an activity-driven process made up of conscious-goal-directed actions (feature 2). It is a constant process of specifying and creating conditions for performing certain actions toward the accomplishment of certain activity. The computer, its applications, and its
interfaces all support the purposeful human activity (feature 5), and therefore their design should fulfill their supporting functions. In other words, all aspects of human-computer interaction should be conceived in specific use situations and their roles be considered as supporting purposeful human work activity. This requires designers to move beyond the interface, that is, to focus on goal-directed actions rather than the details of the interface per se. A logical conclusion of this is that HCI design needs to shift its emphasis from surface representations of systems in terms of interaction style to understanding their use.

Second, because of the interplay between humans and reality, activity transforms objects and is transformed by them (features 3 & 5). Through the computer, human activity produces artifacts—new objects of the world. On the other hand, artifacts (objects) mediate and transform human activity (feature 3). In this regard, human activity, with its possibilities of action and performed by means of computer, transforms or produces the appearance of the objects, which are presented by the interface but without displaying the underlying structure of the computer systems. This creates problems for the human user, who, without the knowledge of the inner workings of the system, may find it hard to associate the objects in their appearance presented by the interface with their appropriate mediating functions. This puts a high requirement on “the correspondence between appearance and effective implementation,” which should be “guaranteed as far as possible” in software design (Raeithel 398).

Third, due to the dual nature of human activity as acting on and being acted on by other activities (feature 5), the social nature of activity must be taken into account in the design process. Since human activity is transformed by
existing artifacts (productive means, or "tools," resulting from other activities) and by communication and reflection (realized by means of semiotic means, or "signs"), it affects not only individual thinking, perception, and regulation of action (in other words, "self regulation") but also shared ways of thinking, shared views, and culturally patterned actions inside social groups (social regulation; features 4 & 6). Therefore, it should be a serious concern of software designers what function computer-supported individual activity has in self-regulation and social regulation; what effect our own perception and ways of human-computer interaction have on society; what shaping impact software design has on the structure and content of collective activity, and hence on the ways of thinking of the members of the group, and further, on the development of social structure.

Fourth, since computer use aims at either the object end, to create artifacts, or the subject end, to communicate with others (feature 1), "the subject/object-directed aspects must support the development of operations" (Bodker 2). Whether such aspects support instrumental actions and operations toward objects or communicative actions and operations towards subjects calls for different characteristics. To provide such support and also support for the shift between objects and subjects, software designers need to incorporate such considerations as consistency in handling the artifact at least for the same type of situations.

Fifth, artifacts as mediating tools should be viewed not merely as things but also in their use. Artifacts are objects around us for us to reflect upon, but they are also used by human beings to mediate their activity, which role is reflected in both individual practice and communal praxis, the "cooperative reproduction of means by social recreation of common forms" (Raeithel 395) or,
in Bannon and Bodker's terms, "the ways of doing work, grounded in tradition and shared by a group..." (241, feature 4). There are two aspects to the mediating function of artifacts. First of all, when an activity is performed by a living body using the natural bodily means, natural abilities of humans are involved and are indispensable, and such abilities have to be manifested "as historically evolved skills" (Raeithel 397). In the evolution of such skills, the involvement of artifacts is unavoidable. The second aspect is that artifacts serve as the means of an "extended body" to mediate activity. When we use the activity approach to look at interfaces and even computer applications as artifacts mediating certain activities, we are shifting our emphasis from simply the nature of the interface per se to the use of computer applications (Bannon and Bodker 243). Thus, the activity approach focuses on "how computer application appears to its user in use" (Bannon and Bodker 245).

Sixth, based on the fifth implication, the user interface, which is the major concern of many researchers in HCI and is sometimes over-emphasized in terms of its nature per se, should be viewed as an artificial artifact in its use. User interface cannot be viewed independently of its use aspects such as the goal or object. Often, the user interface is unjustifiably separated from the application, losing as a result its dynamic aspects. From the perspective of activity theory, application should determine the interface, and it is the use aspects of the user interface that we should focus on: how do they support or constrain different actions and operations on a subject or object? "A good user interface allows the user to focus on the objects or subjects that the user intends to work with" whereas a bad user interface may force the user to focus on unintended subjects and objects (Bodker b 77), or worse, to focus on the interface itself. A good user
interface should also achieve flexibility and simplicity to support the shift of action level, in other words, the mobility of activity, action, and operation levels, which is my next point.

Seventh, the user interface must be flexibly designed to cope with the mobile nature of the levels of activity (for example, an action may acquire an independent motive and become an activity in its own right; or it may lose its original goal and turn into some operations towards the fulfillment of some other action) (feature 1). For example, in the case of an operation breakdown due to some unpredicted conflict between an operation and its material conditions, the operation becomes a conscious-goal-directed action, and the user interface must be able to provide adequate and flexible means for the user to smoothly shift between levels of action and recover from the breakdown. Errors and breakdowns are inevitable, but it is less than adequate to provide only means of avoiding them. "From the human activity theory the continuous encounters with 'errors' (breakdowns) are the driving force, both in understanding how a certain artifact works in real use, and in understanding how the artifact eventually gets changed" (Bannon and Bodker 249).

Last but not least, since software systems including user interfaces must be designed according to the use practice, the design activity will encounter conflicts between different practices involving different needs, different material conditions, etc., which is an especially important concern in software localization. Since design activity is a conceptualization of past practice and visualization of and creation for future practice, it is dealing with a practice which does not yet exist. A proper starting point for design would be to base itself on the existing practice and "let the users and their practice be the origin for
design" (Bodker b 143). The design of the user interface, which should be present throughout the design process, should be integrated with the design of other aspects of the computer system by user interface experts in close association with other design experts and users as well. The aim of design should not be to design a system or user interface that can handle all kinds of use practices and address all kinds of needs and material conditions, which would be a far too ambitious goal for designers, but to design a system or user interface that can address the generalized aspects of different use practices so that it is flexible enough to be readily adjustable to specific use situations.

