DESIGNING SOFTWARE TO MEET THE NEEDS OF INTERNATIONAL USERS

A Project Report

Presented to

The Faculty of the Department of Anthropology

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

by

David Mohr

December 2013

SAN JOSÉ STATE UNIVERSITY

The Undersigned Graduate Committee Approves the Project Report Titled

DESIGNING SOFTWARE TO MEET THE NEEDS OF INTERNATIONAL USERS

by

David Mohr

APPROVED FOR THE DEPARTMENT OF ANTHROPOLOGY

Dr. Chuck Darrah, Department of Anthropology	Date
Dr. Roberto Gonzalez, Department of Anthropology	Date
Professor Awad Awad, Middle Eastern Studies, Department of Humanities, Arabic Coordinator, Department of World Languages and Literatures	Date

Abstract

Most software applications are designed and built in North America, catering to English-speaking, North American users. Ethnography is commonly used to ensure the product addresses the needs of domestic customers, but there is no analogous research with regard to foreign consumers. As a result, often times the features in these applications fail to sufficiently address the needs, expectations, and workflow of non-English-speaking users. Although software engineering manuals minutely detail the technical aspects of bringing an English application to a foreign market and further touch on common design pitfalls, there are no recognized guidelines for how to most effectively design a product to address the requirements and needs of foreign customers. This report follows a concerted attempt on the part of Adobe's Photoshop development team to design features specific to Middle Eastern users, particularly Arabic- and Hebrew-speaking users. Software engineers develop complex applications, but their cultural lenses often prevent them from perceiving the expectations of unfamiliar customers. As part of the engineering effort, the author consciously incorporated a number of anthropological techniques not commonly applied to designing software for non-domestic consumers, particularly ethnography and validation by a distributed structured network of foreign users. There is significant potential for improvement in the design and development process and anthropology appears well suited to enhance software internationalization.

Table of Contents

Introduction

Designing for Culture

We have all at some point bought a product which was designed in another country. Sometimes the clues are minor and subtle, such as engine parts labeled in German for a Volkswagon Jetta, resulting in little tangible impact to the consumer other than a bit of head scratching when looking under the hood at the engine part names. Other times the impact is more substantial, such as difficult to understand instructions for the assembly or use of toys or software gadgets. These results might range from simply a good laugh to a vexing evening trying to put together a complex device or synchronize dissimilar electronic components. In yet other cases, however, the foreign design of the product might belie its use in a domestic context; whether because Mattel's foreign manufacturers painted children's toys with lead-based paint or simply because the American public is not yet ready to check their sick child's temperature using a thermometer with only a Celsius scale. In fact, there is an entire domain in international marketing that specifically looks at how to bring a domestic product to a foreign market. Designing a product from the outset for that market is another matter.

The United States is certainly no foreigner to failed international products. For example, American Motors tried to market its Matador coupe in Puerto Rico; while studies found the name implied virility and excitement, the term means "killer" or "murder" in the local dialect and held problematic connotations for the country's then-hazardous roads (Time 1970). UPS had similar difficulties in both Spain and Germany; their brown vehicles were the color of Spanish hearses while the tag line "what can brown do for you?" was too reminiscent of the Nazi's brown uniforms for Deutschland (Khalifa 2011). Coors, too, made a faux pas when they translated their slogan “turn it loose” into Spanish where it was understood as “suffer from diarrhea” (Khalifa 2011). And yet, all these are simply examples of that first category of international blunder: the product and its design is sound, it is merely the branding or tagline that is problematic.

There are, however, many examples of products which simply do not fit unaltered into a non-domestic context. Pepsodent had such a problem when it tried to sell “tooth whitening” toothpaste in Southeast Asia. The issue is that many locals chew betel nuts in order to blacken their teeth as this is considered more attractive. While dental health is still a valid issue in this part of the world, the entire concept behind a product to whiten teeth had to be reformulated in order to focus on the perceptions and wants of local consumers. Modifying or redesigning a product to match specific cultural needs is more common than people might expect. Although extremely well-received in the United States, Intuit’s famous Quicken accounting software simply would not sell in Germany. The problem was that the program was designed for the way North Americans pay their bills and track expenses. Once redesigned to accommodate a European workflow, the product became a success. It is this context- and need-matching which can make or break a product launch as well as a product line.

Almost nowhere is this a more tangible issue than in the domain of software development. Whereas tangible consumer products are often manufactured in tandem with local marketeers or at least in response to the perception of local needs, the world of consumer applications is still dominated by American companies with little or no experience bringing their products to market in foreign contexts. Whether foreign-born or not, software developers generally design, implement, and test their code in a North American context, trying to solve North American problems, defined by North American needs and wants. Further, because software has become ubiquitous with the rise of personal computers, digital personal assistants, smart phones, tablets, and the like, the proliferation of new applications is skyrocketing, with relatively little attention to the needs, wants, or perceptions of non-domestic markets. The United States overwhelmingly dominates the software industry and as a result it is the needs and perceptions of the American consumer to which the industry caters.

It is true that there has been something of a revolution in the field of international software development over the last twenty years and as a consequence application development has made progressively deeper in-roads into an ever-widening array of languages and cultures around the globe. This process is sometimes smooth and straightforward, such as in Western Europe, but also at times it has been characterized by great difficulty, such as in the Far East. While the lessons learned, especially from Japan (Takezaki 1999, Nakakoji 1994, Hall 1987), have greatly impacted software development, thereby streamlining the process of adapting software for Chinese markets, there are still many aspects which remain unchanged. The underlying needs-assessment and product-design process remains founded on American concepts and while user focus groups may be used, those queried are usually a convenient sample of consumers who match the profile of those doing the research. Almost nowhere does one find software developers reaching out to non-domestic users and making an effort to understand their actual workflow rather than the developers' idealized perception of it.

In the ideal case, the conventional software engineering paradigm of one universal design for all languages and cultures has started to give way to a new, adaptive, and culturally-sensitive model where, theoretically, linguistic and social norms are taken into account during the design phase and incorporated into the final product. There should still be one executable[1], but there should also be a constellation of external, linguistically- and culturally-relevant modules which allow the application to adapt to the expectations and conventions of regional users (Collins 2002, Purvis 2001, Kaneko 1999). In addition to engineering experience, this new paradigm requires cultural and linguistic familiarity with the target audience. It also requires planning for an international design to be a requirement from the outset of the development, rather than an afterthought once the product is nearing completion. In order to be sustainable, adaptable, and effective in the real world without multiple product iterations, software teams need core members on the engineering team who are experts in understanding this international context as well as sensitive to the wide variety of languages and cultures the product will serve. These experts also need to possess cultural capital on and organizational support from the core development team during the length of the engineering cycle in order to effectively incorporate needed design modifications into the final product. In short, this new paradigm deviates from the traditional make-up of product teams, placing a stronger emphasis on one or more members of the development team to act as user advocates for customers with diverse cultural needs and perceptions.

Why Is This Important?

What concern is it of American software companies, especially those in smaller niche markets, to meet the needs of small communities of foreign consumers? What is the return on investment? The answer is simple: over half of all software sales are non-domestic and non-English. Selling to this broader audience multiplies the potential financial return on the investment costs spent on the development work.

Roughly half of all software sales come from non-English-speaking markets (Collins 2002, Kay 1994, Cunningham 1993) but only a few of these markets or their issues are given a somewhat-comparable status to English by core development teams. Usually, the importance of other language-versions are ostensibly stratified by sales of the product in those regions, with English issues trumping all others, followed by Japanese, German, and French. Although there are exceptions, failures in this first tier of languages are generally aggressively resolved, in part because of the cultural expectations of the market; Japanese society is notoriously intolerant of faults while France's Académie française has painful financial consequences for publishers of "Franglais". Commonly, Western European languages come in as the second tier of support, with Chinese and Korean as the third tier, Eastern European languages in the fourth tier, and more exotic languages such as Arabic, Hebrew, Vietnamese, or Thai falling below these others. Notice that this hierarchy has nothing to do with frequency of use around the world, else the order would look something like Chinese, Spanish, English, Hindi, Arabic, etc.

Advances in telecommunications have allowed international trade in general to boom with communication, transportation, and finance playing an ever-growing role in accelerating this trend. Since 1969, the number of multinational firms has more than tripled from 7,000 to over 24,000. These corporations account for one-third of all private-sector assets and have worldwide sales in excess of $6 trillion dollars or more than twice the United States' annual revenue in 2013. The import of goods and services has more than doubled in the last forty years with world trade accounting for 29% of world GDP.

According to DataMonitor, a market researcher, in 2008 the size of the worldwide software industry was US$ 303.8 billion. The United States accounts for 42.6% of the global software market's value which leaves US$ 174.3 billion in globalization sales, a figure larger than the total payments on the US national debt in 2010! And that is doing the current "poor job" in only a handful of countries – estimates vary, but roughly 90% of the software market is in English speaking countries, Japan, and Western Europe. What of non-traditional and under-served markets? The Gulf States of the United Arab Emirates, Qatar, Bahrain, and Kuwait alone represent a larger population base than any Scandinavian country (8.2 million people), read and understand a unified language (Modern Standard Arabic), and have a higher GDP than any nation in the world, including Norway (~$56,000 per person per year vs. $55,000). Yet there is only the most marginal localized software presence in any other these nations. Thus, after overcoming the internationalization design hurdles, developers could bring their product to a population in an untapped, nearly-virgin marketplace with a standard of living and average disposable income on par with any Scandinavian market. The other 350 million potential users of Arabic are "free"; all the development costs would already be covered. Meanwhile, DataMonitor forecasts that in 2013, the global software market will increase in value by 50.5% (Wikipedia 2010).

Neophytes to the field of internationalization will often suggest that it is more important to develop an application for English-speaking markets and then once such a design is proven, redesign it to meet the needs of foreign users. The wisdom and experience of the internationalization community, however, insists time and again that this approach short-sighted. Not only can it cost significantly more to redesign and rebuild a product that is already in use by customers, especially when attempting to support legacy users (Khalifa 2011) but such a path also gives competitors the opportunity to leapfrog into a dominant position by serving the needs of the foreign customer even while offering an otherwise-inferior product (Khalifa 2011). The cost in effort and time to do it right from the outset is small while the potential loss in revenue and market share is huge.

There is another reason why multilingual, multicultural software development is important: social justice. Why should speakers of English and related languages be the only ones to take advantage of the latest software tools, services, and games? Of the ten most widely spoken languages, four are largely unsupported in most software applications while two others usually have full support although this usually also requires elaborate engineering enhancements to achieve. Are the needs of Arabic, Hindi, Bengali, and Punjabi speakers of less import than those of Mandarin or Japanese? It is expected that by 2020, everyone on the planet should have internet access; currently, for many of these people, contemporary applications do not support their languages nor the problems they are looking to solve. A growing trend in the internationalization community, however, is to look at enhancing support for such underrepresented languages. Not only does this democratize the World Wide Web, leveling the field with regard to information access and availability of software tools, but as language is often one of the key components to cultural identity, this may help strengthen linguistic ties, especially across generational gaps, slowing the cultural erosion caused by globalization.

In this report, I introduce the basic terms and concepts of software globalization. This should provide a somewhat idealized image of how world-ready application development is done which I will then contrast with actual practice, as defined by case studies in this process and interviews with informants at various firms, both domestically and abroad. With that road map of contemporary globalization praxis, I will detail the context in which this globalization work is done at Adobe, both in general and in the specific context of my own product team, Photoshop, and the constellation of individuals who are part of the globalization effort. My anthropological project for this report has been to describe and analyze the recent development cycle of Adobe Photoshop CS6 and that team's attempt to enable the product for Middle Eastern and North African (MENA) users, paying particular attention to any changes the inclusion of ethnographic, user-focused outreach methods may have had to the process. I believe these examples will illustrate why culturally- and linguistically-sensitive software application development is important. It is my firm belief that cultural sensitivity and an insider's view are key to building tools that users need and want, a fact that I have tried to blend into every aspect of my work on this project, the CS6 MENA-enablement of Photoshop. The international marketing and industrial design communities are beginning to embrace these concepts as well, as evidenced in both graduate-level MBA as well as globalization management curriculum (Khalifa 2011, Tchernoousko 2011).

With its core competency of viewing and understanding culture in emic terms, anthropology is conspicuously absent in this arena. What little understanding there is of these cultural issues usually comes to developers and designers through the lens of international MBAs and is focused on finding methods to sell an existing, culturally ill-fitting product to consumers rather than holistically understanding such users and incorporating solutions for their needs into the initial design . My hope is that the discipline of applied anthropology, with its goal of applying anthropological theory for the ethical resolution of practical problems, will see a niche for itself in the domain of international product design and play a more pivotal role in future software development projects around the globe.

Software Globalization

Theoretical Process

From the perspective of software engineering textbooks, the process whereby a software application is brought to a non-domestic market is clear-cut, streamlined, and relatively obvious. The program would be designed in such a way that changes can be made relatively easily without affecting performance, thus a modular architecture is advantageous (Purvis 2001). Unfortunately, legacy code lacking such an architectural structure often makes up a large percentage of the code-base, hampering such efforts. As Purvis et al. explain, “The task of internationalizing a software product can be significantly affected by the approach taken by the original developers of the software” (2001:87). We generally only hear about a few success stories amid many marginal attempts. In software development, there is a gap between what is taught in classrooms and what is done in industry. Unfortunately, there is often quite a gap between the two, although even some of the most financially successful examples have room for improvement. “No one does a perfect job,” comments Iouri Tchernoousko, former professor of Globalization Project Management at the Monterey Institute of International Studies, “and that’s really sad because this isn’t rocket science” (conversation with the author, October 17th, 2013). The focus is generally on the marketing and even the packaging, rather than making sure the application meets the culturally-informed needs of the user.

To look at how this all happens more fully, let us now define the concepts and terminology involved in international software engineering, both from the perspective of academic theory as well as contemporary, industrial software practice. The data on which I draw includes my own work experience in the field, interviews with colleagues in the industry, conference presentations, technical manuals on the topic, graduate courses on globalization, and peer-reviewed articles.

We have all heard the term *globalization*, especially as applied to the new colonialism of Big Macs in India, Euro-Disney outside of Paris, and economic exploitation of foreign populations. In the domain of software engineering, however, globalization has a very specific meaning. In this context, globalization is the process whereby software is redesigned and recreated for any foreign market. For developers, the globalization process of designing and implementing a product for multiple languages and cultures follows a specific, idealized path, defined through the experiences and insights of numerous international engineers and project managers and codified in numerous international MBA curricula around the country (Tchernoousko 2011). I will further develop this definition of globalization in software engineering in this chapter.

In 21st Century software development, a basic goal is one executable program for all languages. Coding a single core application to simultaneously support all languages in the basic design is the norm (OpenOffice 2013, Microsoft 2013, Collins 2002, Purvis 2001, Rafii 1995). Borrowing from the terminology of IBM, Microsoft, Apple, and Adobe, in the US-context, *internationalization* is the process of taking a software application designed by English-speaking engineers, in an English-speaking, North American cultural context, that runs on North American hardware and adapting it to support workflows found in other languages, such as Japanese, French, German, etc. (Tchernoousko 2011, Mohr 2006, Yourdon 1994). It is, succinctly, broadening the design parameters to accommodate as many cultural and linguistic expectations as feasible. This would include adapting the program to handle “overt factors” such as date and time formats, currency, character sets, font, reading directionality, keyboards, input methods, and linguistic scripts (Purvis 2001). Generally, the more similar a language is to English, the easier the internationalization process will be. For instance, handling the diacritical markers of Western European languages (å, ê, ì, ó, or ü) is quite a bit easier than those in Turkish or Romanian or Vietnamese (ğ, ı, ş, ă, ồ or ể). The internationalization process has not changed since IBM's initial forays outside the American marketplace in the 1960s and 1970s. Thanks to the work of bodies such as the Unicode Consortium many of these overt factors are well-defined and even a cursory search will give many techniques to streamline this work.

Yet internationalization also needs to handle "covert factors" in the target culture, subtle emic perceptions, such as attitudes associated with certain colors, sounds or images, expectations of user availability due to compulsory prayer, hierarchy, or even differential definitions between words with similar English meanings, such as "temple" and "shrine" (OpenOffice 2013, Collins 2002, Purvis 2001). A technical issue such as converting Japanese kana into kanji has a contextual cultural component which must be resolved[2] and further the social construction of events, such as the meaning, purpose, and norms of a "business meeting", may require culturally-specific designs and solutions[3] (Nakakoji 1994). Software usually attempts to solve some human need, from calculations to communication to organization to simple amusement, but when left unaddressed, covert factors can doom a product design because how these needs are met are not universal. There needs to be a bridge between software design and cultural definitions which bridges the chasm of subtle definitions and emic understandings.

Not all covert factors, however, involve such overarching aspects of the program; many are simply an attempt to avoid being misunderstood or blatantly offensive. For instance, including pictures of animals is generally a bad practice unless the goal is to refer to that specific animal; metaphors are lost across languages and different cultures might understand a given animal in very different ways. A graphic of a dog, for instance, might indicate loyalty, vigilance, a companion, a pet, a farm animal, food, or uncleanness, depending on the audience. Similarly, an icon of a rural American mailbox with the red flag up might seem clear, but this is a socially-packaged image. What does it mean? In most parts of the world, mailboxes appear differently and there is no cultural analog to the pickup flag. Without an explanation, the meaning of the icon can be incomprehensible to foreign consumers (OpenOffice 2013). Other visual or auditory messaging can be patently offensive. The use of body images should be strenuously avoided and there are almost no hand gestures which are not understood as offensive somewhere in the world. While writing this sentence in red catches the reader's attention, is the perception positive or negative? If it is the reader's name in Korea or Japan, red indicates that person is deceased; in such a context, the attention red text elicits is likely to be quite negative. It is not just images and colors which are problematic, sounds too are culturally-defined and informed.

> *While the game show buzzer sound for incorrect answers is well known to people in the US, it is simply an unpleasant cacophonous noise with no meaning to those in other countries. In Japan, making a mistake on your computer can be personally embarrassing; broadcasting that mistake to your coworkers via a buzz or beep may cause shame. This does not boost product sales.* (OpenOffice 2013)

Ideally, product designs should be as language- and culture-agnostic as possible while still meeting user needs (Cunningham 1993). The core application should contain all the cultural variants but include sufficient flexibility so as to access the correct examples via localized versions or user settings.

Rarely is this goal achieved. While either multibyte character sets or Unicode compliance is a prerequisite for most markets outside of North America, Western Europe, and Oceania (Guo 2003, Kaneko 1999), many major applications and most small ones do not attempt this additional internationalization work, often citing that there is "no need" or "we'll do it later" or even "we need proof of concept first." Later might never come and redesigning an existing product can be both costly and problematic. Given that less than half of all software is sold to domestic users, that is the same as saying, "this is a great product, but instead of a small amount of additional effort, I want to prevent sales to half the planet." Although obvious to experienced globalization and marketing professionals, unless spelled out in these terms, the importance of an internationalized design is often lost even on experienced software engineers who then skimp on international design work. As Damian and Moitra point out, there is a "gap between the state of the art and the state of the practice" (2006:18). Even when the case is made to understand and address the needs and wants of foreign users, often this kind of global-minded work is seen as specialized, to be handled only by specialists or trivial work not given due attention. Traditionally, at that point, it is already too late; external teams rarely see software builds before the Beta candidate, the nearly-complete working model, and when they do, it is usually already too late to make more than simplistic changes. Unfortunately, "requirements gathering and analysis need to consider cross-cultural norms right at the outset" in order to do cultural prototyping that meets user needs (Purvis 2001).

Cross-cultural or linguistic issues can exacerbate the gap between core design and internationalized product development. Issues might be vitally important to the audience in the target geography, such as universal translation of all user interface (UI) elements into French for users in France[4] or the reliance on dead-keys to correctly enter important, required characters found in Czech, Polish, and Hungarian. It is the quality engineer's responsibility to think of the target consumer and ensure their needs are addressed in the final product. If the quality engineer, however, is the only one to conceptualize a problem, resolving it becomes quite difficult. An external globalization team may go back to the core engineering team and ask for a redesign of crucial UI elements or feature workflow because problems are not being identified, much less resolved. The product team may see such requests as overly-theoretical or culturally-indulgent unless the problems of the target audience can be explained to them using easy-to-conceptualize analogies with users or markets with which they are more familiar.

For instance, when dealing with a language which does not mark short vowels, the importance of dictionaries and thesauruses is marginal compared to English since these tools function in a radically different manner. Words are not looked up sequentially by character but deduced from context using formulaic models and analogy[5]. The closest quick analogy would be to imagine this whole work was written without a single short vowel – how hard would it then be to distinguish between “vry” for “every” and “vry” for “very” much less offer alternative options for each word? For the reader of these languages, it is context, not spelling which is important and which defines meaning, a level of sophistication that no digital dictionary has ever achieved[6]. If the product design, however, includes a formalized checklist to include foreign-language dictionaries, addressing this need (or lack thereof) falls to someone who must then play a role that is one part political and one part cheerleader to have the requirement modified or dropped entirely. It is usually the quality engineer, especially with regard to non-English products, who performs this job.

Another example of this kind of quality work dealing with covert issues involves translating a problem into an analogous one in another language. Often, explaining an existing issue in terms of one that has already been solved is all that is needed to get engineering and/or product design in agreement to fix it. For instance, if a developer familiar with Japanese complains that supporting Arabic or Hebrew diacritics is not necessary “since they’re not usually used anyway,” then drawing an analogy from *Nihon’go* could be quite helpful. While the Japanese make little use of katakana in everyday life, these are a required character set for foreign and legal words and therefore a crucial element in supporting Japanese type. Diacritics hold a similar place in Arabic and Hebrew writing. No more discussion is required and the conflict is resolved because this concept as well as the cultural and linguistic need associated with it are already recognized in the domain of Japanese software internationalization. This kind of concept translation seems a very logical fit for those with anthropological experience, a discipline with core competencies in both understanding insider perspectives and then explaining them to an outside audience, of translating cultural concepts rather than simply words or phrases.

Once an application has been internationalized, the next step is *localization* into one or more target languages, such as Italian, Ukrainian, Japanese, Arabic, etc. In many ways, especially when the design is already effectively internationalized, this is the easy part. It is important to perform internationalization first so as to leverage target requirements for multiple languages (such as the decimal indicator outside of the US, Japan, and China) while not affecting core components during the actual localization work[7]; the alternative would require numerous costly redesigns near the end of the development cycle for each similar localization issue[8]. Localization includes replacing all user interface elements with those of the target language and adding language-specific tools, such as dictionaries and hyphenation engines, as well as adapting the user interface (UI) to covert cultural norms of these languages' speakers (Purvis 2001). This latter category is rather subtle and can vary considerably with different products, from capitalization paradigms (meaningless outside of Latin and Cyrillic scripts), color usage (in the Far East), directionality and UI mirroring (Arabic and Hebrew), and even supplementary language support (some regions require bi- or tri-lingual language support, due to post-colonial or societal factors[9]).

Globalization is the process of performing both internationalization and then localization together on a product (OpenOffice 2013, Tchernoousko 2011, Wikipedia 2010). Thus, adding the capability to type complex, right-to-left scripts such as Arabic (اللغة العربية) or Japanese Kanji pictograms (such as 日本語) in an English application would be internationalization, as it enhances the core product’s capabilities in a non-English market, while changing the user interface into the Japanese or Arabic language would be localization, and the final result would be a globalized product.

Generally, globalization is a function performed by larger software companies, who each maintain central specialized teams available to assist individual product teams during their design, development, implementation, and localization phases (Aldahleh 2010, Eldawy 2010, Kaplan 2010, Forbes 2007, Mohr 2006). Commonly, such larger firms have entire departments devoted to such work. In a few cases, such as countries with small domestic markets, like Finland, smaller developers will use globalization paradigms throughout the design process (“born global”) and form partnerships, but this is the exception rather than the norm within the English-speaking software world (Kuivalainen 2006, Guo 2003, Kay 1994).

Ideally, this kind of globalization expertise should be available via a specialized, independent internationalization group who can act as a resource to all core application teams as needed during the internationalization of any product into a specific market (Tchernoousko 2011). In the case of a small or single-product firm which cannot financially justify such a dedicated sub-team, vendors could be contracted to serve this same function as outside consultants (IBM 2013, Kuivalainen 2006). Such a situation is not ideal, however, as vendors generally lack the cultural capital to make more than nominal design changes and often are not deeply-enough embedded in the development team to hear about many potential problems until it is already too late for the current product life-cycle. Mockus and Herbslab note that “as requirements change, it is hard for the formal mechanisms of communication, such as specification documents, to react quickly enough" and further "there may be lack of trust and lack of willingness to communicate openly" between teams (2001:182-183).

Based on my personal experiences at Adobe and comments from colleagues working in the same capacity at comparable companies, well-established core application teams tend to be somewhere between uncooperative and resistive to outside developers and as a result are often dismissive of globalization design paradigms, viewing them as needless, intrusive constraints to their initial "elegant" software models. There is tension between "hard-working code warriors who ship product" and "theoretical niche programmers who do not." This, in turn, often results in substandard localized implementations and multiple internationalization cycles because deficiencies in non-English designs are revealed too late in the development process to correct in the current release (Cunningham 1993). This is a recognized problem at Adobe and in an effort to combat it, each product release is graded across a suite of criteria in order to evaluate the software's international fitness. The next release, and therefore opportunity to improve the international design and attract foreign customers, might be anywhere from one to two full years later. The waste in terms of time, money, and other resources is obvious and yet, this outcome is commonly bemoaned-but-accepted in the globalization literature (Collins 2002, Herbsleb 2001, Rafii 1995).

One possible solution, would be to follow the theoretical model and consciously develop a common, collective team for globalization, despite the apparent short-term costs. This would maintain at least one experienced, culturally-sensitive individual within the core product team to liaise with the globalization team and/or act as a cultural and linguistic broker during the design and implementation of various subcomponents. This dynamic of external versus internal team members could ease the tension between insiders and outsider team members while also ensuring the entire process, from birth of design, through implementation and development, to testing and validation, keeps globalization in mind. As explained in Microsoft's Developer Network (MSDN) Step-by-Step Summary for successful globalization, "for a product to be world-ready, the whole design, development, and QA process must be globalized" (2013). Thus, this could be an opportunity to improve the design of international software through a small change in the social organization which creates it.

Organizational Structure

I have worked for Adobe Systems in San Jose, California as an international quality engineer (IQE) for more than 16 years. The inspiration for both this project as well as for earning my masters degree in applied anthropology all stem from my experiences, insights, triumphs, and frustrations doing globalization work for Adobe.

Adobe is the seventh largest public software company in the world, with over $4.4 billion in annual sales (Forbes 2013). Adobe's most frequently downloaded and used software is Acrobat Reader, part of its platform-independent solution for printing and displaying content, both type and graphics. Adobe, however, is better known for multimedia and creativity software, in particular the Creative Suite and the suite's flagship product, Photoshop, which drive over 55% of the company's stock price (Trefis 2013). For more than 13 years, I have been part of the Photoshop team, although I have also worked on-loan with other development teams, such as Premiere, PageMill, and PictureCD, to help them improve their internationalization processes.

The structures, interfaces, and responsibilities of Adobe's internationalization teams have changed considerably since 1997. What follows is a brief historical map for context, with emphasis on the contemporary organization.

In the 1990s, as today, a core team of software engineers works together to design and code an English version of an application based on specifications from both marketing and management with limited feedback from engineering and, these days, user experience[10]. After an initial, rough application is created, this software, or “build”, is iteratively tested, analyzed, and improved. Usually only the basic proof-of-concept will be enabled initially with progressively more features and improvements added. While developers do the coding and superficial testing, it falls to quality engineers to ensure that the resulting application meets the design specifications with minimal unforeseen consequences, particularly errors, known as software bugs. Both software engineers (developers) and quality engineers (QE) are members of the same, core working group, reporting to the same management chain, so both developers and QE work together very closely. Project and program managers help oversee the process and ensure that the resulting application will be commercially-viable, based on market research, tradeshows, and corporate directives.

Historically, all of these employees have been based out of North America and have focused on creating an English language version of the product, although today there are support teams in India and China as well as other locales. In the past, near the end of the development cycle, the internationalization of the product was done by another organization entirely, with quality engineering performed by outside contractors (vendors) and engineering work split between vendors and a shared corporate resource, the internationalization team (see also Cunningham 1993, Rafii 1995, Parvis 2001). Today, in an ideal situation, internationalization is done concurrently with development, but often due to time constraints and lack of familiarity with internationalization, the final product may be buggy or have features removed because they could not be fully supported in other languages. Meanwhile, most of the internationalization development and quality assurance work today is done by teams in India and China. One of the issues with this distributed structure is that the internationalization team has very limited accountability and even access to the core team; questions and miscommunications are common and there is no mechanism to hold the internationalization team to the same quality standards as the core team. As Metiu and Kogut note, rich and intense communication is needed when dealing with complex issues; without a central point of contact, it is not surprising that both tacit and explicit knowledge is lost (2004). In addition, internationalization testers, whether vendors or non-core IQE, sometimes lack a sufficiently deep understanding of how the application is supposed to function while internationalization engineers lack a deep enough understanding of the source code to make meaningful changes.

When I was first hired at Adobe, I worked for the PhotoDeluxe team as a Localization Quality Engineer, a position which no longer exists, but which gave rise to the later IQE role. My manager at that time was a bit of a visionary and rather than depend on an external group to do the bulk of the internationalization work for her product, she decided to build a small group within her larger QE organization to test the application themselves, benefiting in the process not just from stronger accountability but also from having a better understanding of the final product and the nuances of its workflow. As a consequence, since the localization testers were part of the core organization and had access to the nascent product early in the development cycle, we were able to identify and fix many internationalization design deficiencies well before the end of the design phase and as a consequence, could make major changes in support of internationalization for the current product release rather than a future one. These corrections were largely still performed by members of the corporate internationalization team, with the core team only involved for major design changes.

Over the next several years, other development teams at Adobe recognized the advantages of having internal international testers on the core product. I was farmed out on multiple occasions to help other product teams develop their own local internationalization expertise, although the corporate internationalization group, both in terms of testers and developers, remained an important resource. The corporate team, funded by a wide array of products, had sufficient financial resources to hire, either as full-time or as contractors, experts in all the languages we supported at that time[11]. This meant there were always people to consult with about localized versions, even when the individual tester did not understand the language in question. Further, much of the work needed to do internationalization was felt to be too esoteric or unfamiliar for many core developers; having internationalization engineers focus on international enablement frees core developers to do further innovative work on their area of expertise, the core product.

Then in late 2001, the “dot.com” bubble burst and money became tight. Not only did the elaborate shipping parties and shipping bonuses end, but so too did the surplus cash needed to keep multiple internationalization teams employed. Most product teams absorbed some internationalization testers and laid off the rest, turning to short-term, external contractors for testing as well as some development work. The corporate internationalization team survived these layoffs and remained based in North America, but it acted more in an oversight and advisory role, supporting nascent internationalization workers in India and later China. For many developers the international group became irrelevant. While the Photoshop team divested itself of some of its core internationalization workers, it also saw the value in maintaining some level of internal international competency and therefore managed to reserve several dedicated positions for international quality engineers.

Today, the Photoshop team is one of the few product teams with multiple core IQEs and is the only team based in North America with any. Theoretically, every quality engineer on the core team makes a “sanity pass” or basic functionality test over the Japanese, French, and

German versions of the product, the so-called Tier1 languages, which represent the most important software markets. In practice, however, this work is commonly performed by external vendors working for outside international testing firms such as BeyondSoft, SDL, Moravia, LionBridge, and WinSoft who also do some of the routinized engineering work. Much of the engineering effort, however, is still done in-house by dedicated internationalization engineers, predominantly in China. For most teams, these engineers rotate and are a shared resource between products, but Photoshop has managed to maintain a single, dedicated engineering resource based in North America for the last eight years. This is important as there is extensive specific product knowledge needed to do a thorough job of internationalization; by maintaining one point of contact locally over many versions, it has not only simplified the process but allowed the product team to develop a relationship with this internationalization engineer.

I was the only core quality engineer for the MENA work on Photoshop, although another QE was cross-trained to handle activities (testing, meetings, bug triaging, etc.) when I was sick or on vacation[12]. In addition, I did have a team of outside vendors to support me. The number of contractors fluctuated between six and ten over the life of the project, with the Arabic testers based mainly in Tunisia and Syria while the Hebrew testers were based in Israel. Their lead was in Colorado and we had bi-weekly teleconferences to discuss issues. Although I never spoke with the other testers, we wrote each other often via email and in the comments section of bug entries in the bug-tracking database.

Most of the development work for the MENA features fell to one core software engineer on the Photoshop team. He would occasionally interface with members of other corporate teams when dealing with shared components, but largely the work was his responsibility. One exception to this was the text engine, a shared piece of code between Photoshop and

Illustrator[13]. In fact, there were several instances when Photoshop managed to get a feature added to our text engine, allowing the Illustrator team to merely adopt and test the same version to gain the same advantages. To a lesser degree, this also happened with the InDesign team, whose text engine is based on the same one shared by Photoshop and Illustrator, but as you will see later, there were certain limitations.

Additional development work for internationalization issues was performed by Photoshop's dedicated internationalization engineer. This can be seen as the outcome of a long and close association between the internationalization group and the Photoshop group; our internationalization engineer has a very deep understanding of the core application and is quite invested in ensuring a high quality result. This eased the workload on the core developer somewhat and allowed us to probably get a few more features into the final product. There were no other outside developers.

Normally, oversight of the internationalization process falls to a specific project manager who supervises the work for all languages. While this continued to be the case, the Middle Eastern enablement of Photoshop was part of a corporate initiative to penetrate new and emerging markets. As a result, there was a single, additional project manager assigned to oversee all six teams tasked with MENA-enablement. While the roles had some slight overlap, in practice there was no problem. The historic internationalization project manager worried about schedule and coordination of the various efforts involved while the MENA program manager focused on specifying features and triaging bugs encountered, a task which normally falls to a feature designer or user-experience engineer.

A basic organizational chart including all the personnel involved in the project can be found in Appendix One. Note that this group of employees is made up of individuals from at least three different organizations and many functional groups, with relatively little accountability to one another. This is why communication between teams and across functional lines is so important.

The developer, myself, one or both our managers, and one or both project managers would meet on a weekly basis to discuss the feature set we were creating. We frequently had setbacks due to the complexity of this work. Sometimes a new release of the software was worse than the previous one or the vendors may have encountered some unforeseen complication or serious workflow issue. There were many debates about the design of certain features which seemed incomplete or superfluous on further investigation, especially with regard to user needs in-culture. Further, the original list of features to enable was far more than could be accomplished in a single product cycle, leading to quite a bit of wrangling and politicking for one feature or another. The final product contained 17 of the 22 requested features, which in turn is a subset of the features we discussed for this market.

In a typical week, the developer built two or more new releases of the software containing either bug fixes or newly-implemented feature work. I then took a day or more to assess the results, validate the quality of the work, look for any unexpected consequences, detail any addition problems and needs, and report this information via a bug-tracking database. Then, the build were handed off to vendors for more in-depth validation, particularly for specific linguistic and cultural criteria. If any of these remote contractors found problems, they would submit a bug which was routed to me for triaging. If I agreed with the vendor, the bug continued to the developer to be fixed, although it was prioritized with all the other outstanding issues and could still be deferred due to lack of resources. Sometimes, there were a difference of opinion as to how a feature should behave; what one person perceived as a problem could be as the developer or designer intended the workflow. The resulting discussion on what the application's behavior should be began in writing within the bug record in the database, with additional wrangling as needed during our weekly meeting. No outsider to Adobe, however, was a part of the meeting, so it fell to me to represent vendor bugs in this situation.

Contemporary Practice

With the technical terms and general process for software globalization defined, we can now turn to the actual way things are actually done. Despite well thought-out theory, the globalization processes tends to perform poorly in practice, both at Adobe and at other software companies. Often two, three, or even more cycles are needed for some applications to be fully localized into a new target locale, often because underlying internationalization steps fail (Tchernoousko 2011, Mohr 2006, Collins 2002, Kaneko 1999, Takezaki 1999, Rafii 1995, Kay 1994, Cunningham 1993). Each of these attempts at localization costs time, money, and good will (both on the part of developers and on the part of international consumers), all of which tends to be in short supply. In addition to the development and design costs which are incurred not once but two, three, or more times, there is the missed opportunity to attract consumers. Customers may buy one or even two versions of an application, but it is very unlikely they will continue to do so if the software does not meet their needs, especially when there are competing products which might. A badly localized application by its very nature is hard to use and therefore unlikely to meet a customer’s needs. As Rafii and Perkins point out, “The growing importance of overseas markets, the diverse demands of multinational customers, and the narrow profitability window for products with increasingly short life cycles compel software vendors to release internationalized versions within months, if not weeks, of the original” (1995:39). This fact is even more salient today when simultaneous releases have become the industry norm. International customers in the 21st Century have come to expect new software to be released in a dozen or more languages at the same time and each of those versions to be equally functional. Unfortunately, this is rarely the case. The cause of internationalization failures may vary greatly, but most commonly the problem is one of not understanding the customer and their context, and as a result, what the customer is hoping for the software to accomplish (Khalifa 2011, Jantunen 2007, Takezaki 1999, Nakakoji 1994).

For instance, the world's most popular image-editing applications, Photoshop, still has a number of subcomponents which are non-Unicode compliant and have been so for five or more development cycles[14]. These components are thus incapable of being easily rendered into non-Latin-based writing systems, forcing Russian, Chinese, and Greek users to access some of these features via an untranslated user interface. Given the impact of such a problem, the most typical response to such deficiencies, at least at Adobe[15], is to simply pull the substandard component from the localized product, in effect reducing the quality and robustness of the localized version in comparison to the English application. As Iouri Tchernoousko, Senior International Program Manager for Photoshop, explains, "You can't jeopardize the [English] shipping dates without a good reason. Unless it's a major impact, [we] just remove the problem if it can't be fixed in time…we've done this for a number of Japanese features" in both Photoshop and other products (conversation with the author, October 17th, 2013). Meanwhile, in those cases where the flaw is less egregious, such as when only some portions of the UI of the component are limited, the tool is left in the product with the understanding that some capability is better than none.

Due to Adobe's huge emphasis on the lucrative English-speaking market, shipping products with limited non-English support has become a widely-recognized and acceptable trade-off so as not to impact the English shipping schedule. Prior to the enablement effort I detail in this report, for instance, none of the Adobe product line had any native support for the writing systems of the Middle East. Prior to 2005, there was no support for Cyrillic or Greek in these applications either, even though the level of Latin-based typographic support has been quite high. After the final MENA-enabled version of Photoshop shipped, I was put on temporary assignment for six weeks on another product for the iPad. That product was original conceived as English-only due to technical limitations and the final product only had limited support for three foreign languages, French, German, and Japanese. Although non-English offerings are growing, they do not keep pace with the core development work for the domestic market.

When I was first hired at Adobe, I worked with six languages but now I am responsible for 19 in depth and to a limited degree 14 more. While more languages are slowly included in various products, evidence of the emphasis on the English-product is, however, common and not just at Adobe. Although Apple has been producing localized versions of the IOS operating system for more than a decade, it was only with 10.7 aka “Lion,” released in July of 2011 that Arabic was added as a system language[16]. While Microsoft has offered users dictionaries in a diverse variety of regional kinds of English, including United States, United Kingdom, and Australian, English speakers can also employ specialized dictionaries for legal term or medical terms. Logically, it is the size and profitability of the English-speaking market which makes the effort to enable these options financially attractive. For Turkish speakers, however, there is only one dictionary option, which was not available at all until 2000, and which offered no thesaurus option until 2011 (Microsoft 2013)[17]. The logic behind these decisions to expand support for other locales is clearly based on profitability, defined in terms of the return on investment (ROI) for the engineering and development costs required to enhance such support.

In effect, the requirements of the core (English) application take precedence, with only the most crucial internationalization issues having any chance of delaying the release-to-market of the domestic product. “We have to have proof of concept first and then justify return on investment," explains Dave Hackel, Acting Director of Engineering for Photoshop, "Unfortunately, that means that sometimes smaller markets don't get the same attention and resources larger ones do" (conversation with the author, October 16th, 2013). Localization and globalization bugs are thus often deferred due to lack of time, but then not addressed in the following development cycle because "that's the way it was last time and we didn't hear any complaints." At least with regard to my organization, there is always more development work than engineers or time, regardless of what aspect of an application is being considered, so all features and bug-fixes are prioritized based on perceived need. I researched the last five versions of Photoshop, from CS2 to CS6 which shipped over nearly a nine year period, and in each product cycle between 22% and 25% of all bugs addressed[18] were deferred, usually due to lack of time or low return on investment/audience. Meanwhile, each portion of the application has a long list of additional features, our 1-to-N Backlog, slated to be enabled at some future point as time and resources present themself[19]. As Jackie Lincoln-Owyang, Senior Quality Manager for Photoshop and recipient of Adobe's Founders' Award for 2012 explains, "We could take five years and put out a bug-free version, but then go out of business because no one buys the product because it is no longer relevant. There's a balance there" (conversation with the author, October 16th, 2013). New features sell the new version of the product and thus drive the viability of the product as well as Adobe's bottom line. Bug fixes are important, but triaged so effort is not wasted on problems few users will see. And it is here that non-English bugs often "get lost."

Today the core development team is mainly English-speaking and physically located in North America, with satellite teams in India and China. Outside of internal quality assurance and US-based product reviewers (both focused on the English product), the main source of defects come from end-user complaints. These bug reports are sent via a form on the main (English) website or through one of a number of (English-focused) social media websites and thus require a certain degree of English familiarity in order to send. Perhaps for this reason, more than 92% of these reports are in English and/or complain about features in the English application. Less than 9% of all bugs are reported in a language other than English[20].

Bugs in another language are typically ignored by the engineers in question who lack the ability to read the bug report in the first place. If addressed, these bugs are handled by interested parties who then act as advocates on behalf of the non-English users; a somewhat common role for anthropologists and a needed one for nearly all foreign bug resolution. The problem, of course, is these bugs represent only one or two voices amid literally thousands of user complaints, even though the issue might be monumental in the localized product. Since a certain level of English is required to communicate with the team, language -- in this case the inability to communicate clearly in English -- becomes a barrier to communication and a filter on the message the team receives. Meanwhile, those who are able use the English product; it is well-known to most non-English software users that the US version is the least buggy as well as cheapest[21], so those who can use the US version. Similarly, web content, tutorial, advice, and help is predominantly in English as well, so working between language versions does present some trouble which those with a command of English generally avoid (Welcher 2010). This leaves non-English users with less recourse to get help with non-English bugs.