4. Activity Theory, HCI Design, and Localization

Now that we have seen the implications of activity theory for HCI design, we can see how this theory can be applied to software localization, which takes HCI design as its foundation and a prerequisite. Since HCI design and software localization are so closely connected to each other and since, as I have explained, activity theory can be applied to HCI design, we can logically conclude that there are implications we can draw from activity theory and apply to software localization.

Ideally speaking, a localizable software product or system should be so designed that all its components and aspects, including, of course, its interfaces, readily support different activities with different goals defined by different cultures or use practices. An ideal interface as a mediating tool of different activities, then, according to Sukaviriya and Moran, should have a user interface so generic that it enables the user to choose any language, any culture, and any behavior in which the user expects to interact with the computer to perform the
activity in mind (189). Such an interface is of course far from being realizable today. However, a realistic direction of pursuit for software designers and localizers is to develop user interface software tools that can distinguish between culture-dependent and culture-independent facets (Sukaviriya and Moran 189).

The traditional view of an adaptable user interface looks at the user as someone whose computer skills naturally increase with time invested into computer use. As a result, the cultural-variant aspects of the user interface are ignored. Language translation and graphical adaptation are often only adaptations at a superficial level. Only by designing the user interfaces and the other aspects of the computer system in terms of their mediating functions in performing different human activities and allowing flexibility in incorporating cultural considerations can we produce successful localizable software products. In this regard, an activity theory approach toward HCI design should yield internationalized software products as they are designed on the basis of general human practice and therefore afford considerable ease for localization. On the basis of such a human activity approach, some helpful insights can be drawn for software localization. We will look at seven aspects of activity theory as they apply to localization.

First, as human-computer interaction is driven by goal-directed actions, and as the system, its applications, and its interfaces must be understood in terms of their mediating functions in their use, the range of specific functions and the specific activity of the target locale they support become the basis of adaptations. The surface representations of the interfaces, though still expected and by all means designed to meet the cultural expectations of the targeted users, are not the central focal point of designers. Instead, the forms of the interfaces are
determined by their functions, i.e., the activities they help to accomplish, for the same interface can have different functions in different cultures depending on the different activities they support. Thus, the specific activity they are expected to support in the target locale will determine how the interface will appear to its user, what aspects are to be retained, curtailed, or added to the interface, what type of interaction style is to be adopted, and so on. All these will be considered in terms of the influence of social use practice on the specific activity, whether individual or collective. For example, in translating an icon, not only may the icon need to be redesigned as it may have a different meaning in a different culture and sometimes even in different use situations, but the name associated with it, which is often mnemonic, also needs to be changed because its embedded text may have changed. Therefore, for the English initial W representing Write to function in a localized French version of the system, W for Write will have to be changed into E for Ecrivez, and if the W stands for Word, it will have to become M for Mot in French. This all depends on the function of the icon in a specific situation.

Second, as activity transforms objects and is transformed by them, activities affected by different material conditions in different cultures may produce different objects or transform objects in different ways. Accordingly, the activities will be transformed by objects in different manners. This means that since the same activity performed by means of the computer in different cultures may lead to the creation of different artifacts, the interaction style and the resulting artifacts will exert different effects on the interfaces, on the original activity, and, further, on future activities. Therefore, the form of the interface selected and designed for a specific type of activity in a specific culture should
correspond with the way it functions to mediate the activity. Take date format on computers for instance. A form like 06/07/94 means June 7, 1994 in America, but in European countries, it may mean July 6, 1994. If such a form is used in a European country to mean the former, imagine what kind of negative effect it may exert on the end product of computer, how it will affect the user's performance, and how it will affect other people, such as users of the end product.

Third, since human activity acts on other activities, the social nature of activity requires localizers to design system functions that will support the type of activities that can fit into the social structure of the target locale so that such activities will naturally contribute to rather than impose on the structure and content of the collective practice. Only such a design can lead to a positive impact on the social regulation of the target culture, which in return will yield a positive effect on individual self-regulation. For example, when designing symbols and icons of an interface, an appropriate combination of internationally accepted symbols and icons with those unique to the country for which the software is being localized is more likely to increase the informativeness of the interface and its mediating functions and therefore exert a positive impact on the user practice of the country than to just use internationally accepted symbols and icons or, even worse, those specific only to the country for which the software was originally designed. For example, the little-birdhouse-shaped icon for the mailbox is a familiar symbol in the United States. However, when this same icon is applied in, say, the Chinese culture, it may not be recognized because mailboxes in China are not shaped that way. Therefore, in localizing this icon for Chinese users, adapting it into a little square box with the universal mail
symbol, a letter, would be more effective because the box plus the universal letter icon would be more easily recognizable to Chinese users. Such an icon would be a combination of internationally and locally recognized symbols and would be more acceptable to the target users.

Fourth, since the human subject uses the computer either to create objects or to communicate with another subject and since artifacts (applications, interfaces, etc.) thus are used to mediate either between the subject and the object or between two subjects, all the operations of the subject (the user) are directed towards the artifact. In the case of localization, whether certain aspects of an artifact support the object end (to create an object/artifact) or the subject end (to communicate with another subject) depends on specific use situations conditioned by the conventions of the culture. Localizers thus need to be sure of which properties and characteristics of artifacts effect actions and operations toward which end. What forms should be used to enhance users’ ability to carry out such actions and operations depends on the specific activity but is not so important an issue as the consistency in the design of the forms.