So language creates a filter for accessing these bugs; the development team has a wide, dense network in the form of user feedback which provides information on product quality, but this network is largely overlapping and redundant, lacking structured holes to allow for a broader range of (non-English) information. As Moeran points out, networks connect people and information, but the act of choosing one avenue closes others (Moeran 2005). The density of similar, English-centric information effectively limits or closes the possible network of non-English quality improvement, requiring one or more workers focused on the non-English market and non-English users in order to perceive a more complete picture. This is the role I chose to adopt for my work project and to analyze in this product report.

This is not a role most software teams accommodate and usually outside vendors are given this task. Often, they are the only ones with the time and linguistic skills to pay attention to terminology complaints and product reviews in foreign trade magazines. Four years ago, Photoshop added a new feature named “Lens Correction,” but incorrectly translated the term as “aperture correction.” Despite having at least three vendors and one internal tester working on German[22], this error was not discovered and made it into the final, publically-released German-version of the application. The German IT magazines quickly jumped on the oversight with quite embarrassing articles and exposure, yet no one on the development team nor in the US company was aware of any problem. The German sales and development offices said nothing and the whole issue only came to light because of a personal relationship between members of the US and German teams. Once revealed, the translations were immediately updated, but what would have happened had this informal non-English network not already existed? How long would it have taken for this information about the mistranslation, already vetted by a professional translation consulting agency, to have reached the necessary parties in the American office if only the official corporate channels were used? I have found across a number of product releases that German users tend to be among the most vocal with regard to bugs, yet never did anyone report the translation problem directly even though this is Photoshop's second largest linguistic market[23]. Unfortunately, at the time of this project, there were no official networks or processes in use to capture parallel issues in other language-versions, yet it is quite likely that there could be similar issues in less well-canvassed languages today.

My thought to solve potential pitfalls regarding unknown product deficiencies would be to create an explicit, non-redundant network of Photoshop customers and experts with intimate knowledge of diverse, non-English markets around the world – similar to distribution channels, but which could run information from the target country back to the development team instead. I could then consult with these individuals to find salient information which is non-obvious to a worker physically located in Silicon Valley, but which is recognized in-country. Frequently, Adobe's information networks regarding foreign consumers are very redundant, with only a few individuals corresponding to a particular profile involved and most nodes being connected with each other; these resources tend to offer only one or two perspectives which heavily overlap. As a result, these networks sometimes offer limited insight into problems or myopic viewpoints about the norms of non-English-speaking users. Consequently, we miss bugs and lack a deep enough understanding of customer needs which are different than those of English-speaking North Americans. My hope is that a network of informants such as I wish to create would help compensate for the core team's closed networks, facilitating problem identification and this is one of the interventions I sought to implement for my work on this project.

There is an assumption at Adobe, based on the feedback from English-speaking and some European users, that customers will eagerly voice engineering bugs or design deficiencies. Beyond the often-forgotten language barriers, the truth is that not all humans share such behavioral norms and in many societies discussing another entity's failures is something done only in particular contexts or through trusted, established relationships. It is worth noting that this is particularly important to high-context cultures, such as the Far and Middle East, where some forms of direct communication along these lines are culturally-inhibited; a customer might tell you a feature is "no good," but not delve into the specifics needed to address the situation quite simply because without an established connection between you, such behavior is not appropriate (Nydell 2006, Hall 1987). "The reason is because honorable people in Egypt would never consider showing this type of disrespect [a difference of opinion] to their peers or superiors" particularly in public, although such behavior does happen in private (Wilson 1998, 90). As a consequence, especially when dealing with the development of unique features for users in such countries, having one or more team members with recognized, formal connections with influential and knowledgeable experts in situ can allow the team to gather crucial, otherwise-occluded data and needs assessments. To quote Hall:

> *Despite popular belief to the contrary, the single greatest barrier to business success is the one erected by culture. Each culture has a hidden code of behavior than can rarely be understood without a code breaker.* (Hall 1987, xvii)

The important concept to keep in mind here is that tools are culturally informed. Their use is based on a cultural perspective of how a given tool should function defined by social conventions. "For example, it is important to keep in mind that notions about a computer or microwave are, in turn predicated on cultural beliefs about offices or kitchens, which in turn are informed by beliefs and practices surrounding work, play, place, and so on" (Sunderland and Denny 2007, 50). Without some way to understand this insider cultural code, software developers are blindly creating solutions to resolve problems they do not fully grasp. Getting it right can lead to all sorts of new innovations and products, but getting it wrong can just as easily confound a tool's use.

For example, Apple recently introduced a formatting convention for their user interface that uses capitalization to signify all heading categories. Naturally, a number of other firms wanting to maximize the “modernity” of their UI design did the same. The problem is, such a paradigm is only valid for those languages which have capitalized forms of their letters, so it is meaningful only in Latin- and Cyrillic-based scripts while meaningless in Japanese, Chinese, Korean, Arabic, Hebrew, Thai, Devanagari, etc. Further, however, this capitalization paradigm created serious localization problems for certain cultures, such as in German and Turkish usage, where all caps obscures grammatical meaning or is understood as insultingly direct and commanding. While a similar case can be made for English, the magnitude of the problem is greater in these and other languages. But lack of familiarity with multiple languages limits how seriously this problem is taken; it is the monolingual designers rather than the linguistic and cultural experts who are in a position of authority. Without a strong relationship, the voice of those who perceive the severity of the problem will not be heard or heeded on such a topic. Outside vendors or team members almost never have this kind of power within a product team; only a known insider with cultural capital has much of a change to challenge and change a new, pervasive design paradigm. This, in fact, was a major issue of contention during the CS4 development cycle and only resolved by a united effort[24].

The Photoshop team is very lucky. Because our team composition has historically included a number of core members with deep internationalization backgrounds, not only has it generally been more sensitive to such issues, but the team is recognized at Adobe as a leader in the field and given more leeway by management. The team pioneered externalizing all linguistic elements to have a single executable for all languages as well as set the standard for Eastern European support. Because of the team’s track record, we have a little more freedom in our work and support for our ideas. In this case, it allowed me as an International Quality

Engineer (IQE) more of a voice in the design process for our MENA-enablement work. Working with the MENA project manager, we changed the design of a number of features to better target the needs of our Arabic-speaking customers.

As I will discuss in the following chapters, through my interventions enabling Arabic and Hebrew support in the Creative Suite 6 launch, I took my role to new levels of involvement and interaction, trying to use anthropological concepts to improve the design and development of this milestone release of Photoshop. This included learning the basics of two new languages, exploring large regional variation in expectations and needs in the Middle East and North Africa, gaining insight into the cultural value of typography and calligraphy for Arabic- and Hebrew-speakers, soliciting an additional network of experts to act as key informants for each language, reaching out to and learning from the MENA user community so as to better understand their needs, and pushing for changes in design and even paradigms for implementing digital typography.

Project Context and Professional Skills

The Project Proposal and My Plan to Prepare Myself

In the Spring of 2006, my boss met with me and proposed that our product, Photoshop, might soon expand into the Middle Eastern marketplace. Such a development would naturally require quite a bit of internationalization work, quality assessment, planning, and cultural sensitivity in addition to the obvious localization activities. Beyond the overt tasks of supporting the regional languages, Hebrew and Arabic, which are written right-to-left[25], the Arabic language is written using a complex script, that is a form of writing where the shape of each letter is determined by those letters which precede or follow it, resulting in up to four distinct forms of each character[26]. This in turn leads to two issues; first, can the font and layout mechanism in the application's user interface (UI) sufficiently support both of these writing systems and second, is there sufficient typographic support in the application itself for customers to make use of the regional fonts, most of which were not crafted using Western conventions. Further, there was the question of mirroring or horizontally flipping the user interface so as to conform to local paradigms of starting on the right and ending on the left. Beyond this, there was some awareness of different numbering and date systems, regional dialects, novel diacritical marks, and even multiple linguistic domains (as a legacy of earlier colonial territories). There was no thought given to Arabic- or Hebrew-specific typographic features or requirements. Never had any company product even attempted entry into this geographical area, so the details of the terrain were largely unknown.

During the preceding product cycle in 2004 and 2005, I had been tasked with "Central European[27] enablement" for Photoshop, the greatest single expansion of linguistic support

Adobe had ever attempted at the time. In a single release, we added full internationalization as well as localization for six languages at once[28], including two new writing systems[29]. Although all of the dozen or so core product teams were tasked with the same management directive, as revealed by the internationalization group's independent assessment during the post-mortem analysis of the various product releases, only the Photoshop team achieved its goal. Most other product teams, in fact, failed even effective enablement of a subset of these languages, based on the assessment of the criteria of the internationalization group's Globalization Report Card. During our work, the Photoshop team uncovered a large number of unexpected covert issues, such as keyboard support, dead-keys, and language-based cultural conventions among Russian users. These issues took us by surprise and were relatively unknown to other teams. I and the developer working on Photoshop's CE-enablement were then tasked to present our findings and lessons learned to the other product teams during the 2006 corporate Technology Summit. Management's plan was for us to share our insights in order to aid the other product teams in achieving similar results. Because of this and my previous work with French, German, and other Western European languages, I was viewed as something of an expert in internationalization testing as well as needs analysis. It was with this in mind that my boss tapped me for the MENA project.

If someone had asked me if I would ever attempt to learn either Arabic or Hebrew eight years ago, I would have declined as I have my hands full keeping up with the other nine languages I've already invested time in learning. But, based on my then nine years of hands-on experience in the field of software globalization, when my boss asked for my involvement, my first reaction was to plan to take classes in Arabic, the language I perceived as the most difficult and alien from an English-speaker's perspective and therefore the language most likely to have technical and cultural enablement issues[30].

My rationale was simple: I had just spent a year learning Russian for the very same reason during the CE-enablement work and that effort greatly enhanced my ability to understand the issues and granted insight into the expectations these customers actually had. While not perfect, even this cursory linguistic exposure gave me a certain level of cultural competency, allowing me to better understand user perceptions and needs. Basically, it gave me enough common ground to evaluate and set the priorities of bugs reported by vendor agencies and to take a hard look at relevant feature designs. This is crucial in all software development environments I have ever encountered; for a feature to be added or a non-trivial bug to be fixed, it needs a champion who understands the issues at stake and can articulate them both in terms of user perspective, what anthropologists term emic view, as well as in terms of the bottom-line impact or return on investment (ROI) required by management. As noted earlier, outside vendors often are not privy to engineering or feature development meetings except peripherally and as a result, many bugs or needed enhancements are not realized without buy-in and effort from one or more core members of the development team. This is not due to malice, but rather a combination of ignorance and social organization – the core team members do not know what issues should be major priorities and which are minor or common problems and the vendors usually have very limited communication channels with the actual developers doing the work. By combining a rudimentary knowledge of the language and a passing familiarity with the culture, I managed to capitalize on my social capital with the product team to get more done and avoid many pitfalls and show-stopping bugs. I was able to focus on what was really needed and understand the linguistic and cultural issues of outside vendors while negotiating with local developers and program managers. Language acquisition seemed to be my edge and with this in mind, I chose to learn Arabic.

I had chosen to learn Russian for the CE-effort precisely because of my earlier work in support of French and later also German. As a four-time exchange student (although a native English-speaker) I had gained enough linguistic and cultural insight to excel in international quality assurance work through my deep familiarity with both languages. I found learning a given language as the key to learning that culture.

I am the only long-term, American-born person I know of working in this field, something that seems counter-intuitive since my own mastery of these foreign languages is, of course, inferior to that of a native speaker. Comparing my own work experience and skills to other long-term internationalization workers, in part reflected in the following interviews, I attribute my longevity and success in this field[31] to a combination of being a subject matter expert, having insight to generalize between languages, and the ability to sound and speak in a "comfortable, familiar" manner like other members of the core team – "one of the guys" who understands both software coding as well as how American development teams really get the job done rather than as an outsider and/or foreigner, potentially pushing a language- or culture-specific agenda.

Some globalization workers are only familiar with a single context, usually their native society (as this work is dominated by foreign-born employees), but such individuals do not usually remain in the role as they soon become "one trick ponies" with little standing outside that linguistic context and potentially biased toward the needs of the users they support. While members of a given culture are fantastic resources when globalization work is in progress, without a broader range of skills, it is difficult to justify their continued employment once the globalization process in their target locale is complete. A worker with a broader array of cultural experience to draw on, however, can not only speak to issues from a given, familiar culture, but is more likely to draw correct inferences to novel contexts and even study up on such situations. I think of it a seeing culture in parallax, as humans see objects with binocular vision to gain a better perception of depth. In fact, diversity of cultural and linguistic experience is a common trait I find in most long-term internationalization workers, something I only recognized as I analyzed how this work is done and by whom, as I will detail below. Even before I realized this, however, any time in the past I was given a new geographical assignment, I tried to leverage as much language-acquisition as possible based on what had worked in previous projects. I now realize that this is quite effective for the basics, but only an indirect way to get at the truly needed expertise, that of cultural sensitivity.

Research into this topic reveals that I am not alone in my assessment that language is a powerful initial tool to understanding a population's motivation, values, and sensibilities. According to Dr. Deepa Reedy, words and phrases, "for instance, *saudade* in Brazil (literally, the love that remains; nostalgia, longing), the multiple words used to describe "shame" in Arabic, or *amae* (dependence, respect, and obligation) in Japan—provide intimate insight into the organization of social life, and tell us a great deal about permissible emotional expression, the distinctions of public and private, individual and collective responsibility, authority, and so on. These ideas, in turn, point out what is meaningful in a culture and how, or what must be negotiated for effective design innovation." Reedy is a professor of anthropology at the University of Houston, Clear Lake as well as a writer for the UX Design Newsletter, part of Human Factors International. She points out how ethnography has become the new buzzword of the design community and makes the case for this field "to integrate anthropological approaches with those of business and design, so many fresh opportunities to help transform people's experiences of their worlds" (2013).

While the need to understand a customer's expectation of how a product works might seem an obvious advantage to anthropologists and other social scientists, it remains fairly novel or at best implicit rather than explicit in the field of internationalization engineering as well as software development in general[32]. Designers working with user experience (XD) and some marketing specialists might see the value in this non-technical work, but in my experience, few engineers feel the same. Most developers have really only a limited contact with end-users and therefore a rudimentary understanding of the needs and perceptions of these customers. Not surprisingly, internationalization, a domain with the added variables of alternate linguistic and cultural expectations, is a field where individuals with competency in multiple cultural contexts would be at a distinct advantage, regardless of whether this understanding is a product of linguistic expertise or simply intercultural familiarity and analysis. The latter, however, is largely unknown: even though nearly every long-term internationalization professional has a similar skill package which includes cultural insight, the trait is not explicitly recognized as valuable, as can be seen from the following interviews.

Globalization Personnel Snapshots

Over the course of the last two years, I interviewed a number of software professionals working within the arena of internationalization and localization, both at Adobe and elsewhere in the industry. These individuals represent all the various functional areas involved in the project I describe in this report, such as core engineering, international engineering, quality engineering, product management, and project management. I interviewed the range of positions, from individual contributor to Director of Software Development and Product Localization. All have been doing this work for sixteen years or more and several have taught the subject to graduate students. What follows are a few snap-shots of their experiences, perceptions, and skill sets; I use pseudonyms to obscure their identities.

Olga[33] identifies herself as “Slavic” and when pressed will clarify that she is “ethnically Russian” with roots in the Ukraine. She has advanced degrees in Japanese and Russian translation and in addition speaks (in some rough order of fluency) English, Ukrainian, Spanish, and French. When asked about what makes her particularly effective at her job, Olga answers that because “we service a global audience,” her language skills allow her to “understand the concepts better” and they make it “easier to extrapolate the right thing to do” for these users. Her wide linguistic background gives her insight into a variety of different customer perspectives while also keeping her general enough to avoid being seen as a champion of one or another particular geographic audience by those with whom she works. She feels her skill package is somewhat unique and allows her to move from “language-agnostic abstractions to cultural expectations and localized workflows” – in other words, act as a bridge from an over-arching, engineering perspective to a context-specific, culturally-sensitive, workflow need. In many ways, this sounds like the skill set of an anthropologist working between two distinct cultures, bridging them both to facilitate a meeting of minds.

Bob answers that he is “first generation Chinese” when the question is asked domestically and “American” when asked abroad. He speaks English, Cantonese, Mandarin, and Spanish and feels his particular forté is to “invent, fix problems, and bridge gaps” both in a technical as well as familial context. Bob shared that he “definitely sees [his] background as an asset” in this line of work because “those who are very American, not exposed to other ideas, cannot imagine some solutions.” He discussed the topic further, struggling to succinctly summarize what he was referring to. I offered him a paraphrase which he liked: “Monolingual Americans are often in an intellectual and imaginative straight-jacket.” Bob feels he can “see different worlds and views” and can “see from world to world which allows him to do a better job.” He “stumbled” into his field and personally faced the same crisis as many core developers "when time and resources are limited for a new project, many engineers developing version 1.0 don't want to think about all the other languages, because [they] need the feature to sell," and if they cannot make it work in the US, "then it can't sell." The problem is "re-engineering is painful" and many developers "would rather give up a search for new and novel problems, especially in multilingual" contexts. A large part of his work is simply to "evangelize" and show "there is such a thing" as global-ready design, something Bob did not know until he began working in the field himself. Again, we have the themes of sharing insights between perspectives, solving problems, and making improvements for a wider audience.

Steve sees himself as Indian. His second job out of college introduced him to the world of internationalization and he has been doing this work now for nearly twenty years. Steve speaks Malayalam, Hindi, English, and Japanese, and has lived and worked in various parts of the world. This allows him to "more easily communicate and be understood," especially by those coming from a foreign context with which he is familiar. His language skills give him an "extra comfort level" and "privileged access" when dealing with people with whom he "shares a common language." Such people speak with Steve and think "'he understands our culture'" and as a result, it helps him "connect with others" and facilitates casual encounters with them. His strength, however, is his vast history of all the factors involved, both technical and cultural. This allows Steve to see "all the principles involved, which are not the same as what others know" and when he explains his insight, this can result in an "ah, ha moment." He has "a passion for culture, which helps him and is different." This familiarity with other cultures, however, has been a slow cumulative process, starting with wider Indian culture, then Japanese, followed by Chinese and GB18030 support[34], then Eastern European, and finally MENA. In short, Steve has learned to generalize from the familiar to the foreign in stages, using his familiarity with language and culture to gain insight and smooth over potential pitfalls with others.

I also spoke with Julia, a PhD candidate doing research similar to my own. Julia identifies herself as a “global citizen.” Already knowing her background, I pressed for any other labels and her response was “Latina living in the US” although she quickly added, “I am NAFTA.” I asked what she meant and her answer was that in addition to having lived and worked in Canada, the US, and Mexico, her grandparents were Chinese, German, Mexican, and Spanish, so she considers herself something of a mix. Spanish was her birth language, but she also speaks English, French, German, Italian, and Japanese. This was an important point for her, because when I asked what skill set is most useful to her internationalization work, her answer was “language, technology, history, culture, and geography” and these “show [her] different cultures; not all we produce is applicable everywhere in the world. Most things have to be modified for different cultures.” I asked her why. She pondered my question for a while before answering, “language and culture are very tightly intertwined. We must understand both.” I pressed further, asking why language or culture should be factors at all, and she added “It just is. If you don’t understand the culture, then you can’t see the perspective…language is not the first step for learning culture, but it is a requirement to get it right…there is [a] minimal functionality that’s universal and generic, but to have richer products, that will sell, you must address cultural elements.” Julia went on to explain that everyone in the field of software globalization knows this, but “designers and core engineers only agree when they are reminded. They need a cheerleader or evangelist.” She laughed, her work for the last 16 years has been to “continuously evangelize to the non-believers and infidels! But internationalizationers get it…To be successful today, you must get internationalization to become part of the DNA of the company…making sure everything can be customizable and market aware. It’s just what you do.” We talked further and laughed, comparing my thesis-report with her own dissertation. Her final words were, “culture is the most important part of design of them all. It is my passion. Put that in all caps!” Clearly, understanding how people think is a critical and inherent part of this work, a universal skill for everyone interviewed. But an explicit reason why this is so, beyond “it works,” eludes even doctoral candidates and professors of internationalization.

Nadine was my fifth informant and her mission is to ensure Adobe produces “the best experience possible for international customers, in terms of products, websites, packaging, and documentation.”   When I asked her what skill or trait was most helpful in this regard, her immediate answer was that she “was not born in the US.” She laughed and added that of the whole team she works with, “we’re all foreigners…[we can] put ourselves into the shoes of customers more than core teams do.” I asked why being born in Luxembourg was an advantage and Nadine answered, “Dave, you’re the exception with your language and multicultural background – your whole team is, really – but other teams really need us,” citing the skill packages of several of those coworkers I mention above. Just speaking French, Luxembourgish, German, and English is not enough, Nadine “lives the French OS (operating system) as [her] customers do,” and combined with “spending time on the French forums,” this allows Nadine to “see what French users are experiencing every day, with every product.” She could not do that were she limited to English. She is “still technical” and can “talk to the product teams in a language they understand,” but can also allocate time where they cannot. Like Bob, Nadine never intended to work in this capacity, but she enjoys it and has “developed such an expertise that [she has] a bigger impact on customers than if [she] were a member of a core team.” While she sometimes finds it frustrating that she lacks ownership of a core product, and the ability to clearly influence decisions that such ownership represents, in her role

Nadine can positively “touch more customers” than anywhere else – translating ideas between different languages and between different social organizations within the company.

The detailed inventory of skills among these internationalization workers is salient. Not only do these people come from a variety of companies involved in this work, but our combined globalization experience sums to more than a century. The linguistic and cultural backgrounds, the insight into ways people think and conceptualize products, the role as brokers between two or more ways of viewing a problem are all notably similar. I interviewed additional globalization professionals, but found their responses somewhat repetitive of those presented above. Based on this data saturation, I would venture that these key informants represent the common perspective of internationalization workers, the culture of the globalization professional. As Romney, Batchelder, and Weller point out, when dealing with experts in a domain, we can rely on a small number of good informants with a high level of confidence (1986). Culture is, after all, that common shared knowledge of how a community perceives, conceptualizes, and acts upon data. Experts are most likely to agree on the basics. Common to all of these highly-experienced software professionals are a background in multiple languages and cultural fluidity, even though none of them has any formal education in the social sciences. The reader may note commonalities within these interviews, but even when pressed, the informants generally underplay the importance of their common “soft skills.” The explicit recognition of the importance of finesse and sensitivity to cultural issues simply does not exist in the field; these employees are hired for their technical skills but remain viably in their position because of the cultural insight and viewpoint they each shared.

When I mention my own studies in applied anthropology, the first reaction I usually get from technology workers is a puzzled expression. It is only after I begin to explain how culture impacts the use of the tools we design that the nodding begins. Normally, there is an implicit recognition of those traits which contribute to desired outcomes, such as language experience and time spent in-culture, and not much more, at least explicitly. The ability to tease apart and look at culture, especially with respect to anthropological skills learned through academia, does not generally occur to people in this line of work as valuable. This is true even among educators in the field who are teaching at the graduate level (Tchernoousko 2011). Further, the common core of these skill sets is not something people really think of as important attributes among prospective employees; I have never seen the resume of an anthropologist or sociologist cross my desk though I have lost count of the number of international MBAs and designers. I work with user experience designers with degrees in neuroscience, principal computer scientists versed in astrophysics, and market researchers with masters in psychology, but I have never met a single social scientist on any development team. Despite the common trait of exceptional familiarity with different languages and cultures among internationalization workers, no one seems to hire academically-trained experts in cultural sensitivity or research.

The software globalization literature is also quiet on how to design for culture. Jennifer Tidwell's *Designing Interfaces* is generally considered the standard for software UI design and the author notes that a maxim in the field is to, "Know thy users, for they are not you!" (2011:1) yet there is absolutely no mention of how to do this across cultural boundaries. *Designing International Software* by Microsoft Press is similarly a comparable resource for developers, but in more than a thousand pages, there is not one mention of proactively designing for culture, other than avoiding the well-known and established pitfalls (2003). Guo does ask the question, whether or not localization is a science or a social science (2003) and Nakakoji does talk about the importance of correctly targeting your customers' culture (1994). Jantunen discusses gathering engineering requirements, briefly mentioning context but completely ignoring any mention of culture entirely (2007). Otherwise, authors discuss the importance of paying attention to the norms and expectations of target cultures, but there are no suggestions as to how to do this other than by rehashing problems encountered in the past (OpenOffice 2013, Abufardeh 2009, Collins 2002, Takezaki 1999, Rafii 1995). User feedback is mentioned, but there is no thought on how to collect this data much less how to proactively design for foreign customers. In contrast, however, each step of the engineering work is described in minute details (Kaneko 1999, Takezaki 1999). Most of the writing seems focused on the technical aspects of the work. A comment on one forum post on this topic was, "That's right I18N and L10N are mostly about what developers / programmers should do in order for translation people to be able to work.... add some guidelines about product naming and mix it together and... voila - International product guidelines!" (forum message to the author, Ed V. 2013). There is a little on how a sensitivity to culture from inception can impact feature design work but even less on what skill sets or practical techniques might be used to gain this insight or be responsive to these needs. Through this explicit, structured analysis, I hope to provide a better road map for others working in this domain as well as guide myself in obtaining the specific sets of skills I might require for a given assignment, including an explicit background in applicable anthropological theory and practice.

My Project: For Adobe and For SJSU

My task for Adobe was to bring Photoshop to Arabic- and Hebrew-speaking users. In order to do this, many questions needed to be answered. Who were these potential customers? In what way, if any, were these customers any different from existing user communities? Were the differences merely linguistic or are there cultural factors to account for as well? What workflow problems did these users face? What technical solutions would be required to solve them? What resources were available for guidance and insight? And, of particular importance for my work, how do I answer these and further questions? Ethnography and a limited form of participant observation were my among the anthropological tools I used to help answer these questions and an analysis of this process is part of my academic project.

Ultimately, this project slipped out two full product cycles, at that time, roughly three years. This was due to both contractual deliberations as well as the various product teams dragging their feet. Even more than during the CE-enablement process, development teams were unconvinced of the need to do the MENA-enablement work and, frankly, intimidated to deal with such unfamiliar requirements and languages. When work tentatively began in the fall of 2009, not only had new personnel been hired, but management recognized this conflict and defined some rough scope of the work involved. As one senior manager explained, "we (management) have only defined **what** we want done, **how** you do it us up to you," (emphasis in the original) effectively leaving the details to the individual development teams to resolve. For the internationalization group, this was all virgin territory and there was very little expertise or insight they could offer. All major point product teams, such as Photoshop, InDesign, Illustrator, InCopy, Dreamweaver, and Acrobat, received the same executive mandate: produce a version of their application for the Middle Eastern and North African markets. Unfortunately, if you give six different product teams a task to solve, you are likely to have six very different and probably-incompatible solutions to the problem. This is a serious issue when the applications of Adobe's Creative Suite are supposed to seamlessly work together. To help coordinate this effort, Strategic Growth Markets hired an Egyptian-born project manager to give some substance to the vague requirements and drive the work, but he was new to the company and had very limited experience in the field of typography, Arabic or otherwise. Although very professional, this man, Samir (pseudonym), still had to face the same issues as any newcomer to an experienced product team: how much social capital in the form of political influence could he exercise and how effective would he be at staying abreast of issues and coordinating efforts between multiple teams? Worse, unlike members of the Internationalization Team, Samir did not have a background as a programmer and was at an even greater disadvantage. As someone very interested in this work succeeding, I immediately reached out to Samir and shared with him the history of the previous three years as well as the state of the product. It is from this point that I consider my real work began and from this perspective that this report is written.

Anthropological Fieldwork

Pre-Fieldwork: Learning the Basics

Long before I was officially tasked with helping design, develop, and test an Arabic- and Hebrew-friendly version of Photoshop, I began to do research and prepare myself. My first step, to familiarize myself with the basics of the language, culture, and market place, was to take a year of Arabic language at Santa Clara University. My focus was more on the written word and calligraphy, since my goal was understanding the needs of typographers and graphic artists. I was fortunate to have as my professor one of the most well-known Arabic calligraphers in North America, Dr. Fayeq Oweis. From the outset, I explained to Dr. Oweis my goal and he was particularly helpful in explaining not only hand-written techniques but also the pain-points and workflow bottlenecks digital calligraphers experience when working in Arabic script. Talking with Dr. Oweis, it was clear that there were at least two major solutions needed, a basic one, that is to enable writing simple words like "mother" (أم or "umm") or "happy" (سعيد or "sa`īd") and then there is a more nuanced solution which supports the rich and diverse conventions and orthographic grammar of the Arabic writing system, including contextual ligatures, contextual variants, diacritics, justification spacing or taṭwīls or kashidas, and the implementation of actual Arabic writing rather than what Thomas Milo, founder of DecoType, calls Eurabic (2012, 2011). I will return to each of these topics in detail later, but the result of this first year of study showed me that this is a richly diverse part of the world with much regional variation yet surprisingly overlooked by the Western world and Arabic is intricate and complex, quite a bit more difficult than Russian or Japanese.

Although I did not recognize it at the time, I was, in my own limited and clumsy way, following in the same steps as many ethnographers, such as Hortense Powdermaker and Raphael Patai. What they did explicitly, following recognized anthropological theory and wisdom, I fumbled at blindly and intuitively. The first step in fieldwork is pre-fieldwork research; that is, learning the basics, identifying key cultural concepts, and becoming familiar with local means of communication, cultural forms, and methods of conceptualizing and making shared meaning of the world. What my pre-fieldwork revealed to me was that I was definitely not ready yet to go out into the field!

I continued studying Arabic under Professor Oweis the following year. Due to work-commitments and scheduling conflicts, I only managed to attend about half the classes, but I gained further understanding of the script and its uses, allowing me to intelligently discuss the typographic needs and limitations of Arabic in a digital medium as well as truly grasp the significance of Thomas Milo's annual presentations at the Unicode Conference. This groundwork also gave me the insight to engage font designers at Adobe to enhance our then-current offerings so as to better meet the needs of our Middle Eastern user community. Paradoxically, the MENA project had yet to materialize for Photoshop, but I continued to lay the foundation for such an endeavor to the future, both through my personal understanding of the language and typography as well as by addressing needed technological enhancements which would be required to provide this support to our customers in our future. I even tried to push management to accelerate the MENA time-table after having done some crude demographic research and ROI calculations and research (see Appendix Two). Unfortunately, I had very little success; I simply did not know enough to make the inroads I need. Moreover, I still felt unprepared: although I had decent calligraphic technique and insight, could read and write like a child, and could mumble out some basic sentences and pleasantries, I still recognized that there were vast depths to my ignorance.

After two years of exposure to the Arabic language and script, I had as much familiarity as any non-Arabic speaker in the company[35]. I could explain the uses of all the various forms of a letter, when and why to add diacritical marks, and I had a fair understanding of the general philosophy of Middle Eastern calligraphy. These skills would later be invaluable in my work with feature design, bug triaging, and product assessment. Most importantly, they provided me with a framework to start asking the important questions and begin to really evaluate the answers. What I recognized most was that what I knew was very academic and “bookish”; what I needed was how this theoretical knowledge was actually used by real people and adapted to an array of different lived experiences and expectations across a wide geographical area.

This is where Dr. Oweis’ culture class in the Winter of 2009 came in. This course, taught in English to mostly Muslim and/or ethnically Arabic students, introduced me to the diversity of practice and perception across North Africa, the Middle East, and Southeast Asia. While I had read about the various regions, this brought home some of the differences found in the Maghrib, Egypt, the Mashriq, Persia, Indonesia, and elsewhere. It was culture and local understanding, not religion or language, which were the prime factors. Having fellow students from the various regions proved invaluable learning aids; it is one thing to clinically discuss the manifestations of distinct Islamic philosophies or nationalistic presentations of self and quite another thing to hear them debated by stakeholders you come to know personally. This experience made Arabic culture real to me, in a way similar to what I had experienced in France, Belgium, and Germany as an exchange student there. Only this time I could in no way “pass” and the viewpoints were distinct enough from those I had grown up with to keep me fumbling on the “outside looking in.” For the first time, I began to explicitly look at societal factors to understand behavior. Further independent reading in the Spring led me to the path I am on now, studying anthropology in order to gain greater and more directed insight into the perceptions, understandings, and expectations of relatively unknown populations.

Learning How to Learn

I felt I was almost ready to start asking the right questions, even though I had no background at all in anthropological theory. My qualifications were that I had done something similar for the Eastern European languages and I was my team's subject matter expert. While far outside the normal expectations of feature design or quality engineering work, I thankfully still did not feel completely confident to proceed. The MENA-enablement of Photoshop began to move forward in the fall of 2009, but I felt sure that there were techniques known to ethnographers and anthropologists which would be invaluable to our efforts. I had taken a few introductory classes toward an international MBA which were somewhat insightful, but what I wanted was an emphasis on international not business; cultural understanding and analysis were the skills I wanted to bolster my linguistic competency, not Gantt charts or financial reporting. Recognizing that culture is the primary domain of anthropology, I applied to San Jose State University's masters' program in Applied Anthropology. My goal was to gain better insight into how to learn about a given culture and gain an insider's perspective without having to learn a new language and/or live in situ each time.

At first, there seemed little overlap between classroom and real-world experiences. What I was learning seemed to have limited relevance to the problems I was facing in the office. Because I did not have a solid background in anthropology, I took a year of upper division courses to learn theory. Then, in the Fall of 2010, I entered the graduate program. It was not until the end of my second year, Spring 2011, when I had finally started to really digest all the theory I had been exposed to, that I started to see ways to apply anthropology to international design and customer outreach, albeit in ways that I would not have initially expected. Meanwhile, the work to release a MENA-enabled version of Photoshop was well underway. Ironically, it was the topic of my first anthropology class which proved my point of departure: ethnography.

Probably the most important starting point for any product design work is to get inside the user's head and understand how they will use what you are making. To answer this, you need some idea of what problems the customer is facing and how this tool is going to help address the customers' wants and expectations. "Ethnography is all about getting to know a group of people" (Sunderland and Denny 2007:85). In my context, that of enabling Arabic and Hebrew in a professional computer graphics application, this means having a command of what these users experience today, how this experience is the same or different for MENA users as compared to other users, and what expectations this community has for bridging such gaps. With then more than a dozen years of product experience, I felt I had a good idea of the general case, but with regard to both Arabic and Hebrew users, I lacked insight both with regard of what were their pain points and what they wanted. So my first step was to collect data, to find users and ask them what their experiences with the product were.

Ethnographic Research

After years of work on the various user forums, I have established contacts with a wide variety of users around the globe, including Arabic ones. Digging out these email addresses, I sent messages asking if I could speak with these customers about their experiences using

Arabic in Photoshop, and further, if they had other colleagues who might be willing to talk with me. Unfortunately, it was rather slow going at first, with very few meaningful or insightful exchanges. It was later that I realized I was going about the process wrong; a formal introduction facilitated by a known and respected individual, a cultural gate-keeper, is far more effective. Dr. Oweis provided me with a couple and my project manager others. As time passed and word spread, I gained more contacts. I also followed up with Dr. Oweis for his personal insights, later expanding to other contacts I had made through the Unicode conventions over the years. What I was particularly looking for and which I did manage to achieve was a particularly diverse network with relatively little overlap. That is, few of my contacts had direct associations with or even knew one another and thus their opinions are not the repetition of a small community replicated again and again, but rather a fairly heterogeneous web of unique users. My sample might be something of a convenient snowball, but it turned out to be remarkably stratified.

My informants fell into two broad categories. The first set were users of the conventional, Adobe-designed program who were frustrated in their inability to use Arabic in any aspect of the program, from typography to user interface. These customers wanted to simply be able to type Arabic text into Photoshop layers, just as they could in Word, PowerPoint, CorelDraw, Firefox, Outlook, or nearly any other software application. Unfortunately, without complex and arcane steps, these users were stymied; prior to the inclusion of Photoshop's World Ready text engine, there was simply no native way for the application to render Arabic as text. While this blocked most, it did not stop all users; they created Byzantine workflows to solve our enablement deficiencies. Their requests were vociferous, justifiably-irritated, and relatively simplistic. Instead of opening Word, enabling a virtual Arabic keyboard, typing “ اهلا و سهلا”, taking a screen capture of the Word application with this text, switching to Photoshop, tracing the edges of the text into a path, and then manipulating the path as a shape in a manner similar to how text is handled, these users just wanted to be able to open Photoshop and type Arabic. And given that most applications and both consumer operating systems supported Arabic text, this expectation was reasonable. Unfortunately, until we were able to share a functional application with Arabic enabled, there was relatively little more these users could ask for. These consumers were starving for any solution; they wanted something to sink their teeth into and therefore not very critical of what that solution might taste like.

The second set of nascent Arabic users I found worked with the WinSoft version of the Photoshop application. WinSoft, a French firm based out of Lyon, had years before created region-enabled versions of Photoshop for Russian, Greek, Arabic, and Hebrew users. Unfortunately, not only did these versions use a fairly different code base, but they often had serious bugs in their features as well as were somewhat incompatible with Adobe’s versions. In fact, I first encountered many of my informants struggling to upgrade their product or access features which WinSoft did not enable. As a result, users in this category were often struggling with their software and had experienced limited interaction and support from Adobe, since contractually, WinSoft was obligated to troubleshoot and support WinSoft’s own code. The relationships generally began badly and improved as I was able to offer solutions and/or limited insight. Due to a cost nearly double that of the American version and a limited, buggy feature set, some users had both versions and would switch between them based on project need, effectively leap-frogging two or three releases at a time. These users proved fewer in number, but more savvy and nuanced in terms of wants and expectations. For them it was not enough to simply type “أهلاً و سهلاً”, they also wanted to add and adjust the diacritical marks of the vowels, control the justification of words in a line, and make use of alternative, contextual forms of certain glyphs[36]. When it came time to design and build on a feature set for Adobe's enablement of Arabic, the insight these users offered proved particularly valuable.

In addition to these key informants, I collected data about Arabic typography from less dynamic sources, such as local-based Persian weekly publications in Mountain View, Arabic-oriented websites such as Al Jazeera and even Wikipedia, news reports from the MENA-world such as Mosaic, attending relevant talks, and asking questions of experts at the Unicode conference as well as on-campus presentations, and reading up on Arabic and Muslim identity. In short, these sources form a sort of ethnographic record of Arabic culture with a particular focus on typography[37].

Of course, like any good research, this work led to new questions. I recognized that while I had spent a fair amount of time learning about the needs of Arabic users, I had done no work on Hebrew ones. Further, the distinctions between Muslim, Arab ethnicity as identity and Arabic as writing system are blurred in most cases; given vague corporate requirements, just how large of a swath of the world would we be expected to support and how granular would our sensitivity be to local variations? Finally, while all of my informants used Arabic script, they all also spoke English. Previous product experience made it clear that non-English speaking Arabic users would have their own unique set of expectations and desire, but how could I with my limited command of Arabic gain insight into their perspective?

Determining the Target Audiences

I had chosen to focus my studies on Arabic because it is widely recognized as more alien and complex for English speakers than Hebrew, making Arabic more difficult to learn and, logically, more problematic to support. Both are written right-to-left with bidirectional numbers and while Hebrew has a handful of letters with contextual forms[38], all letters in Arabic have at least two distinct forms, usually four, and they are all written using a complex cursive system similar to Latin ligatures unlike Hebrew's simple blockish format. In addition, most Americans have at least a passing familiarity with Judaism; while only roughly 1.7% of the US population self-identifies as Jewish (Jewish Virtual Library 2013), Hanukkah cards, kosher food, and references to menorahs and dreidels are relatively easy to come by. Although Muslims in America make up more than .8% of the population (PewsResearch 2011), it is much more difficult to find Ramadan cards, halal food, prayer mats or hijab accessories. I have multiple family members and even more friends who incorporate some sense of being Jewish into their identity, but outside of those people I have met since 2006, I cannot think of one who is Muslim. This was significant for me because while I could fairly easily find information about Jews and Judaism, but analogous information about Muslims was difficult for me to acquire. At the same time Arabic speakers number between 5% and 23% of the world population[39] while Hebrew speakers, ostensibly the population of Israel, but potentially all Jews on the planet, number roughly .2% (Silverman 2012).

As part of my academic coursework for my masters' degree as well as preparation for this project, I took two classes to familiarize myself with Jewish history and the Hebrew language. The history course gave context to issues of Jewish, Hebrew, and Israeli identity, both today and over the last five thousand years. More interesting and enlightening was my class on Jewish mysticism, folklore, and beliefs. Beyond learning simply the forms and pronunciation of the letters, we learned their cultural associations and religious significance. The two classes also gave me a more complete picture of the meaning of time, calendars, holidays, celebrations, food, clothing, and events from a Jewish perspective. Of particular note for my work was the role of Hebrew diacritics, analogous to Arabic ones, but more frequently used and arguably more important when writing, particularly unusual or foreign words[40]. This coursework allowed me to do further research into Hebrew typography, including special characters, expectations, and even musical notation for religious singers. Without the background gleaned from these classes, however, I doubt I would have even been aware of such concepts, much less how to ask about their implementations and cultural constructions. I will return to these particular concepts when I discuss the implementation phase of MENA-enablement.

It was not until near the end of design phase of the project that our project manager addressed the scope of the world of Arabic script. We would be supporting: Arabic-speaking only. This leaves out some 60 to 110 million Persian (Farsi) speakers (Windfuhr 2009) as well as 61 to 104 million Urdu speakers (Ethnologue 2013, New World Encyclopedia 2013). While disappointed that we could not officially support these two groups who easily number twice that of the world Italian-speaking population, easily equal the world German-speaking population, and potentially rival the world Russian- or Portuguese-speaking population, I did push for as much inclusion of their typographic requirements as possible. Not only was it the right thing to do, it was also part of the original proposal in 2007. With regard to font support and numeric characters, I was quite successful, although with regard to text layout and typography, I failed, as detailed below.