Fifth, activity theory argues that artifacts as mediating tools should be understood in terms of shaping and extending human capabilities. Such a shaping effect and extending functions are different in different cultures, depending on the purpose of use. What counts is not the form of artifacts but the specific use they are put to. One of the concerns of the designer should be to design computer-based artifacts in view of their cultural-specific use situations so that they won’t become the objects that users work on but be mediating tools in support of the activity. For example, in terms of text direction, Roman scripts run characters from left to right on horizontal lines that go from top to bottom.
Arabic, on the other hand, runs text from right to left. More complicated are such languages like Japanese and Chinese, which, though they may run texts in the same way as English, may also adopt a different directional flow of text, i.e., characters written from top to bottom on vertical lines that flow from right to left. For example, we need to design a device on the computer interface that will afford easy shifts between different text directional flows. If this device (an artificial artifact) is well designed and can accommodate any text directions, it will provide considerable ease for the user and fulfill its mediating function. If this device is ill designed without considering the culture-specific factors and unique text directions, the user will have to spend much time coping with the unhandy artifact in order to figure out a way to change text direction when needed. In such a case, the artifact is no longer a mediating tool but an object the user has to work on.

Sixth, extending this last point, we should look at user interfaces also as mediating artifacts and design them as such. Depending on the culture, some aspects of the user interface may appear more familiar to users of one culture while other aspects may seem more comfortable to work with to users of a different culture. Since the user interface should be designed as a helpful mediating tool to effect its operationalized use by the user, the more unfamiliar aspects that are likely to cause the interface to become the object of action by users of a certain culture should be so designed that they will provide the maximum ease of use and support for the intended activity and avoid being the object of action. For instance, in localizing the user interface of an American computer system into different cultures, features like character representation, input methods, keyboards, number formats, etc. may appear familiar to French,
Spanish, German, and many other European users whereas sorting orders, address, date, and currency formats, colors, etc. may seem less familiar and are likely to cause more problems. With Asian users, it is a different case. They will find more aspects of the user interface of the American system problematic than their European counterparts. Therefore, in view of cultural considerations, designers and localizers need to focus on those aspects that are more likely to cause problems to users of a specific locale.

Seventh, since activity, action, and operation are of a mobile nature and can interchange at any time, the flexibility of the user interface to cope with such mobility is especially important in a localized software product. While some operations are so easy and operationalized to users of the original culture for which the software was designed, to users of the target culture, they may become difficult maneuvers that require conscious actions and in some cases may even become activities in their own right. This causes breakdowns and will considerably affect the ease of use and the accomplishment of the activity originally intended. For example, while American users are so familiar with such terms as "mouse" and "menus" that they have already internalized their concepts and mechanized (operationalized) their operations, Chinese users may find it hard to grasp their concepts and thus even hard to locate them on the interface, not to mention operationalize their maneuvers. This is because Chinese usually name devices according to their operational functions or a combination of functions with appearances instead of merely their appearances. This is not to say that after a certain period of use the Chinese user will not be able to learn the terms and concepts and their use but that since these devices are termed in a way contrary to the custom of his culture, there will be a conscious
element in the user’s mind to remind him of such an anomaly even after he has familiarized himself with the terms and their use to a certain extent. One cannot tell if or when this may cause an occasional breakdown to the operations and change the level of action. This is only a minor case of cultural differences that, if ill defined, may lead to the unnecessary mobility of levels of action. Sometimes, such cases are unavoidable, but as designers, localizers should not only try to minimize the possibility of such occurrences but also to provide means to smooth out the mobility situations.

Eighth, as a self-evident point, the design and localization of software products should be based on the specific use practice of the target culture. This means that, to accommodate the specific user needs, material conditions, ways of practice, etc., the design and localization processes should incorporate users from that very culture, who are usually the only people capable of providing a unique insight on the work environment, the work psychology, and indeed everything else concerned in the actual use from the users’ perspective. Usability testing may be seen as a positive effort, but to include users only in this stage is no more than a make-up measure and will inevitably fall short of a full consideration of the factors in the interactive environment of computer use.

Even when designing and localizing a program for different users of the same culture, different needs from these users have to be incorporated. For example, a Chinese user interested in using a software system for desktop publishing purposes will have vastly different needs from another Chinese user who is interested in the system mainly to use e-mail. The desktop publisher would probably like to see such features as text-directional flows, different Chinese fonts, graphic-making devices, ready made pictures and icons specific to
the Chinese culture, and any other artifacts in the system that will help make his
desktop publishing job easier whereas the e-mail user may not be interested in
these features but that the system will enable him to create Chinese characters
(and English text if he has the need to correspond with an English speaker). A
consideration of the different user needs in design and localization will reflect
the emphasis of activity theory on actual use.

Software localization is a complicated process which involves a number of
factors that can hardly be exhausted. The past translation approach and the
existing three-step designing-writing-translation model have proved inadequate.
The human activity approach provides a unique perspective by looking at
human-computer interaction in terms of actual use. To sum up, the human
activity approach I have proposed here toward software localization introduces
the theory of activity into HCI design so that human-computer interaction can be
designed in terms of its dynamic and interactive nature, by means of which
software products can be successfully internationalized so as to be readily
localized into any culture-specific locales according to its unique use practices.
With its flexibility and comprehensiveness to accommodate cultural differences
arising out of different use practices, the human activity approach, I believe, will
provide a feasible alternative for software localization.
1. An Overview

In this section, I will conduct a case study of Chinese Language Kit, a localization tool and an extension to Macintosh System 7.1, to demonstrate how the human activity theory approach can be applied to software localization practice and how Chinese Language Kit illustrates an innovative localization process distinct from the conventional one.

Before we talk about Chinese Language Kit, however, a distinction has to be made between the conventional system software localization and localization by means of language kits employing WorldScript technology. In the conventional localization practice, localizers convert the system software in the original language into the target language, by which they have actually created a new version of the system software to replace the original one for the target culture. If the system software needs to be localized into 10 different languages, localizers have to create 10 different versions of the system because the original system is unable to support different scripts of language. Two problems usually result from this approach: one is that localization becomes a time-consuming process; the other is that it makes it hard to use multiple languages on the same computer as it is difficult and takes a lot of space to install several systems on the same computer, and it is very troublesome for the user to switch between systems.

Language kits, however, represent a superior alternative to the conventional localization method. By employing WorldScript technology, a technology that enables the system to accommodate any script, the original
operating system acquires the ability to provide support for multiple languages, and therefore it no longer needs to be replaced with a localized version. Only tables of script-specific features and culture-specific behavior need to be developed as part of system resources to be used whenever needed. Language kits provide exactly such resources for the system. Language kits often contain such components as character sets, fonts, input methods, language-specific dictionaries, etc. Language kits are designed to simplify and quicken the localization process and to better serve user needs by making possible the independent development and implementation of any script system and the use of any language kit on any Macintosh anywhere in the world.

2. WorldScript and Language Kits

With over half of its Macintosh computers sold in markets outside the United States, Apple Computer, Inc. is now seeing its computers used all over the world. Localized versions of Macintosh systems software have been available for years, but it was in April 1992 that Apple first developed and introduced its Language Kits to the global marketplace. So far, at least two language kits have been developed: Japanese Language Kit and Chinese language Kit. These language kits are based on the WorldScript technology for the Macintosh computer, released as part of System 7.1 in October of 1992 (Apple Computer, Inc. 1994 2).

Macintosh System 7.1 has been localized into 35 different versions. Each version represents a different language. These languages have different characters, which can be divided into two categories: one-byte and two-byte scripts. Languages like English, French, German, Spanish, etc., have Roman
characters and require only one byte of memory to fully represent their character sets. Some other languages like Arabic and Hebrew use non-Roman characters which also require one byte of memory to represent each character. However, other languages, such as Japanese and Chinese, have character sets that require two bytes of memory to represent them. One problem with such a difference between one-byte and two-byte scripts is that one-byte scripts usually contain no more than 256 characters whereas two-byte scripts can contain as many as 50,000 characters or even more, such as in Chinese, which means it is impossible for the operating system design of one-byte scripts to support two-byte scripts. This requires system designers to design different representational forms and input methods (Apple Computer, Inc. 1994). 

Another problem arises out of the distinction between Roman and non-Roman languages. Roman languages use similar characters and are thus easy to support in the base operating system. Non-Roman languages, however, use very different characters that are very different and often impossible to support with one system. Therefore, when localized, a software system like Macintosh System 7.1 requires designers to reengineer many different language systems. This creates problems especially for multiple-language users, who, in order to use two or more languages on the same computer, have to keep multiple systems on the desktop, thus wasting valuable space and time storing and switching between the systems. Such a user is also likely to suffer delayed delivery of new technology due to the need to reengineer the system, inconvenient and inefficient text-input methods, and limited product selection. This also creates problems for developers, who have to waste a great deal of time on time-consuming localization, the implementation of features in different languages, and
complicated updates. For Apple, it has to conduct time-consuming localization and suffer difficulty in penetrating new markets (Apple Computer, Inc. 1994 8).

The creation of WorldScript technology and the language kits has provided a solution to these problems. WorldScript supports multiple languages in the base operating system and makes the functionality of localized systems available on all systems while language kits sold separately provide a means of delivering the functionality to end users with different needs. With its ability to support multiple languages, WorldScript has revolutionized the design of script systems. A script system for a language determines the various components of the writing system: character encoding; keyboard layout; number, currency, date, and time formats; input methods; etc. Without WorldScript, all scripts have to be developed independently, which necessarily means that all the components of different writing systems have to be designed separately. When all these scripts are installed together with their respective modifications to the overall system, conflicting features and behavior often arise and therefore affect the performance of the system. With WorldScript, only a generalized script system is used that defines the common rule and behavior of all scripts. All script-specific features and behavior are defined by tables in system resources, with any script being supported by either of the two extensions of WorldScript: WorldScript I, which supports one-byte scripts, and WorldScript II, which supports two-byte scripts. These two extensions can exist and function together, with system resources ready to support any script-specific behavior at any time. With such a system, any additional script can be developed independently of the system to be added to it at any time through a language kit. WorldScript makes it possible to support multiple languages in a single system software version.
WorldScript provides an excellent means of internationalizing software products, greatly reduces localization time for developers, thus making it quicker to deliver the product to users' hands, and improves text-input methods for some users, especially Asian users. This internationalization is a key step to the localization of a product into multiple cultures, and it makes languages kits possible. Language kits provide system-level support for additional languages on a single system by installing language resources, an input method, and fonts for a specific language. Compared with the localized system, a language kit is similar in that it has the same functionality as the localized system and different, and also superior, in that it can be installed on the English system, and in fact on any language system. For example, Japanese Language Kit can be installed on the French version of Macintosh System 7.1; Chinese Language Kit can be installed on the Japanese system whereas Japanese Language Kit can be used on the Chinese system. Theoretically speaking, this makes possible any combination of languages on the same system provided that the language kits needed are available.

3. Design Rationale and Target Users

With the world fast becoming a global village, more and more educated people are able to speak one or more foreign languages, among which English is undoubtedly the most popular and most widely used world language. Such a trend is also found in China, where more and more people are learning and using English. There is an increasing need among computer users in China to use two or more languages on the same system. As a result, the two major computer companies in China, Great Wall and Stone, are producing computers
that have both Chinese and English language systems. What such a fact suggests is that a localized Chinese version of System 7.1 that supports the Chinese language system only may prove to be inadequate to many Chinese users, especially when they want to combine Chinese with English or some other language in the same document. Under such circumstances came the birth of Chinese Language Kit.

Chinese Language Kit targets not only users who use computers mainly to produce Chinese documents but also those not usually accommodated by the conventionally localized version of System 7.1 in Chinese. There are four types of such users:

1) Those Chinese-speaking people living and working outside their countries of origin who need to create both English and Chinese texts in their documents

2) Providers of foreign language (including Chinese) services or products, such as translation companies

3) Education users such as teachers and students who need to use both Chinese and English in their teaching or study.

4) Those with an occasional need to use Chinese on their computers, such as those businesses that deal with Chinese-speaking customers.