The question of how to address the needs of Arabic-speaking users versus those who also had a command of English was rendered moot. Although the original plan had been to ship a localized version of Photoshop in Arabic and Hebrew, this requirement was almost immediately dropped once work began in earnest in the Winter of 2010. The project manager determined through a series of business trips across the region as well as through meetings with local experts and even governments that not only was there no local demand for an Arabic user interface, but that regional graphic designers felt that such a change would actually make their work harder and/or create a less prestigious perception of the product. In country, command of English was linked to a greater sense of modernity and technical sophistication. As a result, graphic artists actually preferred to work in English. This led to our first conflict as the Photoshop product team butted heads with its new project manager.

It is at this point, I think, that from an anthropological perspective, my initial fieldwork ended and my hands on involvement in the project really started.

Conventional Wisdom versus Local Conventions

Choosing Cultural Expectations over Globalization Paradigm Standards

When the project manager dropped the requirement for a localized, that is translated, Arabic and Hebrew product, he faced a mutiny. All of the core Photoshop team members working on the MENA effort, with a combined globalization background of nearly 70 years of hands-on experience doing this work in over 20 languages, immediately challenged the decision and even the credentials of someone who would take such a stance. In software engineering, conventional wisdom says that selling an English product in a foreign locale is either a statement of disinterest in the foreign marketplace or a statement of technical inability on the part of the developers. As Adobe's MENA project for the Creative Suite was envisioned as a tour de force for the Middle East and North Africa with Photoshop as the project's cornerstone, neither alternative was accurate or acceptable. To meet both market and corporate expectations, the product team felt that Photoshop needed to be fully translated into Arabic and Hebrew.

In every meeting, this issue came up and was hotly contested. As the team which produces the company's flagship product, the Photoshop developers are used to getting their way or at least "working the system" favorably. As with previously-mentioned XD capitalization paradigm, the Photoshop team was ready to put up a fight to do what they felt was the most professional and market-embracing thing, namely, fully-localized Photoshop into Arabic and Hebrew. Not only would this effort mean quite a bit more work, but it was also in support of what earlier vendors had done for the same markets and it was also part of the original Software Requirement Specification (SRS) as late as January of 2010.

The project manager was young and, at least initially, viewed as even more of an outsider than members of the globalization team. Extremely personable and determined, he was fairly well-received but with regard to product translation, his views were disregarded as naïve or uninformed. But he claimed to have spoken to many users in this regard and was quite confident about his position. Further, as a native speaker who had met with a wide variety of customers in the region, it would be logical to think he had a better command of expectations than the team.

Thinking back to the concept of structured networks I learned my first year in the graduate program, I began to wonder how in touch with local MENA conventions we actually were on the development team. I reached out to my independent network of Arabic-speaking contacts, asking their opinion on the importance of a localized version of Photoshop. The answers surprised me a little at first, until I thought about some of the cultural attitudes I had learned in Dr. Oweis's classes.

To the majority of Muslims, Arabic is the language of Allah, God, the Divine. It is an article of faith for most Muslims that the miracle of Islam is how an illiterate merchant could create such a well-structured and eloquent masterpiece of Classical Arabic literature; this is the signature of the divinely-inspired nature of the Qur'an. For many Muslims, all concepts, indeed all words, appropriate to the human experience are found in this holy book. Even newer human creations which are endemic to modern life usually find their linguistic roots in the Qur'an with minor linguistic adaptation. For instance, the word for "airplane" (طائرة or ṭā'irah) is formed from the word for "bird" (طائر or ṭā'ir, found in Surah 6:38) with a feminine ending added. Arabic is a remarkably logical and coherent language with a very formulaic structure, as if it was created in toto rather than having evolved from another language through human usage and adaptation[41]. A student can learn a single Arabic root and suddenly be able to grasp the meaning of 10, 30, even 50 other words of all types, in a manner quite more straightforward and rational than the system of Latin-, Greek- or Old English-roots we have in English. And the Arabic structures are both quite common and relevant to nearly every native word in the language. Adding a number of foreign words corrupts this linguistic purity and some Arabic speakers even see it as offensive[42]. Whether you transliterate “Gaussian blur” or simply translate the concept, it is likely to confuse the native speaker, possibly somewhat jarringly[43]. As one user on the prerelease forums put in, “translating many of these words would make them harder to understand than leaving them in English.”

There was also a second reason why a translated product would be more problematic for Arabic-speaking user: tutorials and help files. Photoshop is a complex application, designed for customers with a degree in photography, graphic design, fine arts, or other highly-technical background. Not all users have this training and it is quite common for Photoshop users to find help and support through a wide variety of sources, with the technological ones, especially the Internet, being most frequently used. On the web, users can find detailed instructions if not complete, step-by-step videos guiding users through nearly any Photoshop task or workflow. While there are a few such guides in other languages, such as Japanese, German, or Russian, these are the exception rather than the rule. In particular, the world community for photographic and digital arts overwhelmingly uses English as their *lingua franca*, so much so that Russian-speaking users have fairly uniform standardized, user-created, third-party English-language-based workflows, keyboard layouts, and expectations[44]. With English used for more than 55% of all website content and Arabic less than 1% as of August of 2013 (W3Techs 2013), it would not be surprising to learn that the richest and most detailed web content is in English rather than any other language[45]. For an Arabic-speaking user, there simply are very few technical resources available without resorting to the English ones.

A translated user interface for Photoshop would make using such online resources more difficult. It is notoriously vexatious, even for those of us who work with multiple language-versions of the product on a regular basis, to recall the exact translations of each menu item, filter, and process. As one coworker and former professor at the Monterey Institute of International Studies regularly quips, "give eight translators a unique technical term and you'll have at least nine different opinions of what it should be." Since many of these items are sorted into language-specific, alphabetized lists, anyone without a particularly powerful command of both the tutorial language as well as the product language is liable to quickly become confused trying to apply the English steps to a localized product. This is even a long-standing problem with Help content. Even though Adobe offers all of user content in all supported languages, seldom is there a parallel for user-created content, such as that found on independent websites or even Adobe Labs, what we refer to as the Photoshop ecosystem. Quite a lot of peer-to-peer help and inspiration is simply lost in translation.

With this understanding as well as the feedback from the actual, contemporary Arabic-speaking graphic arts community, I changed my position and supported the project manager's recommendation. Further, I tried to use cultural examples from previous shared work or experiences to help convince other members of my team to do the same. I drew parallels between the attitudes of Arabic-speaking graphic artists with those of Indian and Japanese users[46]. I tried to clearly demonstrate the difference between linguistic and engineering expectations of both consumers and the globalization community when compared with the cultural and experiential expectations our Arabic users shared with me. This led to discussions and comments both in public and in private, because there were misgivings about following the project manager's recommendations. On the whole, the Photoshop team has been the model of internationalization practices at Adobe and generally tries to live up to the globalization ideal of EJAL -- English is Just Another Language. To willingly chose *not* to embody the best practices of the globalization community is inconsistant with the team's values and practices. It would be like asking an anthropologist *not* to give credence to the lived experience of the people she or he is studying; serious evidence and explanation would be required. In some ways, it did take something of a leap of faith for some coworkers to accept the concept that an English-language version of the product would be better received in-country, more easily understood with few linguistic mistranslations or misunderstandings. Localizing an application entails considerable work and Adobe's development team has never attempted the additional engineering effort to flip (also known as mirror) the user interface of its applications to support the flow of text, images, and concepts in a right-to-left context. With estimates that less than 5% of all Arabic-users had any desire to use a product that required such a manpower-intensive effort, it made further sense to drop the earlier requirement for a translated product.

Turning a Liability into a New Feature

The problem remained, however, that Arabic- and Hebrew-speaking users would need some form of localized Help files to navigate the application. This requirement had existed since the first official MENA project presentation in 2007. As already mentioned, Photoshop is a very difficult and technical application requiring detailed feature explanations. While a given non-English-speaking user might have sufficient command of English to follow the steps to perform a specific task, understanding the nuances of color space management or generating 3D models is another[47]. Seeing both sides of the issue, I tried to offer a solution that met everyone's needs as well as solve a long-standing support problem. I recommended that we design a new interface for our Help application which would allow users to seamlessly shift from one language version to the other while still viewing the same topic. MENA users could then use Help to understand the technical nature of a problem or workflow in Arabic or Hebrew and then shift back to English (or French[48]) to follow the step-by-step process in their non-localized application.

While shifting between languages for a text browser with screenshots and images may seem like a fairly easy engineering task, it had never been done before and my previous request had met stiff resistance. I had worked out an informal solution that worked in roughly 90% of the cases, but before it could be rolled out to the public, the solution would need 100% coverage. The problem was cost; at the time we officially supported in excess of six hundred pages of Help in two dozen languages for Photoshop alone. Further, there are 11 applications in the Creative Suite Master Collection alone, in addition to many other programs Adobe offers. The sheer volume of work involved made the issue a non-starter. Working with the MENA program manager, however, I was able to get the change request accepted and built into the Help for the entire Creative Suite, as the quandary presented by Arabic and Hebrew alone justified the return on investment (ROI) for the work. That the solution also enhanced the workflows, tutorial, and help needs of users in the other 24 languages Photoshop supported was a benefit, allowing, for instance, the author of the authoritative German-language book on color management to simply find the terms he needed in German, change a single control, and note the English equivalents without confusion (Diessner 2006).

That was one of the key outcomes of this project: by working together and sharing both problems and insights, we managed to solve a number of highly contentious issues across multiple product teams and working groups. Not all had such a profound affect for ostensibly unrelated users and Photoshop CS6 finally shipped with several MENA-related bugs and many unresolved, MENA-related features, but the general quality of the program was superior to anything that had come before as it was an integrated whole rather than an amalgam of applets, plug-ins, and last-minute fixes. Further, in most cases, solutions between different Adobe products also shared a very similar layout and workflow, thanks in no small part to the diligent work of the MENA Project Manager. Photoshop's MENA team in particular really tried to deeply understand user needs and offer a high-quality technical solution, even if it meant bending the established "rules."

Breaking Western Traditions in Support of Eastern Norms

Another noteworthy example of this kind of paradigm change which we incorporated into the final product was our update to the *kashidas* algorithm which inserts automated *taṭwīls* into Arabic text. To clarify these terms, we have to discuss some of the details of how Arabic is written and the linguistic and social conventions which surround its use. In a nutshell, a taṭwīl is a character inserted into an Arabic word to make it longer, much as English users use a space key to add distance between two words. Kashida is an (unfortunate) Turkish term now used in digital typography to note when the software adds virtual taṭwīls to text in order to enhance paragraph justification. This terminology is relatively new and quite specialized; educated Arabic speakers will know what a taṭwīl is but only a handful of non-Turkish digital typographers will have even heard of kashidas. Needless to say, very few Arabic customers made use of kashidas in earlier versions of the software as this feature was difficult to understand, required a number of obscure and seemingly superfluous settings to be just right in order to function, and used a specialized, unfamiliar foreign word as the feature name.

Arabic writing is a complex script written in a cursive manner and used not just for the Arabic language, but also for Persian (Farsi), Urdu, Malay, Nubian, Dari, etc. The shape of each letter in each word is dependent upon typographic grammar and logic, such that words are written so that most or even all letters in a word form a single flow or “skeleton” of text. In fact, Arabic typography insists that all words must be made up of at least two letters whenever possible, such that single syllable adjectives, prepositions, or modifiers are often attached to the words they modify, such as بالعربية (bil-`arabīyyah or “in Arabic”) made up of ب (bi or “in” or “with” or “using”) and العربية (al-`arabīyyah or “the Arabic”) or معي (ma`ī or “with me”) made up of مع (ma’ or “with”) and ي (ī or the indicator for the first personal singular possessive). In the following Levantine sentence, each independent word, using the Arabic definition, is marked with a different color: كيف حالك اليوم (kīf ḥālak il-yōm or “how are you today”)[49]. Thus far, Arabic writing is largely analogous with English or French cursive. But the similarity ends when long runs of text are rendered into formal, fully-justified paragraphs.

This paragraph is written using the Brush Script Std font and full paragraph justification. As you can see, the individual letters which make up each word are linked, such that words break into discrete chunks of meaning separated by white spaces. With the paragraph fully justified, that is, with no ragged edge on either side, the word processor adds blank or white spaces between the words to evenly fill the unused space of a given line to the length of a full line of text, thereby aligning both the left and right margins with the paragraph edges. This is known as “rivers of white” since, depending on the unique content of the text and the dynamics of the font chosen, a given line of text using this format can have wide or *narrow swaths of white space added to achieve the desired effect. But according to the esthetics of Arabic typography, this layout is either ugly or wrong.*

Arabic typography demands a certain sense of balance and geometry. There are at least four and as many as a dozen distinct scripts for writing Arabic, including Kufic, Naskh, Thuluth, Nasta'liq, Riq'a, Diwani, and more. Each is governed by a very strict set of geometric rules concerning the height and width of each character, how much space is allowed between letters, how far or near diacritical marks should appear to a word's skeleton, etc. These conventions are then extended between words as well, such that only a specific amount of script-dependent space is considered appropriate between individual words. The end result is much like the left justified text in this paragraph; each word has the equivalent of roughly a single letter between them. Well-written Arabic, however, is also much more demanding with regard to being fully justified, that is, in the vast majority of cases, a given string of text begins at one margin and continues all the way to the edge of the other margin, just as in the paragraph, above. All newspapers and magazines as well as books and pamphlets are fully justified. Anything less has a somewhat less professional connotation. You may wonder, given these constraints, how is text so routinely justified if extraneous line space is not distributed between words?

To fully justify text, Arabic adds spaces to the length of individual letters in words. This can be done by elongating certain letters[50] or by manually adding a lengthening bar, what is known as a taṭwīl (تطويل), to certain letters in a run of text, again following strict typographic geometric grammar. Not all letters or combinations can take a taṭwīl, while others commonly incorporate them. For instance the connection between the letters in في (fī or "in") or لا (lā or "no") are never disturbed while the sīn (س) and yā (ي) are commonly elongated, as can be seen in the following bit of text[51].

مرحباً يا صديقي! كيف حالك اليوم؟ عندي سؤال حول عائلتك. وكم أخ وأخت عندك؟ هم جميعاً في البيت؟	مرحباً يـا صديقي! كيف حالـك اليـوم؟ عنـدي سـؤال حـول عائلتـك. وكـم أخ وأخـت عنـدك؟ هـم جميعـاً في البيـت؟	مرحبـاً يـا صديقـي! كيـف حالـك اليـوم؟ عنـدي سـؤال حـول عائلتـك. وكـم أخ وأخـت عنـدك؟ هـم جميعـاً في ا لبيـت ؟	مرحبـــاً يـــا صديقـــي! كيـــف حالـــك اليـــوم؟ عنـــدي ســـؤال حـــول عائلتـــك. وكـــم أخ وأخـــت عنـــدك؟ هـــم جميعـــاً في البيـــت؟

In the example, above, the left-most paragraph has no taṭwīls or black space added and therefore to achieve full-justification, the fourth and fifth lines have quite a lot of added white space. Each paragraph further to the right has progressively more taṭwīls added, resulting in less added white space. This text, however, is actually employing the kashida feature of Photoshop, which automatically adds programmatic taṭwīls to the text based on an elaborate placement algorithm, rather than requiring hand-worked user intervention. While this may seem trivial to someone unfamiliar with Arabic, it saves considerable time for Arabic typographers, allowing them to add "rivers of black" to their words and avoid "rivers of white" with only a few clicks of their mouse. While the algorithm still has a few bugs which need to be resolved[52], it is an improvement over earlier attempts to solve the same problem and other than a few minor issues follows the rules of native typography. Not only are individual taṭwīls better divided into segments appropriate for the letters in context, but as you can see from the example above, this feature also attempts to balance the lines of text vertically also. You may notice that the first two examples are one line shorter, even though they all contain the same text. In the later examples, the words have been spread out over an additional line so as to balance out how many taṭwīls a single line gains; because of the extra line, all six lines of text contain roughly the same amount of added black space, in keeping with the geometric conventions of Arabic calligraphy.

The introduction of this kind of Arabic-specific typographic functionality was rather revolutionary because the conventions of European and East Asian typography dictate that this specific kind of text reflow -- wrapping to new lines in order to accommodate paragraph justification spacing -- should never happen. For a typesetter in Finland, Italy, Russia, Greece, Japan, or China, this “feature” is actually a “bug.” It breaks the conventions. That said, it is the culmination of multiple iterations attempting to solve this problem, each tested and vetted by a number of Arabic typographers, including newspaper and magazine publishers. What began with observation and limited ethnographic research with Arabic calligraphers was crystalized into codified changes to how kashidas, automatically inserted taṭwīls, were added to words. Further, we streamlined the user interface so activating the feature was easy and did not require a number of harmonized controls; in a fully-justified Arabic paragraph, kashidas are now inserted by default, much as happens with blank space in Western languages. This new version was then given to a variety of users to evaluate, with numerous iterations to address suggestions and complaints. They approved the end result as it generally captures the esthetic and geometry of their language in a straightforward manner, rather than forcing Arabic writing to conform to foreign notions of how digital typography should work.

This particular feature is worthy of particular attention. Not only did it require having native graphic artists share their calligraphic techniques, insight, and expertise into digitally-rendered Arabic script, but it also came from collaboration with another team. Almost all of the actual prototyping and engineering work for this feature was actually done by the MENA engineer for the InDesign team in Seattle. The Project Manager and I had been discussing the problem with her over lunch one day. She seized upon it and created a number of prototypes which we tested for design flaws and bugs, both internally and externally with Arabic prerelease users. Once the InDesign engineer had a solution that solved most of the problems, working with the Photoshop MENA engineer, she managed to incorporate this kashida feature not just into InDesign, but also Photoshop and Illustrator. Even though the font group and other typographic interests at the company were convinced reflowing text to accommodate kashidas/taṭwīls was a violation of typographic best practices, all three teams and user communities ultimately benefitted from the collaboration. It helps that it is hard to argue with a working model that has the stamp of approval from several luminaries in the field of Arabic calligraphy…

Although probably the most unusual, an entire feature created by an external team and dropped almost completely unchanged into Photoshop, this was far from the only example of cross-team synergy and cooperation. For most of the feature work, the Photoshop team took the lead both in terms of culturally-sensitive feature design as well as engineering, but the InDesign team, working with a more complex and nuanced type engine[53] also pioneered other enhancements. The automatic recoloring of diacritical marks is a noteworthy example.

A Missed Opportunity for Excellence

In routine text, Arabic does not include characters to indicate short vowels; they are implied through context and mentally supplied by the reader, much to the consternation of novice readers, public speakers[54], and software engineers. For instance, the common greeting مرحبا (marḥaba or "howdy") is made up of the symbols for m + r + ḥ + b + long A, so that the sound and position of the short vowels must be already known to the reader and added mentally. Theoretically, based on the skeleton, it could be pronounced murihbaa or mirHubaa or any number of other combinations. It is pattern recognition and context that allow a reader of Arabic to render these symbols into intelligent sounds. While a reader of English, German, or

Russian can encounter a completely novel word and have a good chance to pronounce it correctly, this is far less likely in Arabic. As already mentioned, the process to automatically determine and insert the correct vowel programmatically was so difficult that after several years of development, Google abandoned their work on such an effort, the Tashkeel project.

While short vowels are normally not written, they are added in religious texts and children's literature in order to help with pronunciation, meaning, and clarity. Similarly, these marks are sometimes used in newspapers to avoid ambiguity, such as the gender of an individual addressed or the voice of a given verb. The question أين بيتك (Ayna baytuka/i or "Where is your house") cannot be correctly pronounced without knowing if the speaker is addressing a male or a female. In such cases, where important, a vowel mark is added. The vowels are indicated using diacritical marks, somewhat analogous to how an English speaker perceives certain letters in French, German, Spanish, or Italian, such as à, ç, é, ü, ñ, and ó. The problem is, the vowel symbols are quite small and it is rather easy to confuse certain diacritical marks with the various dots which distinguish different letters in a text's skeleton. In fact, at various points in the history of the language these diacritics were written using a different color ink (Milo 2012), precisely to make recognition easier.

Today Arabic speakers from different regions have different sensibilities about colored vowel markers. In Egypt, generally only a single color is used for both the skeleton and the diacritics, but in other regions or contexts, coloring the diacritics red or some other color is often preferred. For instance, the calligraphic instructional videos for the second edition of the Al-Kitaab series, the most commonly used college-level Arabic text book for English-speakers, all use black ink for the skeleton of text and red for diacritics. Dr. Oweis once explained that this makes it easier for those new to the language, especially children and those who use Arabic only as a liturgical language, to discern the difference between the vowels and non-connected portions of a word's skeleton[55]. Likewise the usage of colored diacritical marks is desired in Turkey as well (personal discussion with the imam of İsa Bey Mosque in Seljuk [Ephesus]). An example of how this looks can be seen in the example, below. In the string on the left, diacritics are not clearly differentiated from the dots and other marks which define consonants and long vowels. On the right, the string appears less cluttered, with required marks all in blue and the optional ones in red.

أَينَ بَيتُكِ vs أَينَ بَيتُكِ

Thinking of Arabic typography, then, a potentially useful feature would be a method to automatically recolor all diacritical marks in a run of text, just as the skeleton can be recolored. I came up with this feature early in my studies with Dr. Oweis and it was part of the SRS for Photoshop from a very early date. Unfortunately, due to resource constraints the capability to do this automatically was de-emphasized. Part of the problem was a perceived lack of need; the idea of a feature that automatically finds and recolors vowel markers has no value for English speakers; while rather labor-intensive, it is not something used in our language and it looks pretty odd and even unprofessional when we see the results. Moreover, wider experience with other languages, whether French, German, Hungarian, Russian, Romanian, Turkish, or Greek support the English-based paradigm that this is a minor, unneeded enhancement; it would look odd and even wrong to a German, French, or Turkish speaker to see ç, ö, or ğ instead of ç, ö, or ğ. In short, non-Arabic speakers do not do this and do not see the value in having it done. As a result, without a strong advocate to explain the linguistic and cultural need, the enhancement will never be made. I tried to fill this role and share with the team why this feature was useful.