The first type of such users is those Chinese-speaking people working outside their country of origin, be it from mainland China, or Taiwan, or Hongkong, or Singapore, or Malaysia. Working in countries where some language other than Chinese is spoken, these people often find the need to use on the computer system both Chinese and the language of the country where they work. For example, a Chinese person working in Los Angeles is most likely
to be running the English version of System 7.1. With the installation of Chinese Language Kit, he will be able to use his favorite Chinese word processor (e.g., Word Perfect 2.2 Chinese) so that he can produce a Chinese document or one that combines Chinese and English. Besides, files created on the Chinese system can also be used in Chinese Language Kit. The user thus has only to keep one system on the computer instead of two. And if this Chinese user has a Japanese colleague who wants to use Japanese on the same computer, Japanese Language Kit can be added to the system, again without having to add another system to the computer. Apple claims the Macintosh to be the only computer in the world to have such a capability (Apple Computer, Inc. 1994 12).

A second type of target users for Chinese Language Kit is providers of foreign language services or products. These users, wanting to expand their business, need to create documents, brochures, sign boards, posters, advertisements, business cards, restaurant menus, product labels, etc. that use one or more languages including Chinese. They include publishing houses, multinational companies that have business in and/or with Chinese speaking countries, translation companies, churches, schools, and any other businesses or individuals having the need to create Chinese-related bilingual documents. Many of these users have been using fully localized systems in Chinese, but Chinese Language Kit will bring them greater ease in handling two or more languages at the same time.

A third type of users is education users who need to use both Chinese and English and/or some other language. Bilingual teachers and students may need to create letters, resumes, papers, assignments with both English and Chinese; language labs may need to run interactive language learning programs
combining English and Chinese. This group includes people who study Chinese as a foreign language and those Chinese students and scholars who study or work overseas. They often comprise a substantial portion of potential consumers of Chinese Language Kit.

A fourth type of users are those with an occasional need to use Chinese on their computer. This type includes those non-native speakers of Chinese who may speak Chinese fluently or a little but have a need to use Chinese occasionally in documents or letters. Many American business people, for example, frequently travel to Chinese-speaking regions and are able to speak some Chinese. Though they work primarily in English, they may have the occasional need to insert some Chinese characters to more accurately describe concepts, proper names, geographical names, etc. Of course, they have the option to use Roman-script phonetic transcriptions like pinyin in place of ideographic Chinese characters, for example, zhong guo for 中国 (China). However, due to the large number of homophones, which is probably unique to Chinese, such Roman-script phonetic transcription may sometimes be very ambiguous and thus inaccurate in its representation as no single Chinese character has one and only phonetic transcription unique to itself. For example, a Roman-script phonetic transcription such as zhong represents dozens of different Chinese characters such as 中 (middle, center), 钟 (clock), 忠 (loyal), 终 (finish), etc., and the absence of its proper tone will add many times more characters represented. This also explains why using pinyin (the Roman-script phonetic transcription) is still such a challenge to most Chinese, even educated Chinese, today. Using Chinese Language Kit to present accurate ideographic Chinese characters may
improve the business efficiency of and bring benefits to business people even though they may need to use Chinese only occasionally.

It is obvious that unlike a conventionally localized version of the Macintosh System 7.1 in Chinese, which targets users only in Chinese-speaking regions like China, Taiwan, Hongkong, etc., Chinese Language Kit also targets such countries as the United States, Britain, Canada, Australia, and so on, where there is a population of Chinese speakers and users. These Chinese Language Kit users provide a good potential market also for other multilingual software programs that are compatible with WorldScript-savvy systems and applications.

4. Chinese Language Kit Examined from the Human Activity Perspective

Chinese Language Kit, both as a tool and product of localization, tackles localization issues in a unique way. In terms of its overall design, it bases its design on actual use while taking into account differences in use practices between Chinese and American cultures. Since Chinese Language Kit does not change Macintosh System 7.1 but only adds to it features specific to the Chinese language and cultures in the form of tables as system resource, it is able to incorporate considerations of Chinese cultures while retaining the advantages of the original English version of System 7.1. Examined from a human activity perspective, Chinese Language Kit displays some distinct features in terms of its design and use. I will analyze seven of these features in terms of both merits and shortcomings of the program.

First, designers of Chinese Language Kit have designed two versions: Traditional and Simplified Chinese. Chinese is used as a native language in several countries and regions, mainly in mainland China, Hongkong, Taiwan,
Singapore, and Malaysia. Due to historical reasons, two different systems of the Chinese language are used in these regions: traditional and simplified. Simplified Chinese, as a simplified version of traditional Chinese, emerged mainly as a result of the New Culture movement in China around the 1920's, which aimed at educating the masses by, as one of the several means, simplifying the ideographic representations of Chinese characters so that they would be readily teachable and learnable. Simplified Chinese typically uses fewer strokes to represent characters than traditional Chinese. Due to geographical separation, lack of communication, and probably other reasons, simplified Chinese was not popularized in all Chinese-speaking regions. It is now used almost exclusively in mainland China while traditional Chinese is used in the other regions. To accommodate both types of users, designers of Chinese Language Kit have created two versions, Traditional and Simplified. This not only satisfies the need of a user to use his preferred type of Chinese but also meets the need of some users to occasionally, and in some cases often, create a document in the type of Chinese they do not conventionally use. For example, a business company in mainland China which usually uses simplified Chinese in their domestic business relations may need to use traditional Chinese in order to communicate with their business partner in Hongkong. The same is likely to be true vice versa. This distinction between traditional and simplified Chinese is of course something one cannot afford to turn a blind eye to in designing Chinese language software, but in actual design, to address such a distinction involves the double design of almost every aspect of the program: fonts, input methods, dictionaries, etc. Chinese Language Kit certainly reflects a great deal of efforts in this respect. Therefore, designing two versions, Traditional and Simplified,
reflects a conscious awareness on the part of designers of the difference between users, their cultures, and their needs in actual use. Such a design is also a reflection of the principle of social regulation of activity theory, i.e., human activity acts on and is acted on by other activities.