This feature had been “languishing” for some time; no one was convinced of its importance. I was able to get some time put into solving it, but the engineering manager was not entirely convinced. Moreover, neither he nor the engineer coding the solution really understood the problem or why it needed to be solved. So, I took Dr Oweis’ *Pocket Guide to Arabic Script*, scanned and blew up the page on all diacritical characters in Arabic. I then highlighted the eleven symbols that need this treatment, giving the engineer a reference for which code-points need to be intercepted.

In order for both the developer and his manager to understand the nature of the problem, I transliterated their names into Arabic, intentionally mispronouncing them due to missing vowels. I went through a couple examples of this, using humorous interpretations to drive the problem home. At this point both understood the linguistic need for diacritics, but not the cultural advantage of changing their color. I then shared a Persian newspaper I happened to have handy, demonstrating how infrequently diacritics are needed in contemporary text. When added, depending on the font, they make the lines of Arabic type appear cluttered and sometimes more difficult to read (أَينَ بَيتُكِ vs أين بيتك). Coloring the diacritics is a way to both draw attention to them while also making them easier to mentally filter out (for experienced readers). Running through an example with a doubled diacritical, such as used when writing a double consonant with a vowel, further demonstrated the real-world example of usage and why a single-glyph solution would not work as these diacritics can appear in two-part clusters as well as not at all.

Unfortunately, there were many requirements for the MENA effort and too little time or resources. Each product team only had a single core engineer dedicated to the task. In addition, focus was placed on the requirements of Egyptian speakers as one third of all native Arabic speakers live in Egypt. In the case of recoloring diacritics, this meant the feature’s priority was reduced as in this part of the MENA world, this capability has less value. When time is short and a feature only appeals to a limited audience, it does make sense to cut it from the schedule. Although our Project Manager was sympathetic to the idea, after a few abortive tries, this feature was cut from Photoshop's MENA goals for CS6.

The InDesign engineer was more receptive to the idea. While her feature list was equally as long, she did work to have this capability added to her product. Unfortunately, low-level architectural elements hampered the work and rendered an integrated feature in the application unfeasible without direct work on the type engine. As the type engine is a large and complex component, requiring a large undertaking both in terms of developmental engineering and quality assurance validation, this limitation effectively killed the feature, at least until such time that the engine is updated. The InDesign team were thus similarly unable to get this feature into their product, but they did manage to create an external script which allows simple, global changes to diacritics. While not ideal, it is better than the tedious process of manually changing each glyph by hand. Photoshop tried to use this same methodology, but, unfortunately, our type engine is sufficiently different so that the script failed. That said, at least some users, those using InDesign, did benefit from the design recommendation and this feature is in the Photoshop backlog for future MENA enhancements.

Even in failure, however, we did manage to make some progress. Although users cannot globally change diacritical coloring independent of text in Photoshop or Illustrator, users can make these changes manually. While tedious, I know of no other programs than the Creative Suite applications which even allow this possibility, which is why all the examples in this report which are recolored or not in-line with the text are actually images of Arabic text rendered using Photoshop.

Diacritical position, that is the user-defined placement of these vowel and grammar markers, is a related and far more successful example of MENA feature work. In earlier versions of the software, there was some support for this feature, which gives typographers the option to space diacritics higher or lower from the skeleton of Arabic text. Unfortunately, as originally envisioned, all marks were moved higher or lower. This is a problem, however, as vowel and grammar marks can be both below and above a given run of text. By shifting them all up, those on the bottom can wind up overlapping the text's skeleton. Similarly, by shifting them all down, those above can have the same problem. My suggested design change, therefore, was to have the diacritics spaced farther or closer to the base-line of the text's skeleton, rather than simply higher or lower. An example of this can be seen on the left in the graphic, below. Here, the vowel marks above the text are all higher and clearly separated from the actual letters while the lone vowel mark below is lower and also distinct. Prior to this change, users would have to set the majority of diacritics and then adjust the lower ones, as there are far fewer of them, by hand so as to not overlap. Now, one click and the work is done.

أَنا أُحِبُكَ يا حَبيبي vs أَنا أُحِبُكَ يا حَبيبي

Another problem earlier implementations of diacritical positioning had were vague placement definitions which could be counter-intuitive at times. The categories were None, Loose, Normal, Tight, and Open Type. Theoretically, the middle three settings are supposedly ordered by height over the line, from highest to lowest. Unfortunately, for a large number of fonts, shifting between say, Loose and Tight, sometimes resulted in the diacritical mark moving up away from the text's skeleton, the opposite shift expected. Further, None is supposed to put all diacritics at the theoretical highest point for the font in question, but seldom performed this way with respect to actual fonts which contain Arabic glyphs. In short, the results were not predictable resulting in a sub-standard use experience. In fact, of those users I surveyed, only one even used this feature with any regularity, the rest complaining that diacritical positioning was unreliable and too difficult to figure out. One or two users had made a concerted effort to figure out the logic involved, but abandoned it when the feature behaved differently with different fonts. Yet the linguistic and even artistic need this feature targets remain valid today.

In order to alleviate this problem, we changed the design of diacritical positioning to use numbers instead of words. Using a numerical scale abated most of the ambiguity involved with positioning; a shift of 100 was twice as much as a shift of 50. We also redesigned the feature so that the shift direction was not up or down but rather away from or closer to the baseline of text. This was, positive values moved vowel marks both above and below the line further away from the text while negative number shifted the diacritics closer to the text's skeleton. The resulting feature is logical and consistent, giving users more predictability when setting the placement of their short vowel and grammar markers. To ensure that the new design was easier to use and met consumer expectations, we presented it to a number of publishing houses in the Middle East. We did not receive a single negative comment. Further, we asked Arabic-speaking prerelease users to evaluate the new design, to which they responded favorably.

These are each examples of ethnography in product design. The first step is to do basic research, to learn something about the group involved, so as to be familiar with their unique needs and expectations as well as to be able to understand the basic interactions which will be observed. Then, researchers make detailed examinations of user experiences, noting both what works and what does not, and when possible, asking users what their expectations for the feature are. Next comes the analysis of the data collected, searching for patterns. These patterns can help indicate workflows which are particularly effective, needed, or troublesome. Armed with this data, the researchers can attempt a new, hopefully superior design, innovations of what come before, which can then be iteratively tested with the community in question again.

What makes this particularly interesting work, however, is the community of users the ethnography is targeting. Not only is there a linguistic barrier vis-à-vis most consumers, but they hail from a novel culture from the perspective of the development team, with needs, wants, and expectations which are non-normative when compared to other, more familiar communities. Without this concerted effort at researching these users, their viewpoints would be largely inaccessible to English-speaking, North American-based development teams, resulting in less effective and compelling software tools for Middle Eastern consumers.

Diversity of Arabic Experience

Linguistic Uniformity

Much like North American, European, or East Asian users, Middle Eastern and North African users come in a dazzling and diverse array of unique desires and viewpoints. While engineers and marketeers alike might be more comfortable with a monolithic, easily-identified, and easily-specified user community with discrete needs and expectations, this is a point where anthropology can inform both product design and marketing. As a discipline, anthropology is both more cognizant and comfortable with the idea of many over-lapping cultural identities, even in a single individual. Identity, whether due to language, religion, or society, is not one-size-fits-all and thus consumer tools, especially software applications which are usually available broad regions, need to take this diversity into account.

This is an issue of particular importance when dealing with Arabic-speaking users. While there are a handful of languages which regularly transcend national or regional borders, such as Arabic, English, Spanish, or French, this is not the norm. Most languages are spoken by distinct communities with mutually agreed upon standards which usually conform to distinct, often territorial, boundaries. For instance, although Turkish and Romanian might be occasionally spoken abroad, there is no debate among native speakers that these two languages are found in specific countries and the norms of these national dialects represent are the norms for the language as a whole. Similarly, while German is found in Switzerland and Austria as well as Germany, generally, the version of *Hoch Deutsch* represented in the Duden is usually considered “the” official German just as the norms for numeric formats and measuring scales are the norms for the German-speaking community, regardless of location[56]. French is spoken by very diverse populations around the globe, but the French language is rigorously monitored and defended by the state-sponsored, Parisian-based Académie française such that the dialect of Paris is generally considered “proper” French, used in language-acquisition classes worldwide, mass media broadcasts, and any sort of print media. Along with the language, the conventions of Paris are also normative in the French language, thus the formatting of dates and numbers or measuring scales in Quebec might be different, but generally the norms of continental French are expected and observed, particularly when the audience is wider than local consumers. In the Spanish- and English-speaking worlds, there are multiple, competing linguistic *axis mundi*; for English, for instance, the United States and England form the two major, politically- and culturally-powerful hubs for linguistic and cultural conventions while Australian and India supply two minor, somewhat-regional alternatives. But the Arabic world is far more fragmented, breaking into at least four large regions due to historical, social, cultural, and linguistic reasons.

To accommodate the diversity found in English, it is common for software products to come in multiple versions. Usually there is at least a US and a UK option, with others also possible, depending on the target audience. Sometimes the user interface is altered to account for regional spelling variations, such as color versus colour. In either case, however, it is almost certain that the application will have separate dictionaries for spell checking American English versus British English, sometimes with other versions included as well. Similarly, cultural conventions for formatting and dating will vary between these versions so as to conform to the audiences in the country indicated. English is not alone in this kind of geographical cultural adaptation; Spanish and Portuguese similarly often break into Iberian and New World varieties, at least. Arabic too has this requirement; software catering to the Arabic-speaking world needs to allow for quite a bit of regional diversity, appropriate to a language which is spoken over such a wide geographical area and used as a liturgical language.

The Arabic-speaking world can be roughly broken into the following regional chunks: the Maghrib or West, Egypt and the region immediately around it, the Mashriq or East which can be thought of as Greater Syria, and the oil-wealthy Gulf States. In addition, other regions can include Greater Persia, sub-Saharan Africa, Southwest Asia, and Southeast Asia as well as even Europe and North America[57]. For Muslims, the Qur'an can only be truly understood in Arabic and it is expected all followers of Islam have some level of ability in this language, regardless of ethnic background or geographical home. This again leads to the aforementioned stacking of different kinds of group membership; to the outsider, the Arabic world might appear monolithic and undifferentiated, but is as valid as saying all Christians, all English-speakers, or all North American residents have a single identity. The regional chunking across the Arabic-speaking world is important to keep in mind as it helps give some substance to different conventions found across the region.

Eastern Conventions

English speakers refer to the characters we use for our digits (1,2,3,4,5,6,7,8,9,0) as Arabic numerals, but only some Arabs use these symbols to represent numbers. The western region of the Middle East, from roughly Morocco to Libya, does use the same digits as we do to represent their numbers. But what Arabs refer to as Hindi digits or Indic digits (١,٢,٣,٤,٥,٦,٧,٨,٩,٠) are commonly used in the eastern portion of the Middle East, known as the Mashriq, and roughly corresponding to Jordan, Syria, Lebanon, and Iraq, as can be seen in these license plates and signs from the region (see Appendix Three). Although Palestine and

Kuwait are usually considered part of the Mashriq, when it comes to representing numbers, both nations almost exclusively use English digits, demonstrating how regional variations can be highly nuanced and contextual. In Egypt and the Gulf States, both Arabic and Hindi digits are commonly used, depending on context; Western numbers are often used with Latin text and in all things high-tech. From my time in Egypt, those regions which were particularly touristic tended to use Western numbers, but cafes and restaurants usually displayed prices only in Hindi digits and the same was true for most local road signs and addresses. The money of many of these nations uses both sets of symbols. Meanwhile, Persian- and Urdu-speaking regions, such as Iran, Afghanistan, and Pakistan, use another, similar set of digits called Farsi or Perso-Arabic or Eastern Indic digits (۱,۲,۳,۴,۵,۶,۷,۸,۹,۰) for their numbers[58]. Clearly, there are several ways to represent numbers in the Middle East and the choice of which numeric notation systems to use is based on location, context, and target audience.

Obviously, given the prevalence of such diversity in numeric representation with highly subjective usage patterns, support for all three sets of symbols would be valuable to graphic artists as well as those creating text-based web mockups or simply creating imagery designed to appeal to diverse Middle Eastern audiences. Such a feature was part of the earliest draft of the Product Requirements Document (PRD), which lists all the expected capabilities of the final application. Since this earliest draft, however, what we internally called Digit Type elicited quite a lot resistance from developers. Japanese and Chinese also have additional glyphs for numbers, but never has there been thought to enable these; supporting not just one but two such sets of numeric glyphs for MENA users seemed overly indulgent and a waste of resources.

Often, especially when schedules are tight or there is a lot of work to be done, feature development can become politicized. It is very normal for management or a designer from user experience to create a list of two dozen features to be implemented, but then go through that list and prioritize each item. As work progresses, developers will usually push back based on how much work remains and how serious the existing bugs are. Sometimes a given feature will be cut because the amount of work needed is beyond the scope of what is available. Usually, however, those features which are not perceived as important are the first to go, and many times anything which is non-critical winds up being cut[59].

Although Digit Type was ranked as “highly important” and listed as the fifth requirement in the PRD, the developer perceived it as a frivolous request. The way I solved this conflict was to actually go on the web and search for quotidian examples of numeric representations across the Middle East. With the help of my vendors in both Tunisia and Syria, I managed to collect a large number of images of currency, vehicle license plates, and even road signs in order to demonstrate the ubiquity of these different sets of glyphs in the region. I then gathered analogous images from the Far East, demonstrating that while there are region symbols in that part of the world as well, they are not frequently used. This logic helped smooth over the objections and ensure Digit Type did not get cut from the final product.

Going further, however, I wanted to clarify the problem we had had internally with all the different names for the various Digit Types. Routinely, I would refer to 1, 2, 3 as such when I sited Arabic or Western digits and similarly started adding ١, ٢, ٣ when I was discussing Hindi or Farsi digits. But this did not fully clarify the problem as both sets of symbols share the first three glyphs; the differences are found with how the numbers 4, 5, and 6 are represented, either as ٤, ٥, ٦ or as ۴, ۵, ۶. In the earlier drafts of the PRD, the request was to label the control which allowed users to switch between numeric sets simply as Arabic, Hindi, and Farsi. Thinking of how hard it was to keep conversations and emails straight internally, I wanted to ensure what each setting would mean to end users, especially given that the product would be translated into 23 languages. So, I requested a change, namely the inclusion of all ten digits within the label, so as to give a clear and unambiguous example of what each digit type entailed. This way, there could be no confusion.

My request to change the feature led to two problems. First, it required a fairly large amount of space to include all ten digits plus the term label in a single dropdown menu. Secondly, it required the inclusion of non-English, non-ASCII values to the specific non-Western numbers to be included in the core (English) application. Each of these created further friction for the feature.

The first problem was solved by a compromise. It is a rule of thumb in globalization that user interface designs need to allow for additional room in translation. How much is a question of the target languages supported, but for Western European languages, 50% is considered fairly normal, given the length of German and number of tiny words French, Spanish, and Italian often require. When supporting Russian and Ukrainian, however, strings can stretch to roughly double length, given both the size of the individual letters in the Cyrillic alphabet as well as the length of words[60]. In the final product, the dropdown list for Digit Type has the name of the set of numbers plus the first five digits in that language, since that is enough characters to make the difference clear while also allowing enough room for all languages to include this in the space provided.

The second problem required some creative thinking. Due to a number of technical limitations, Unicode characters outside of the first 255 ASCII characters are not found in the core application. In order to include the first five digits for Hindi and Farsi digits in the dropdown list, we would have to break this rule, which could have unexpected consequences and could cause havoc for the compiler. The proposed solution was to externalize these strings, just as is done for all other languages, but to apply this process to the English version of the product as well. Normally, the English version of the application is treated as the baseline;

other languages are based on the English plus additional modifications, usually through externalized files. Using an externalized file for English was merely a departure from this model; the only reasons not to do with it were historical not technical. Ultimately, the change was made using the logic that English is Just Another Language (EJAL), an example of best practices in international software design.

Digit Type was included in the final version of the product, allowing users to easily and seamlessly switch the numbers used in their files from one to the another of these numeric systems. And while the translation of these terms proved rather difficult as many languages do not make sufficient distinctions, the inclusion of the first five digits as a sample solves any possible confusion.

While we were able to solve this problem, usage and displaying of these same digits within the structure of the application, that is within fields and controls, proved less straightforward. On Windows, users can freely switch between any of the three numeric systems using the settings under Control Panel. On Mac, however, there is no such analog. As a consequence, users of Hindi digits in Iraq could use ١,٢,٣ in a text layer, but if they tried to set the font size to ١٢ pt (12 point) or the opacity to ٧٥% (75%), the application would not recognize the symbol as a number and throw an error. From a workflow perspective, this is a serious issue and something which I am still pursuing to have fixed in the next version of the product. Working with the type developer, we did manage to identify and fix nearly all instances of this problem within that context, which is logically the most likely place a user will encounter such issues and where switching to another keyboard layout is most problematic. But a universal solution required working on too many different aspects of the code and has been deferred. While I have received complaints from only three users, I believe, based on the same linguistic argument demonstrated earlier, that there are probably many more people experiencing the bug but lacking in the technical and linguistic skills to communicate their displeasure to Adobe. This, unfortunately, is one of the areas where the final product did not live up to our normal levels of quality.

Western Conventions

Another example of regional differences is found in the countries of the Maghrib, namely Morocco, Algeria, Tunisia, and Libya as well as Mauritania and the Western Sahara. It was France rather than England which was the colonial superpower and occupying nation. As a result, these countries have a stronger bond with a francophone identity and use conventions of the French language in lieu of English ones. While not an official national language in any of these nations, French is widely used in government as well as taught in schools and for many of these countries French is an obligatory language to study, with subjects such as the sciences, business, and technology usually exclusively taught in French. Software runs on *l'ordinateur* rather than *the computer* in this part of the world, with all the nuances that entails.

The computer keyboards found in the Maghrib display French letters as well as Arabic ones, just as those in Egypt are marked with both English and Arabic characters. While this might sound like splitting hairs as the alphabets of English and French largely overlap, the problem is the layout of keys on the keyboard pointedly do not match between these two languages. In comparison to English's QWERTY keyboard, the French AZERTY keyboard is one of the most distinct Latin-based layouts in use, with a large number of keys as well as numbers in different locations.

The question, of course, is why would English or French or any other language be present on an Arabic keyboard? The answer is simple: compatibility. Only in the last few years have non-Latin scripts begun to be acceptable for use in Internet URLs or as filenames, and even then they are often not supported. Further, as already mentioned, software translated into Arabic is often hard to find and untranslated software requires an untranslated hardware interface to use it. Due to American dominance in the field of computer science, most regional authorities around the world adopted English as the Latin-based alternate for their language. Today, keyboards from Tokyo to Beijing to Moscow to Cairo to Jerusalem all display English letters alongside the foreign ones. But because the countries of the Maghrib followed French examples of modernization, their keyboards show French instead of English letters with the corresponding layout.

The problem here is it has become standard practice for software developers to assume an English keyboard layout when dealing with writing systems which are not based on the Latin script. Such a model persists across all of the Far East today, where both shortcut keys and accelerator key combinations are denoted in all portions of the user interface using the Japanese, Chinese, or Korean command with the English letter accelerator and/or shortcut (see Appendix Four). Users can, of course, write in Japanese or Chinese or Korean, but because of the complexity of these languages and the nature of the input method, the application only accepts English-based keystrokes for commands[61]. Although a handful of applications allow for non-Latin keyboard shortcuts, this is the exception rather than the norm; nearly all applications use English letters and layouts even today for Cyrillic (Russian, Ukrainian, Bulgarian, etc.), Greek, Hebrew, and Arabic versions, as seen from the screenshots from a variety of applications and operating systems (see Appendix Four). But user from the Maghrib, users whose keyboards display a French rather than English layout, problems arise when they try to do simple commands. Trying to follow this standard model, these users get the wrong results. The universal shortcut for “Select All” is to press the keys Control and A (CNTRL+A). But on a keyboard from Tunisia or Algeria or Morocco, since the layout of keys is based on the French AZERTY scheme, the A key is in the location of the the Q key in English. Thus, the software receives the command Control and Q (CNTRL+Q) which is Quit, leading to disastrous results! Similarly, Undo's CNTRL+Z yields Close because the French Z is in the location of the English W. For these users, nearly one quarter of all their letter keys are in different places, resulting in quite a bit of frustration even for typing, much less trying to control their software.

One response would be to allow these users to use a French layout for command, shortcuts, and the like. Unfortunately, this is a departure from the traditional model, notably different than the way things are handled for Russian, Japanese, Chinese, and Greek users. As a result, Engineering balked at the request. Although a serious problem for the workflow of Maghribi consumers, no one was willing to make the code changes needed to allow some language other than English to be used as a default keyboard layout. We have received seven user-reported bugs on this topic and I am currently using those to request this change be made in a future version of the application, but as this is a problem localized to four or five countries with low economic resources, other bugs and feature work has taken precedence and probably will continue to do so for the foreseeable future.

This bug should not take more than a few days to solve, but reflects what happens when developers cannot empathize with the needs of the customers they are trying to serve. If the problem is perceived at all – this case, one developer simply kept saying users should use the "correct" keyboard, which smacks of insensitivity to the norms of this region – it is still not important enough to fix. But, as this could be solved with a simple case statement and a few days testing, had I done a better job expressing how big a problem this bug was for customers, one reported quite early during our first phase of testing, I am confident it could have been solved quickly and easily without fuss. The next chapter deals with another such failure, this time on my part, to get a clear enough insider understanding.

Hebrew Support

When I was first given the MENA assignment, I focused on the most difficult topic, learning Arabic. I knew I did not have time for mastery or even deep knowledge, but I did figure that this experience would help me work through the issues with the easier example, in this case Hebrew. However, I did not realize that the cultural component involved in modern Hebrew would take years to fully grasp. How the language is used today spans a very broad spectrum of spiritual beliefs, cultural statements, and social ideologies that it is nearly as daunting a task as scratching the surface of Arabic typography.

Before any coding or testing began, I bought *Hebrew in Ten Minutes a Day*[62]. On a flight back East, I read the first several chapters and thought that although the alphabet presents a hurdle to access for an English speaker, the way short vowels are omitted and the general structure of the language makes for close comparisons with Arabic. Naively, I did not see any aspect of the Hebrew language that was not covered by supporting the features found in the Arabic language. Further, I spoke with Dr. Mira Amiras, one of my former professors at SJSU, who had done field research in Morocco with extensive experience across the entire Middle East. At the same time, Professor Amiras is also the daughter of a rabbi and is extremely well-versed in Judaism, so much so that she teaches a class on the topic. This made Dr. Amiras a valuable resource, able to look at both sides of the Hebrew/Arabic world in critical and insightful ways. I learned, in fact, that much of modern Hebrew's vowel placement was reconstructed using Arabic as a model, so as to reduce the influence of Yiddish and return to a "pure" root[63].

About the time of our final Product Requirements Document (PRD) in August of 2011, I began to fill in my knowledge of Hebrew language, history, and culture. I wanted to enroll in a course on contemporary Israel, but none were available. So, since I could not get access to the present, my first step was to learn about the shared past of the Middle East through a semester long course on the *History of the Ancient Near East* in which I learned about the roots of the Jewish faith and the cultures which shaped and influenced Jewish thought to this day. These proved valuable, giving me some perspective and understanding for the other course I took that same semester.

I had considered trying to take a semester of college Hebrew, but I decided against it. Instead, I enrolled in her *Jewish Mysticism* course, which the students affectionately renamed Kabbalah for Dummies. Instead of merely learning the sounds and shapes of letters[64], I also came to understand each character's symbolic meaning as well as the extremely rich and nuanced understanding some Jews have about the origin and important spiritual nature of Hebrew. Modern linguistics notwithstanding, it would be disrespectful to disregard Jewish sensitivity to the divinely-inspired aspects of Hebrew, particularly its written form. Further, this course taught me about Jewish culture, albeit from a liberal, spiritual perspective. The green sheet included several dozen movies; students were to choose two or three to watch in order to gain some glimpse of what being Jewish means to Jews and how language, particularly the spiritual understanding of the origin and nature of Hebrew letters, influences this identity. I watched every film on the list which I could find, giving me some small level of appreciation for this topic. I also now had a point of departure to delving more deeply into Hebrew identity as a vehicle to understand how customers might use Hebrew script in Photoshop.

I went on to attend a series of seminars taught by professors of Hebrew from Stanford University as well as watch Israeli and Jewish-themed film screenings and generally get a grasp on the contemporary Israeli perspective. In many ways, this was my first attempt to learn a culture without also trying to learn the language. I gained some valuable insights, but in retrospect, I do not know if I just learned too little too late, or if the rich, historical nature of Hebrew usage over time was too complex to grasp without also learning the language. While I now feel that it is possible to learn the basics of a culture independent of language, I am uncertain if it is the best method.

Nearly all features which we enabled for Arabic support were also relevant to Hebrew users as well. Unfortunately, in the final release of the software, we did not allow for a Hebrew implementation of several of these features. In some cases, this was due to technical limitations but in others, we simply did not realize that the feature was relevant to both communities of users.

The most obvious MENA feature which is enabled for Arabic users but disabled for Hebrew users is diacritical positioning. This is the previously-detailed capability for a user to manipulate the horizontal and vertical placement of these linguistic markers, known as *niqqud*. Such a feature did exist in earlier versions of the project, but the results were unpredictable and buggy. Faced with a choice between supporting Arabic and Hebrew, we jettisoned the latter and deferred the feature for this release. Unfortunately, this turned out to be a much-desired feature among users and as I write this, it is one of the top five MENA issues from our backlog of enhancements to be considered for a future release. But by the time we developed a method to elicit rich feedback from Hebrew-speaking users, it was already too late. Although the lack of this capability was a common complaint, we were out of time and the shipping schedule was inflexible.

The fact is that diacritics in Hebrew do not just indicate short vowel sounds but also differentiate certain consonants, such as between “s” and “sh” or between “p” and “f,” which only underscores their importance. Because placement of these marks varies between fonts and is often incorrect when multiple marks are included, this actually becomes an important issue for Hebrew users. In Arabic, diacritical marks rarely have layout problems unless many are used; Arabic users mainly reposition their diacritics based on aesthetics. Hebrew users, on the other hand, actually need this capability to unambiguously define meaning in their language. As a result, I would argue that this feature is more salient and needed by Hebrew-speakers than Arabic-speakers, but our implementation was the contrary.

Beyond the implementation of grammatical and linguistic diacritical positioning for Hebrew users, near the end of the design-phase of the project, I learned that the language has a second set of diacritics as well. These *cantillation* marks are used to inform how the word is chanted or sung[65]. Their use is not common outside religious contexts, but their use begs another point: even more so than in Arabic, in Hebrew, especially when cantillation marks are used, there is a desire to color marks differently than text so as to avoid confusion. In this case, text would be black, regular vowel and gemination markers in red (analogous to the use of red for Arabic diacritics in some contexts), and cantillation markers in blue. The return on investment (ROI) on such work is liable to be small and I understand why this work not done, but it was not even considered and there was no knowledge that these marks existed. While needs analysis is technically part of my Adobe job position, pointing out user requirements that may be overlooked is.

Another feature which we did not consider for implementation in Hebrew, but which is relevant for future enhancements, is the idea of kashidas, just as in Arabic. I was not aware that Hebrew has similar typographical conventions as Arabic in this regard, although in modern usage this has become fairly rare. It was when Professor Amiras shared with her class one of her family's handwritten *Torah* scrolls that I saw first-hand the implementation of this concept and took the picture found in Appendix Five. Discussing this with her and with others versed in

Jewish culture and history, I slowly pieced together the uses of this form of character spacing. I also tried to determine why this form of writing is not commonly acknowledged.

One possibility why the usage of elongated glyphs has become less frequent which merits further research would be that Hebrew typesetting developed early and moveable type has no real ability to accommodate kashida-like structures. Arabic, which has a particularly rich and deep-seated tradition of calligraphy came relatively late to the domain of either lead-based, movable printing or digital typography, and as a result, still retains more of its hand-written heritage and conventions. This is in keeping with the research of Thomas Milo, author of *A Model for Handling the Arabic Script* (2005) and *Arabic Script and Typography: A Brief Historical Overview* (2002), annual speaker at the Unicode Conferences, and founder of DecoType. In contrast, after the invention of moveable type, the typographic conventions which swept through Europe also influenced Hebrew writing. With only a few exceptions, Hebrew letters are atomic and non-complex, just as Latin and Cyrillic letters are, so that this evolution of Hebrew would be expected.

While this is another example of a feature with a low return on investment for the initial release of a new language, it is something that should have been at least noted as a possibility. Enabling such capability would have required modifying at least one font, but for this effort Adobe did update one entire font face to support Hebrew, so this would have been an excellent time to at least consider such work. The ROI for such an effort may be small, but we certainly have a number of equally esoteric features in Japanese and Chinese.

While none of these issues are particularly troublesome in isolation, my concern is that there was insufficient effort placed on getting to know the Hebrew customers and address their needs. It begs the question: What other issues did we fail to consider and did any of these result in substandard experiences for these users? This is actually an investigation which I am continuing to carry out through my admittedly-limited network of Hebrew-speaking Photoshop users. In many way, Adobe's attempt to enter the MENA market was focused on Arabic-speaking users, particularly those with a command of English and/or in Egypt or the Gulf States.

Validation and Evaluation

Creating a Distributed Network of non-English User Feedback

A theme throughout this report is how hard it is for North American software engineers and designers to understand our foreign customers and tailor our product in such a way as to offer them the best, most culturally-relevant software experience possible. One of my biggest goals in my anthropological intervention for my Adobe project was to find a way to do a better job hearing and understanding our users. In some ways I think I was quite successful and in others less so.

Watching customers work and observing how they use the product and noting what activities they attempt and which cause them frustration, can be helpful. It provides opportunities to see what parts of the workflow are effective and which are troublesome. It also gave me knowledge of what users actually do rather than what I think that they do or what they report that they do. Unfortunately, it is usually not practical to actually observe customers, given that my focus is on users who do not speak English – most are physically located in other countries and, based on my experiences shadowing local graphic artists, I gain the best insight when I am actually physically present in their office with them, watching every move they make. Other solutions are clearly required.

The initial step, however, is simply finding customers. As mentioned earlier, this can be a challenge, especially in cultures where a formal introduction is needed. Luckily, both myself and my international project manager each came up with several potential solutions.

The first, of course, is getting technically savvy, experienced, and engaged foreign contractors to evaluate the current state of the product and offer feedback. Here money acts as the gatekeeper, allowing some egress to a viewpoint similar to that of our customers. And this feedback is quite valuable. Our vendors identify many bugs and give some great insight into the expectations of Arabic- and Hebrew-speaking users. Of course, relying on vendors is also problematic. While they represent the customer linguistically, it is a safe assumption that they do not share the customers' technical depth; Photoshop was designed for customers with degrees in photography, digit media, fine art, or other highly specific skills sets. In general, individuals with such skills would be better rewarded financially by applying those skills to use the product professionally rather than be paid as a linguistic tester by a localization house. As a result, vendors are unlikely to be as sensitive to highly technical workflows or functionality. More importantly, as money is the incentive, when the contract is over, these points of contact are no longer available.

Finding customers myself in the MENA market proved quite a challenge. While I did make contact with several, my Arabic is very basic and language can be a strong barrier. Further, as mentioned earlier, this is a context where formal introductions are very helpful and while I did have a few such offers, generally, I did not have much luck making lasting connections on whom I could rely.

My international program manager, however, was able to offer me a surprising opportunity. He and Samir, my MENA program manager, used their contacts as well as Adobe's various social media channels to inform the Middle Eastern user community that there would be a prerelease program offered specifically to our Arabic and Hebrew users. A prerelease program is an opportunity for users to try out a product under development and offer feedback during the process. In addition to getting a peak at the next release, participants who prove particularly helpful or insightful are given free copies of the application when it is completed.

Typically, with Adobe's emphasis on the English market, a prerelease program is something that is almost never done for other languages, except possibly Japanese, French, or German. When such a program is offered in another language, however, it often gives users an opportunity to improve both translations as well as issues confounding foreign workflows, because a plurality of native speakers can discuss the merits of various solutions. Photoshop had recently launched a German and French prerelease program to improve our Tier1 offerings and my international program manager was able to enable a similar opportunity for MENA customers. More than 1,000 customers, mostly Arabic-speakers, were invited to participate. To help deal with the extra work, one participant for each language was selected as a moderator, which included translating any topics not written in English for me. The results for both Arabic and Hebrew were rather surprising.

In the span of roughly five months, users discussed nearly 150 different topics, 84 for Arabic and 58 for Hebrew. In the case of several of these discussions, there was a fair amount of overlap: among the Arabic users, 63% of the issues raised were ones we were already tracking and among the Hebrew users, this number rose slightly to 66%. These users tried out the innovations we made and gave us feedback on our design, although due to time constraints, several issues had to be deferred for this release. As native speakers and experienced users, these customers found issues which we had missed, made suggestions about our designs and conceptual models, and generally validated many of the assumption under which we were operating.

For instance, after initially unveiling the redesigned kashida engine to a few corporate users, we shared this work with the prerelease community. They tested it out and quickly reported back how much of an improvement it was over the earlier design. Most of the comments on these threads were actually questions on how to activate the feature and how to make it easier to use and more accessible, something we had already been working on. Unlike our corporate informants, our prerelease users represented a variety of users from all over the Middle East, with different levels of technical sophistication. They served as a very distributed network of subject matter experts with little overlap among themselves. This diversity was helpful as it allowed us to be more confident with other product decisions as well.

At least for the Arabic speakers, the prerelease discussions put to rest any idea of localizing the product; several customers used humor to make it clear that even an attempt to translate the product would be an exercise in futility. The digital ecosystem -- the constellation of services, training, and support for digital typography and imagery -- is still evolving and the reliance on English for technology in general and digital imaging in particular would actually frustrate users of an Arabic UI. Certain terms simply are not translated in culture, so an attempt on our part to impose an artificial translation would merely lead to confusion. Over the whole span of the Arabic prerelease program, only two users voiced any dissent from this assessment and even they admitted that the majority voices were correct.

This was an interesting contrast with the Hebrew users. There, although the majority preferred an English user interface, the idea was much better received. In fact, the topic was initially broached by customers who wondered if a Hebrew interface would be available. This is another example of how these two populations really need to be viewed in distinct terms; there might be some overlap, but in general, our design did not give enough weight to the perceptions and expectations of our Hebrew-speaking customers.

The prerelease users did not agree with every implementation we made and the Hebrew users in particular asked for feature enhancements, such as supporting diacritical repositioning and allowing users to change the text engine associated with a given layer, both features that we had originally hoped to deliver but which we later removed from the product due to constraints on time and quality. Both of these are topics which are still under discussion today and which several of these prerelease users have continued to voice in other forums and through other communication channels.

This also highlights one of the cultural differences between the Hebrew and Arabic users. While we had roughly the same number of users in each final population of participants (96 Arabic and 89 Hebrew), our Hebrew users were more vocal for improvements and changes to our design, reasonable given that Israel is a world leader in high tech. Further, their posts tended to involve more complex issues, rather than simple “how do I…?” inquires. Unfortunately, 65% of the suggestions from our Hebrew-speaking participants had to be deferred, most commonly due to design limitations and lack of resources because of our schedule. The Arabic users submitted more bugs over all, but the number we actually fixed for each population was the same and most of our Arabic reports came from four specific users, while the Hebrew users’ bugs were more evenly distributed.

Implementing this prerelease program allowed us to build a fairly diverse network of informants. Unlike with most of our other contacts in the Middle East, most of these users did not know one another and only had the program in common. Whereas the results of numerous customer visits to the region by our project manager, Samir, seemed to elicit a fairly uniform set of responses, our prerelease users seemed to hold a more diverse range of opinions, particularly the Hebrew participants. At least eight bugs from this effort were identified and fixed and many more recorded for future implementation. In short, the prerelease program served as an alternate means of testing many of our operating hypothesis using a novel group of informants. This is important as, based on my experience in the field, often information networks can be very redundant.

It is worth noting that this prerelease program was unique and has not been repeated. Further, due to our efforts, there was quite a bit of attention paid to these users' complaints. The total number of Arabic posts even exceeded our German forum (84 vs. 58), although the Germans did manage to discuss their topics in far more detail (407 replies vs. 228). The next largest sets of users, the French and Hebrew speakers, came in at a fraction of this: 49 vs. 33 topics and 152 vs. 92 replies. Moreover, since this project, no attempt to survey or reach out to MENA users has been attempted.

Corporate Evaluation

From Adobe's perspective, the MENA-enablement of Photoshop was a great success. All critical features were enabled as well as all of the high priority and medium priority ones. Of the project goals in the final project requirements document (PRD), we achieved more than 77% of them. While a few bugs and unimplemented features remained, "the product quality going out in CS6 was way better than what we started [with]…We actually made quite a bit of change, fixed multiple issues, updated UI, added new fonts, and more" (email to the author, Samir 2013). Further, by collaborating with our colleagues on the InDesign and Illustrator teams, all three products benefited from additional capabilities and shared enhancements and features. Adobe's internal system to track successful internationalization, our Globalization Report Card, showed a marked improvement for all three products as well. The MENA effort had no impact on the Photoshop shipping schedule for CS6 and none of the changes made caused any other problems nor triggered a dot-release. The prerelease effort not only reached more users than ever before, but served as a model for later expansion of other languages.

Potentially, the MENA-enablement effort allows for half a billion additional users to effectively use the software and is relevant to consumers in 25 countries.

Financially, the MENA effort was quite successful for Adobe. The company more than exceeded all of its growth targets in the region and now the Middle East and North Africa have become standard markets for several of the product teams (Tier5). One contractor who greatly contributed to the effort was hired permanently to help address MENA issues. Arabic and Hebrew enablement concerns are now considered when implementing new features and several bugs and enhancements discovered during this work are under consideration for future releases. In recognition of these corporate goals, the project manager for the MENA effort was promoted and most of the people working on the project received various rewards.

This is not to say the results of this process were perfect or ideal. The number of deferred MENA bugs was particularly high at nearly 48%, roughly twice the typical deferral count. Further, not a single one of those issues was addressed in the following release either. In addition to these problems, there were different implementations of MENA-enablement for different teams.

Although Photoshop, Illustrator, and InDesign share all the same basic Arabic and Hebrew typographic capabilities, due to architectural differences noted earlier, only InDesign was able to include a script to recolor Arabic diacritics. At the same time, however, only Photoshop enabled the suite of MENA-enhancements in the core product; Illustrator and InDesign users must purchase and install a specific version of the product which is only in English or French to access these features. This is problematic for several population segments which the project manager identified, such as Ukrainian-speaking Hebrew users and Turks who wish to do Arabic typography. Because I work with Photoshop, most of my contacts with MENA users are in this context and as a result the most frequent question I get on my website[66] or on forums is how to activate the Arabic features in Photoshop. The second most frequent question is why InDesign and Illustrator do not also allow for the same universal access to MENA capabilities.

Due to the existing product architecture as well as limited resources and anxiety about potentially destabilizing the core type functionality in the products, the MENA features were added to the products using a supplemental type engine. This is not a standard practice any longer, dating from when different geographical regions used different product components. Unfortunately for customers, although other applications might have similar solutions, this complexity is completely hidden and as a result, the requirement for a user to proactively define what kind of text they are going to type before typing is effectively unique to Adobe's product line. This results in many questions and complaints from users who try to type مرحبا (marḥaba or "howdy") but wind up with م ر ح ب ا (nonsensical garbage) because the software has the wrong text engine active. In order to combat this problem and raise awareness of this issue, I created a video on YouTube detailing how to access the MENA features in Photoshop. This has proven to be an excellent alternative method of reaching out to users, particularly those who are not corporate customers. Further, based on the feedback from this source alone, our customers are extremely happy with our MENA support[67].

Anthropological Evaluation

From the perspective of an anthropological intervention, I am afraid that this project was successful, but nowhere near as much as I was hoping. As detailed in the preceding chapters, the final product was culturally sensitive so that it was designed with a broader perspective which included a number of feature modifications which were meaningless to

English-speakers but which support the needs, wants, expectations, and perspectives of MENA customers. There were, however, several aspects which reflected a lack of anthropological experience, rigor, and general naivety on my part.

Through the prerelease program, I was able to create a diverse network of MENA users who represent a different set of diverse viewpoints than the corporate customers who would otherwise been the only validation for our beta release. Thanks to my international project manager and my MENA project manager, Adobe managed to contact more than 1,000 Arabic users. I was quite intimidated by this number and was thankful when roughly only one tenth actually joined the program. I felt certain that with more than a hundred users, we would have an adequate sample of perspectives across the Arabic world, from the Maghrib to Egypt to the Mashriq and the Gulf States and beyond. Unfortunately, this was not the case. Our sample was a snowball sample and therefore relatively biased; we had very few customers from northwestern Africa and as a consequence our support of this region suffered.

Looking back with hindsight, I realize that it would have been better to allow for an additional North Africa French forum, so as to be more inclusive to users from the Maghrib, since French rather than English is the technological *lingua franca* for this community. I believe, based on earlier experiences with French-speaking users on English-centric forums as well as my current efforts to locate Moroccan customers, that users without a strong foundation in English will often avoid commenting or participating in these kinds of exercises rather than put themselves at risk to fumble with language. This is one of the very reasons our forums for European languages are explicitly welcoming to both English as well as the language in question, with emphasis on the latter rather than the former.

What could have also helped alleviate our lack of Western input would have been to make an effort to actually stratify our sample of users. Instead, I simply trusted that the sample size to be sufficient[68]. Although Egypt represents roughly a third of the native Arabic speaking world, it and Syria seemed over represented compared to other regions, particularly so given their rather low per capita internet access. On the forums, as an ice breaking exercise, the moderator asked users to introduce themselves; I recall many users from these two nations but none coming from either the West or the Gulf States. Admittedly, most participants skipped this exercise, but the impression remains that our pool of experts was not as diverse and inclusive as it could have been. This was fine from a corporate perspective, but not so from an anthropological one and I further believe that ultimately it was this lack of diversity among users which somewhat polarized what made it into the final product; only a few voices complained about the lack of AZERTY support, for instance.