However, in terms of the distinction between traditional and simplified Chinese, Chinese Language Kit appears to have ignored, or at least inadequately addressed, an important aspect—the rhetorical differences between the two. When it set out to simplify Chinese, the New Culture Movement simplified not only the ideographic representations of Chinese characters but also the structure of the language, both linguistically and stylistically, as the old Chinese (evolved from archaic Chinese) was verbose and difficult to understand. This to some extent changed the rhetorical structure of the Chinese language. Traditional Chinese, on the other hand, has retained more of the original rhetorical forms of archaic Chinese. Though such rhetorical differences are not readily displayable on the interfaces where no sentences of extended length but short command terms are used, they are likely to be shown in the manuals. However, an examination of the two Chinese manuals seems to suggest that at certain places in the Simplified Chinese Input Method Guide, the style seems to be more of the traditional one. As a result, a native speaker of simplified Chinese who is also fluent in English may prefer to read the English manual rather than the Chinese one. Examined in terms of the principle of social regulation of activity theory, this problem reveals an inadequate consideration of the history of the Chinese language in the design of Chinese Language Kit.

Second, since the emphasis of activity theory on the specific goal-directed actions performed by means of the computer requires designers to shift their
focus from the surface representations of the system to their actual use and since the interfaces must be understood in terms of their mediating functions, what actions, or activity in a broader view, that the interfaces support should be the basis of design. Designers of Chinese Language Kit have made two types of efforts in this regard. On some occasions, they have created two forms of representations to fulfill the same function since this is needed sometimes when the same function has to be completed in different ways in Chinese and English. For example, punctuation marks and some commonly used signs and symbols have to be created on the interface in different ways in Chinese and English as they take different forms. Insertion marks like brackets and parentheses in English, for example, can usually be produced by directly typing the corresponding keys on the keyboard. Chinese, however, since its characters can be written in vertical lines, have different and more insertion marks, many of which are not represented on the keyboard. Therefore, designers of Chinese Language Kit have created a special command called Punctuation Marks and Signs and Symbols which, when prompted, will present a pop-up window of all the available punctuation marks, signs, and symbols for the user to choose from. On other occasions, the same commands are sometimes used to perform different functions in English and Chinese. For example, the keyboard shortcut commands such as Option-Shift-C for Pinyin input method and Option-Shift-S for two-byte Roman input method in Chinese may represent different functions in English, depending on whether the application is in the Chinese mode or the English mode, the switch between which can be easily carried out by clicking on the appropriate window menu command. Thus, in creating bilingual documents using Chinese Language Kit, different representations on the
interface may be used to perform the same function, or the same representation may be used to perform different functions. The form of the interface components is decided by its function in actual use. In terms of the theory of activity, such a design has obviously tied the human activity on the computer directly to its goals.

Third, the design of the input methods in Chinese Language Kit reflects a concern for the fact that human activity acts on other activities, objects, collective practice, and the social structure as the input methods are designed to support the kind of activities that will fit into the social structures of different Chinese cultures. This is revealed in two aspects. First, due to the huge Chinese-speaking population and the vast areas they are spread in, Chinese users in different regions differ from each other so much that they can be considered to be of different cultures. This means that they perceive the Chinese Language in different ways, especially the characters. It follows then that these different perceptions of Chinese characters will necessarily lead to different input methods since the keys on the keyboard do not match Chinese characters directly and there are not enough keys to match the great number of characters, unlike in English where there are only 26 letters and a small bunch of punctuation marks, signs, and symbols, which can easily be represented by matching keys on the keyboard. So, in addition to pinyin, Chinese Language Kit provides several different input methods respectively for users of traditional Chinese and users of simplified Chinese. For example, for traditional Chinese, there are the input methods Cangjie, Dayi, Parrot, and Zhuyin; for simplified Chinese, Wubi Xing, Wubi Hua, Quwei, and Code. These different input methods reflect different perceptions of Chinese characters based on the different socio-histories of
different groups of users and thus their different applications of the linguistic tool, Chinese.

A second reason behind the design of different input methods is to support different activities by different users even within the same geographic region or culture. Due to historical reasons, especially the evolution of educational methods, different users, especially from different generations, in mainland China, for example, have been taught to perceive Chinese characters differently. The young generation is probably more familiar with *pinyin* while the older generations are less familiar with *pinyin* but may be more comfortable with wubi xing (five-stroke shape) or wubi hua (five-stroke structure), which younger people find hard to use. If Chinese Language Kit had been equipped with only one input method, it would be able to accommodate only a part of the total Chinese user group.

One problem does exist, though, but it is hardly a problem only with Chinese Language Kit. Some Chinese users have never been taught to learn Chinese characters by means of *pinyin*, or *wubi xing*, or *wubi hua*, or some other methods. They have learned Chinese characters simply by remembering them one by one. For these users, who are not a small population, none of the available input methods would be comfortable to work with because they have first of all to learn *pinyin*, or *wubi xing*, or *wubi hua*, or *quwei* before they can learn their corresponding input methods. The input methods may not be very difficult to learn once you are familiar with the character-learning methods on which the input methods are based. However, even with experienced users,

---

* I have used *Pinyin* to refer to the input method in Chinese Language Kit and *pinyin* to refer to the method of leaning Chinese characters. The same is true with other input methods and learning methods.
these methods may not be very efficient and sometimes are time consuming. This is because in Pinyin, for example, each input has to be converted into Chinese characters and in all cases the user will be given a handful of choices. He would be lucky to find the character he wants highlighted as the first choice when the input window pops up; otherwise, he will have to move the mouse or the arrow keys to pick his choice. The most ideal input method would be that only one key is needed to type each Chinese character, just as we do in English. However, as there are far too many Chinese characters and far too few keys on the keyboard, such an ideal will probably forever be an illusion, but we certainly expect to see more efficient and convenient input methods. If we examine this problem in terms of activity theory, we can see that this problem is likely to lead to unnecessary shift of levels of action—from intuitive operations to conscious actions, especially for those who are unfamiliar with the input method.

Fourth, in terms of the mediating functions of artifacts as shaping and extending human capabilities, the design of artifacts and of the user interfaces in Chinese Language Kit has been successful at some places and perhaps not very successful at others. For example, in order to provide ease and improve efficiency in inputting Chinese characters, designers have imitated one feature of the design of conventional Chinese typewriters and created the Associated Word function. In Chinese, certain characters often go together to make up words so as to make sense or to create a new meaning. For example, 中 (middle, center) and 国 (nation, country) make up 中国 (China), which is obviously different from when the two characters are used separately. In the case of the Chinese typewriter, when it is bought new, the typewriter comes with several boxes of chinese characters and a couple of character plates with hundreds of empty
sockets in which to insert characters. It is the task of the typist to insert the characters into the sockets on the plate in a way he prefers. When the typist designs the arrangement of characters on the character plate on the Chinese typewriter, he always tries to arrange the words that often go together to be close or next to one another. Depending on the purpose of use, the typist selects the most frequently used characters and put them on a replaceable plate, which often holds several hundred characters. The extent to which the typist successfully selects and arranges the characters is a determining factor for the extent of the ease and efficiency of use. The same is true with creating Chinese characters on the computer. To enable users to improve efficiency and ease of use, to increase the mediating function of the computer artifacts and user interfaces, and to let users shift their focus of attention from artifacts and interfaces to the actual activity, designers of Chinese Language Kit introduced the Associated Word function to the interface. As a result, using this function, when you enter the character 中 , you are given a choice of several characters and phrases that go with it, so that you can easily make up words like 中华民族 (the Chinese nation), 中国 (China), 中学 (middle school), 中毒 (intoxication), 中断 (terminate), etc. Examined from the activity theory perspective, it is obvious that such a feature extends the human bodily function in his performing the activity.

On the other hand, one possible artifact that could have benefited some users of Chinese Language Kit is not available, i.e., a command for setting text directional flows. As Chinese characters can be written from right to left on horizontal lines or from top to bottom on vertical lines, some users, especially those who use traditional Chinese, would probably prefer to see such a function available on the interface. Imagine what a user has to go through if he has to
create a Chinese document that has characters written from right to left on horizontal lines or from top to bottom on vertical lines. In that case, the user's attention will probably be focused on the interface rather than the activity.

Fifth, to maximize the ease of use and minimize the problem index of the culture-specific aspects of the interface, Chinese Language Kit has retained, added, and altered the functions of some aspects of the interface. Since Chinese Language Kit has as its main purpose to enable users to create bilingual documents, most commands and their functions in the English system are retained. However, many new features are added to the interface such as Chinese menu commands that are not available in the English system. In other cases, the functions of some devices have been altered. For example, in entering Chinese text, for the most part, the space bar loses its original function of creating spaces between words as this is seldom needed in Chinese because each Chinese character forms a distinct spatial unit, always takes exactly the same amount of space, and has a meaning by itself. As for whether two or more contiguous characters form a word to create a new meaning, it can be determined from the context. Therefore, designers have assigned the space bar a new function—to present a selection window of all the alternatives for, for example, the same pinyin if Pinyin input method is used. Of course, when the program switches back to the English mode, the space bar restores its original function. Leaving a device in the interface useless or with an inappropriate function is likely to cause problems to users, and the proper mediating function of the interface will not likely be ensured. So, Chinese Language Kit uses the space bar well from an activity theory perspective.
Sixth, efforts to avoid and enable the user to recover from breakdowns that can cause unnecessary change of levels of action (for example, a breakdown causes the user to unnecessarily focus on certain operations, which then become conscious goal-directed actions) is also evident in the design of Chinese Language Kit. For example, many dialogue windows that pop up when the user executes certain commands present the user with two options: OK and Cancel. When translated into Chinese, this omnipotent OK can easily cause problems because this word is full of denotations and connotations and there is no equivalent in Chinese that is as rich and informative as OK. Translating it into a consistent term can be very problematic because no Chinese term will be able to accommodate all the use situations that OK can accommodate. Therefore, in designing Chinese Language Kit, designers have translated OK into different terms, depending on its different functions and different use situations. Thus, in place of OK, we now have 行 (perform), 完成 (complete), 好 (good, yes), etc., depending on its use. Such a design is likely to prevent unnecessary shift of levels of action in terms of activity theory and avoid unnecessary breakdowns.

Another example is the translation of the aforementioned “mouse” and menu. “Mouse,” when translated into Chinese in the manuals of Chinese Language Kit, becomes 滑鼠 (sliding mouse), a combination of both function and resemblance in appearance. Such a translation complies with the Chinese custom of naming devices according to their function while at the same time retaining the feature of the original term. On the other hand, “menu” is literally translated into its Chinese equivalent 菜单 (a list of courses), which I find problematic when used in computer terminology. I tested this Chinese term with several Chinese students here. Some found it unacceptable; some said
“maybe” if no better terms could be found; and some found it colorful but would prefer to see it in figurative use as used in literary works but not as a technical term. All these students, however, are experienced users of the English system and have more or less internalized the concept of the term, and they commented that they did not know what they would think if they had never used the English system and were encountering this term for the first time. One thing is certain, however, that if the user finds the term problematic, it is likely to cause problems in their operations and may even lead to some kind of breakdowns.

Seventh, since Chinese Language Kit is designed for Chinese-speaking bilingual users, the actual use practice of such users should have been the basis of design and such users included in the design. The product development team of Chinese Language Kit headed by Ker Gibbs consists of a group of bilingual system designers. These people are not only experts in system design but also very familiar with the Chinese cultures. The design of the two different versions of the kit, Simplified and Traditional, of the different input methods, of the word association function, and of many other features reflects a knowledge of the Chinese cultures and a history of the development of the Chinese language.

One problem with Chinese Language Kit as a localization product is perhaps that it is designed for users who understand at least some English. As stated in the manual *Macintosh Simplified Chinese Input Method*, users are assumed to be “familiar with Macintosh operating conventions,” and if they are not, they will have to refer to *Macintosh User’s Guide* (only in English). This could be a setback to those users who do not understand English and wish to use Chinese Language Kit only to create Chinese documents. However, as mentioned earlier, Chinese Language Kit is mainly designed for bilingual users.
At a time when even the leading computer companies in China like Great Wall and Stone are producing computers that use both Chinese and English language systems, we can expect that the increasing trend of world market globalization will ease the problem as users improve their linguistic skills and computer companies come up with better localized software products.

What is discussed above is only some of the aspects of Chinese Language Kit examined from the human activity perspective. As a localization product and also a localization tool, Chinese Language Kit involves so many aspects, every one of which, to my mind, is a worthwhile topic for discussion in terms of activity theory. However, due to the limited space and my limited knowledge about computers, it is impossible for me to explore them all in this paper. Yet, from the above analysis, we can already see that the design of Chinese Language Kit is giving us some valuable implications for software localization.
V. CONCLUSION

Though computers have been developed for no more than a few decades, computer technology has come a long way and is still evolving at a tremendous pace. Likewise, software localization, which emerged with the birth of computer technology, has treaded over a long and winding path. Its evolution from the translation model to the three-step designing-writing-translation approach has reflected progress in people’s perception of localization issues.

The translation approach, though considered outdated today, played an important part in the transfer of computer technology to various regions and cultures in the early days. Translators, especially technical translators, in past decades, had been accumulating valuable experience in their practice of software translation and had been arduously exploring into the ergonomical issues involved in software translation. Though their efforts were not able to materialize into substantial, effective localization theories, they certainly constituted an essential step in people’s constant attempt to perfect localization practice and laid a good foundation for later research in this field.

The existing three-step localization model of designing, writing, and translation reflects a growing concern for the human factors in software localization, especially cultural considerations since software localization inevitably involves differing cultures. It represents a positive step in the line of research for a better approach towards software localization. However, perceiving the localization process as a sequential, linear process falls short of a full consideration of the cultural aspects that cannot afford to be ignored in
software localization, thus denying the integrative and interactive nature of the localization process.

As human-computer interaction design lies at the heart of computer design and its success determines to what extent software localization can be successful, much current literature is now focusing its attention on psychological issues at the human-computer interface and is looking at human-computer communication as an interactive process. Many researchers attempt to explain human-computer interaction in terms of information-processing psychology (cognitive psychology). However, as many researchers have realized, cognitive psychology, which assumes an identical resemblance between human information processors and artificial (model) information processors, is inadequate in directing human-computer interaction design because of the crucial differences between human-computer communication and interpersonal communication. As a result, many researchers have been exploring alternative directions, one of which is to expand the theoretical base for the design of HCI psychology.

Among many possible theories proposed to guide HCI design, the theory of activity in Soviet psychology provides a unique perspective towards human-computer interaction. Proponents of the activity theory for HCI design have claimed that activity theory enables designers to look at human-computer communication as an activity-driven process and thus to consider HCI in terms of its actual use. The distinguishing feature of activity theory is that it shifts designers' focus from surface representations of the interfaces to their mediating functions in use, which is what computer design is ultimately for.
No one, however, has attempted to associate activity theory with software localization. In this paper, I have proposed using activity theory as a guiding tool for HCI design so as to ensure successful internationalization and localization because I believe with its perspective to base software design on actual use, the human activity approach towards software localization will enable designers and localizers to incorporate into their design a full consideration of the comprehensive aspects of different use practices so that the localized software product will be able to accommodate culture-specific user needs and practices.

As an example of the human activity approach towards software localization, Chinese Language Kit represents a whole new way to localize a software product. What distinguishes it from other localized software products is its contribution to the internationalization of computer systems by providing multiple-language support to the same system, thus enabling the use of more than one languages on the same system. Theoretically speaking, the development of language kits will one day enable the combination of any and all languages on the same computer system. This will liberate localizers from the burden of creating multiple systems for multiple cultures and make localization a much easier and effective process. Language kits represent a major step forward toward the perfection of software localization. As Apple has claimed, it is already exploring better ways of software localization by participating in the development of Unicode, a new international industry standard that will make it possible to encode all of the characters used for written languages throughout the world in one table, which will make localization all the easier.

However, as has been shown in my discussion, we do see some problems with Chinese Language Kit. It could have taken into account the rhetorical
differences between simplified and traditional Chinese, especially in the writing of the manuals; it fails to provide an easier input method for those Chinese users who have not learned Chinese in a systematic fashion; it should have provided a device for setting text directional flows; it could have come up with better translations for some of the command names; and it should better accommodate the needs of those Chinese users who do not understand a single English word. Nevertheless, Chinese Language Kit represents a positive, creative localization effort in terms of the theory of activity. It points in an illuminating way to a positive direction of software localization, which is definitely worthwhile to pursue.

This paper represents one more effort to the attempt by numerous researchers and practitioners to develop an effective theory to guide HCI design and software localization. Though past research has had a limited impact on software design and localization and many researchers are far from being optimistic about the possible effects of future research, I believe persistent efforts in the exploration of this field will eventually lead us to a sound theory base for HCI and software localization.
BIBLIOGRAPHY


Raeithel, Arne. "Activity Theory as a Foundation for Design." Software Development and Reality Construction. Eds., Christiane Floyd, Heinz...


APPENDIX. COMPONENTS OF CHINESE LANGUAGE KIT

Chinese Language Kit consists of the following:

- Chinese Language support for both Traditional and Simplified Chinese and system fonts, Taipei and Beijing.
- Input methods
  - Traditional Chinese: Cangjie, Dayi, Parrot, Pinyin, and Zhuyin
  - Simplified Chinese: Wubi Xing, Wubi Hua, Pinyin, Quwei, and Code
- TrueType fonts
  - Traditional: Apple LiSung Light and LiGothic Medium
  - Simplified: Song, Hei, Fang Song, and Kai
- Installation and user’s guide in English with a quick reference guide in Chinese

The actual package of Chinese Language Kit contains 21 items:

2. CD Rom disc: CD Install
1 Notice to customer
1 System software registration card
1 Software license agreement.