A second aspect where I could have improved the process would have been to more carefully analyze the needs of the Hebrew user. I failed to sufficiently delve into the cultural perspective of this population. The truth is that the Hebrew writing system lacks the technical complexity that Arabic has. There are only a handful of contextual shapes, none are joined, and due to the mechanics of software layout merely mirroring a left-to-right text flow solves the majority of issues. Because of this, I did not dig further into the linguistic needs and expectations of Hebrew users, assuming that these would all be met through the support of our Arabic customers. While this is partially true, because Hebrew-focused needs were not discussed, on the whole Hebrew-facing features were an afterthought in the design process. In many ways, I treated Hebrew as many core programmers treat internationalization in general; a necessary task embraced only near the end of the product cycle rather than a planned requirement from product inception.

This brings me to a certain level of naivety about both my own part in the process as well as the perceptions of my colleagues. With consent from my supervisor, I modified my work roles and explored different aspects of this project in order to attempt a few solutions that were more anthropologically-informed than is typical in international software design. Although I believe this ultimately led to a better corporate as well as academic outcome for this project, not all of my coworkers were enthusiastic about my additional role. My boss and project managers were very supportive, but I quickly realized that through these changes I confronted added resistance. I did not understand this at first and simply dismissed it as anxiety and even resistance to supporting such alien languages as Arabic and Hebrew, which were novel and unique at the time. After analyzing the interviews of internationalization workers as well as quotes from my colleagues working on the core product, I think that some of this tension stems from my own overly simplified view of how the process works.

I assumed that there would be a theoretical standard for how internationalization is done and I tried to compare and contrast that with actual practice. While these were valid categories, I think these are only two extremes with many stages in between. While these categories varied between individuals, one of the things which I discovered through analyzing this process was that generally there were closer to four distinct states to contrast.

The first category is fairly straightforward: the academic/literary standards for globalization and internationalization as taught at MIIS and as written about in the literature. Yet, based on interviews, conversations, and even presentations at the Unicode conference, only advisory committees and non-profits, such as the Unicode Consortium and Wikipedia, actually use this as a guide. The reason is simple: this model places linguistic equality at such a high level that profitability suffers. This works for volunteer organizations, but not corporations interested in making money. Drawing from this year's conference, for instance, Wikipedia has invested in and fully supports the multiple scripts of Burma, even though many of these languages are dying out and the number of speakers numbers in the thousands, with far fewer having computer access (Sharma 2013 as well as Cornelius 2013). Support for these languages is complex and difficult with little return on investment for this effort.

This leads to what I found to be second category of internationalization standards, that which experienced professionals in the field feel are reasonable, in light of the demands of profitability and return on investment. These could be thought of as a constrained theoretical model and it is these standards which I personally use as a goal. These are very similar to the academic standards, but recognize the balance between idealistic support and practical requirement to turn a profit. The perfect solution is tempered by a need to conform to schedules with limited personnel and time. Throughout my interviews, globalization workers discussed how they try to meet development teams "in the middle" while still pushing for inclusive internationalization standards. In general, this category is what these colleagues are advocating. There is more recognition for limits to support, based on a variety of factors.

Much of software internationalization work is constrained by what core product teams are willing to do, which is the third category. Internationalization departments generally lack much authority to enforce corporate standards and product marketing is focused only on the most lucrative of markets. This is as true at Adobe as it is at Amazon (Phillips 2013) and Intel (conversation with the author, Danielle Tullo October 17th, 2013). As detailed earlier, product teams are often very resistive to changes in product design simply to enable international users. The need to support non-English-speaking customers is recognized, but a strong financial case must be made in order to justify the work to allow this. I would say that these were the standards under which the project detailed in this report actually operated.

Finally, the fourth category is what is actually done. Due to architectural limitations, resource constraints, lack of insight, and compressed schedules, what is planned is often not what is achieved. As Pam Clark, Director of Product Management, put it "when things get tight, internationalization is the first to go" (conversation with the author, October 3rd, 2013). This category then sums up the typical final product.

If I were to restart this project today, I would more carefully look at where different players in this process clump across these four categories. Based on Pam Clark and David Hackel's comments quoted earlier, it seems fairly obvious that the commitment to internationalized designs even at a managerial level lags rather far behind the goals and, as both Bob and Julia mentioned, evangelism of globalization workers. This strikes me as a good topic for further research, especially in light of which products and teams have successful internationalized releases.

One thing that my anthropological work on this project brought into focus is that software development models which do not take into account culture produce not only less effective tools for foreign users, but they continue a form of cultural imperialism -- in the realm of typography, the unique expectations and paradigms of other cultures are forced to conform to Western, often American, models as these are the only digital solutions available. In the minds of many MENA customers, even if the software does not fully support regional paradigms, digital technology is often conflated with modernity, a state they desire. Further, time is money for most people and digital tools offer the potential to expend less of both. Unfortunately, support for expressions of culture which are not relevant to a European-focused workflow are frequently overlooked and thereby potentially lost. As ubiquitous software applications such as Word, Photoshop, and Excel, enter their third decade on the market, the capabilities of these programs enhance Western workflows while challenging other users to choose between "the old way" and a streamlined, culturally-limiting digital solution.

Generalizability

This report details the international design and development process at Adobe with specific attention to how the Photoshop team accomplishes this work. Based on interviews and interactions with other employees on related teams doing analogous work, such as Illustrator and InDesign, these findings are largely applicable to them as well. I have only interviewed a few individuals at other companies and spent no time studying these other organizations, so a direct translation of these findings to those contexts is neither justified nor fair. Such a study would make an interesting extension of this work to see whether or not and to what degree other software development firms incorporate international perspectives into their designs and what actions can be taken to enhance this process.

To the extent that Adobe is fairly typical of most software development firms, then I believe it is reasonable to generalize these findings elsewhere. This is based not on the process in which the work is done but rather due to what the final product is designed to do. Here I am analyzing the software tools these companies produce, looking for designs that are inclusive of other perspectives and workflows.

I spoke with a friend working at Microsoft and gave him an earlier working copy of this report. He reported back that Microsoft does not operate along these lines and that his particular group, Web Services, is particularly focused on a high-quality internationalized product. He cited for me how geographic maps had to be carefully screened for potential political statements, such as whether or not the State of Israel or Palestine is marked or how Taiwan is referred to. With this in mind as well as comments from my academic advisor, I greatly reduced the initial scope of my claims.

Then, while preparing some of the graphics in this report, I was struck by a thought. If the designs of other software developers are internationally informed, then that work should be fairly obvious in the final product. This analysis is in part about how language, a codified form of culture, impacts tools and their usage. So I looked for internationally-sensitive designs in one of the oldest and most commonly used software product today.

Researching Microsoft Word, I found that while there are a couple of enhancements included since 2003 for right-to-left languages in general, the application lacks any support for Arabic-specific features. There are a number of universal features which have been altered to support Arabic, such as a cursor that changes to indicate text flow and added controls for text directionality (Microsoft 2013). None of these features, however, is unique to Arabic; they are enhancements of existing tools which are relevant to English users or wide categories of users in general, such as all right-to-left languages. There is no capability to do to Arabic text something that is not relevant also to Latin text, such as automatic kashida insertion or diacritical positioning or coloring. Basically, a localized Western model is being given to the Eastern world with only a very limited attempt to adapt to that region's needs or conventions. In short, it sounds like the underlying design for Word remains fixed as a tool for European languages rather than a word processor which embraces emic conventions of non-Latin scripts.

In case the lack of specific linguistic features was only an issue with Arabic and Hebrew, I then researched Japanese support in Microsoft Word and while there are a few enhancements, most of the emphasis remains on adapting Western designs to fit Eastern standards. There is no support for Mojikumi or Kinsoku for instance, and most of the emphasis seems to be on supporting Japanese input methods and character conversions in addition to offering customers standardized document formats. Rather than include integrated features relevant to non-English speakers, Microsoft Word seems to have adapted existing English models, in keeping with a development model which has internationalization done at the end of product development.

Thinking of other applications which could have language-specific features, I recalled from the 2007 Unicode Conference a talk by Forbes and Li about their work to enable MENA support in Google's gmail application. This is particularly relevant as two of the individuals cited in this report also worked on that product. While Forbes did mention the importance of cultural sensitivity, especially with regard to testing feature design and user workflow, I could find no specific Arabic or Hebrew features in the final product. Again, the final capabilities of the software are analogous functions to what is available in the English product.

As a result, while this report is specifically focused on Adobe's process for internationalized software design, there is evidence that some aspects of it could be generalizable to other firms doing similar development work. The specifics and level to which this information is transferable, however, would require someone to conduct analogous research at the company or companies in question.

Limitations

This research only applies to commercial, off-the-shelf software development. While the case could be made that medical software, for instance, should respect local conventions and cultural definitions of illness and wellness, the very nature of those applications require a much more rigid standard of testing and validation. Quite simply, health outcomes and lives are involved; the appropriateness of medical software designs is a question which involves medical practice and biological science. Further, while software is just another tool, this research is not relevant or necessarily applicable to the development of non-software tools. While it is logical that a case could be made along these lines, such is completely outside the scope of this work and would require quite a lot of supplemental research.

I would also feel uncomfortable assuming that this report applies outside the context of desktop software. While there are similarities in the design and development processes for tablet and smart phone applications, the real world concerns of financial feasibility and return on investment are notably different. The speed and scale of software development in this arena changes the context of the discussion. The interaction of specific hardware, availability of local markets, and even social conventions for appropriate tool usage would require much further research. Naturally, culturally-informed designs are advantageous, but there are liable to be many additional considerations in this context which I do not address here. As a result, more inquiry would be required.

Summary and Reflections

International software development can benefit from the inclusion of an anthropological perspective, particularly during the design process. While the importance of culture is acknowledged in the literature and in practice, the actual methods to gain and even apply a sensitivity to foreign perspectives is usually ignored in industry. Wikipedia's article on this topic sums up this attitude best, "To some degree (e.g. for Quality assurance), the development team needs someone who understands foreign languages and cultures and has a technical background" (Wikipedia 2010). Further illumination of what particular skills are helpful is almost totally lacking as are methods to implement cultural knowledge. Research shows many examples of the kind of cultural faux pas to avoid in the final product, such as with colors, images, sounds, capitalization, and technical standards (OpenOffice 2013, Microsoft 2013, Abufardeh 2009, Purvis 2001, Rafii 1995) as well as very detailed steps leading through each of the technical aspects required to internationalize an application (Microsoft 2013, Kaneko 1999, Takezaki 1999), but there are no proactive suggestions on how to fine tune a product to meet the needs of a diverse group of foreign customers.

The lack of ethnographic research into the desires of foreign customers is sharply contrasted by how frequently domestic customers are studied. It is very normal for Adobe to not only have prerelease programs for English-speakers, but to literally have a dozen different programs, each focused on a particular aspect of the product, such as 3D printing or image cropping. Meanwhile, members of the user experience team (XD) regularly conduct site visits with American customers in order to observe their use of the program in an actual work context so as to improve the overall design and workflow. On the Photoshop team in particular, both developers and quality engineers are encouraged to accompany XD at least once a year on these visits. Such effort, however, is almost never focused on foreign customers, as was done in this case for Middle Eastern users of Photoshop. There are only two examples of culturally informed software design of which I am aware and neither incorporated the modifications into the core product[69].

A culturally sensitive development model can inform not just product design for non-English-speaking customers, but all aspects of the human-machine interface. Thinking of the software as a tool that exists in a specific context is salient for Arabic users, but it is also useful when designing a 3D modeling feature which could be used by doctors to assess DICOM scans of a patient or when prototyping HTML-enabled style sheets for web designers. In both cases, the user audience is markedly different than the typical user, with unique perspectives and expectations that are most easily ascertained through ethnography and observation. Further, using distributed networks of users with structured holes allows for a more stratified and generalizable survey of potential customers and their reactions to design modifications.

In this Adobe project to enable Middle Eastern and North African linguistic support into Photoshop, I attempted to incorporate a culturally sensitive perspective into both feature design and quality assurance work. The first step was to gain some personal insight into the languages involved as a means to see the typographic and socially-informed needs of the user community. Then, in addition to paid experts, I sought out a network of diverse and disparate users from the region in question who could serve as a population to validate our design hypotheses. Further, during the actual research and development phase of this effort, I tried to advocate and include as inclusive and informed definitions of potential customers and their workflows as possible, so as to create a program with broad appeal rather than one which met the needs of only a small segment of users. This included incorporating all culturally-specific elements into the final core application, so that even if our assumptions about who might want access to these features were incorrect, the tools we developed would still be available. This is particularly important when thinking about users who span multiple definitions of language and culture, such as, second generation Turkish Muslims living in southern Germany who may wish to reproduce a line from the Qur'an. Without access to the full suite of Arabic typographic features in either the German or Turkish version, such an endeavor would simply not be possible.

There is room for improvement in how Adobe in particular and how desktop software in general is designed and delivered to non-European and especially non-English speaking customers. Understanding one's customers is generally recognized as crucial to having a successful product. There is little effort, however, to learn about those customers that come from different geographical locations, particularly those from other cultures. The international software literature details the technical aspects of the work as well as warns of common cultural pitfalls, but there is no further discussion about bringing a sensitivity to culture into the design process. My hope is that the interventions I made during my work on the MENA-enablement project of Photoshop CS6 have uncovered additional, nuanced ways to perceive and address these needs. Based on the extremely positive financial and customer-based feedback Adobe received with regard to Photoshop's MENA efforts, the final design was very successful. From the perspective of cultural diversity, I find it to be slightly less so, but still worthy of appreciation. Unfortunately, the way products are designed at Adobe has not changed; the effort to enable support for MENA customers has not impacted existing models, though what the future holds is unknown as I have been circulating drafts of this paper to various decision makers in the company.

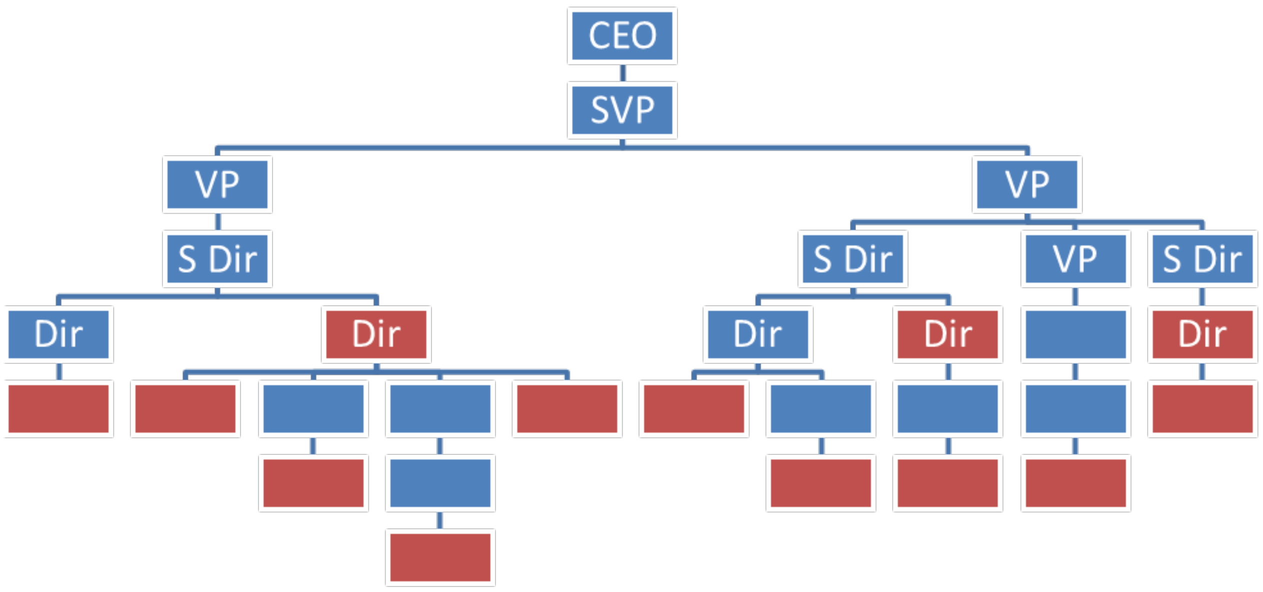
Today there are very few social scientists in the field of software design. As revealed through interviews, employees are hired for their deep technical skills, are assigned some aspect of globalization activities, and seem to remain working in such a capacity through a combination of personal desire and applicable cultural sensitivity, yet nowhere does there seem to be any recognition of the value of this sensitivity or even effort to recruit individuals with these talents. While there are exceptions, I believe that many anthropologists are not entirely comfortable with technology and may therefore lack the needed familiarity with software and programming languages to effectively interact with Engineering. I also believe that anthropology's desire to problematize is a great asset at the beginning of the project but can be damaging to morale and scheduling later; something must ship, whether perfect or not. Improvements can always be made in the next release, but only if there is a revenue stream to pay for future efforts. At the same time, anthropology's comfort with perceiving nuances and shades of gray is a welcome addition to software engineering where the working paradigm is often black or white, on or off, one or zero.

Comfort with novel and ambiguous situations is a valuable trait when designing for a foreign audience. While it is theoretically possible to hire engineers and designers from the culture in question, such a solution is largely unrealistic. Finding qualified candidates who are willing to be hired only for the duration of the project is the first barrier, particularly since the skills needed are in very high demand. Creating a team of such individuals is the second barrier; stability is crucial to software longevity. Beyond that, however, is the problem that natives of a particular culture are not likely experts in that culture and are likely to have difficulty identifying needed requirements which are outside their personal experience. Just as not everyone can write a concerto or code an application, not everyone can observe a population and gain useful insight into what that population wants and needs.

My hope is that in the future there can be a bigger seat at the table for social scientists and cultural experts dealing with international software design and development. More than other disciplines, social science in general and anthropology, particularly ethnography, has much to offer to this field. A survey of both seasoned globalization experts as well as topics addressed at the annual Unicode Conference (Phillips 2013, Allawi 2012, Forbes 2007) and by international MBA and Project Management programs (Khalifa 2012, Tchernoousko 2011) show the importance of cultural expertise, but such skills are generally not actively sought in the field nor even recognized as being particularly important professionally. Hopefully reports such as this and others can help change this perception and make a case for the inclusion of cultural skills in software design.

Appendix One

Globalization Respondents



The individuals interviewed or quoted in this project report correspond to the red boxes on this organizational chart. Employees who no longer work for Adobe have been placed in the position they had prior to leaving and employees of other companies have been placed in analogous positions, based on their corporate hierarchy.

Appendix Two

Crude Demographic Calculations

These spreadsheets were part of my initial attempt to both understand and pitch for an Arabic-version of my product, Photoshop, and potentially other products in the Creative Suite. The documents lack sophistication, but give an idea into how I was looking at the issue and how Arabic-speaking nations compare to Adobe's more traditional markets. This document was written in 2007, long before any of the corporate work described in this project report began.

Conservative Estimate of Worldwide Muslim Population:	1.6	billion
Estimate of Worldwide Human Population:	6.7	billion
Percent of human population that's Muslim	24%	
English speaking population of the World:	375	million
French-speaking population of the World:	140	million
German-speaking population of the World:	100	million
Japanese-speaking population of the World:	127	million
Tier1-speaking population of the World:	742	million
Spanish	358	million
Potruguese	250	million
Italian	62	million
Dutch	20	million
Danish	5	million
Norwegian	5	million
Swedish	9	million
Finnish	5	million
Tier2-speaking population of the World:	714	million
Tier1- AND Tier2-speaking population	1.456	billion
Chinese	Sorry -- no idea how to figure this	
Korean	78	million
Russian	167	million
Ukrainian	47	million
Romanian	26	million
Polish	52	million
Czech	12	million
Hungarian	14	million
Turkish	61	million
Greek	15	million
Arabic	422	million
UAE+Qatar+Bahrain+Kuwait	8.2	million

Sizes for various linguistic populations supported by various versions of Photoshop

\

	Pop in millions	Per Capita income thousands of USDs	SUM
TIER1			
US	303	46	13938
Germany/Austria	91	37.4	3403.4
Japan	127	33	4191
TIER2			
Italy	58	31	1798
Netherlands	16	38	608
Denmark	5	37	185
Norway	5	55	275
Sweden	9	36	324
Finland	5	35	175
TIER3			
China	1330	5.3	7049
South Korea	49	24	1176
TIER4			
Russia	140	14.6	2044
Ukraine	46	6.9	317.4
Romania	22	11	242
Poland	38	16	608
Czech Republic	10	24	240
Hungary	10	19	190
Turkey	71	9.4	667.4
TIER5			
Morocco	34	3.8	129.2
Algeria	33	8.1	267.3
Tunisia	10	7.5	75
Libya	6	13.1	78.6
Egypt	81	5.4	437.4
Jordan	6	4.7	28.2
Saudi Arabia	28	20.7	579.6
Yemen	23	2.4	55.2
Oman	3	19.1	57.3
United Arab Emirates	4	55.2	220.8
Qatar	1	75.9	75.9
Bahrain	0.7	34.7	24.29
Kuwait	2.5	55.3	138.25
Syria	19	4.3	81.7
Lebannon	4	10.4	41.6
Iraq	28	3.6	100.8

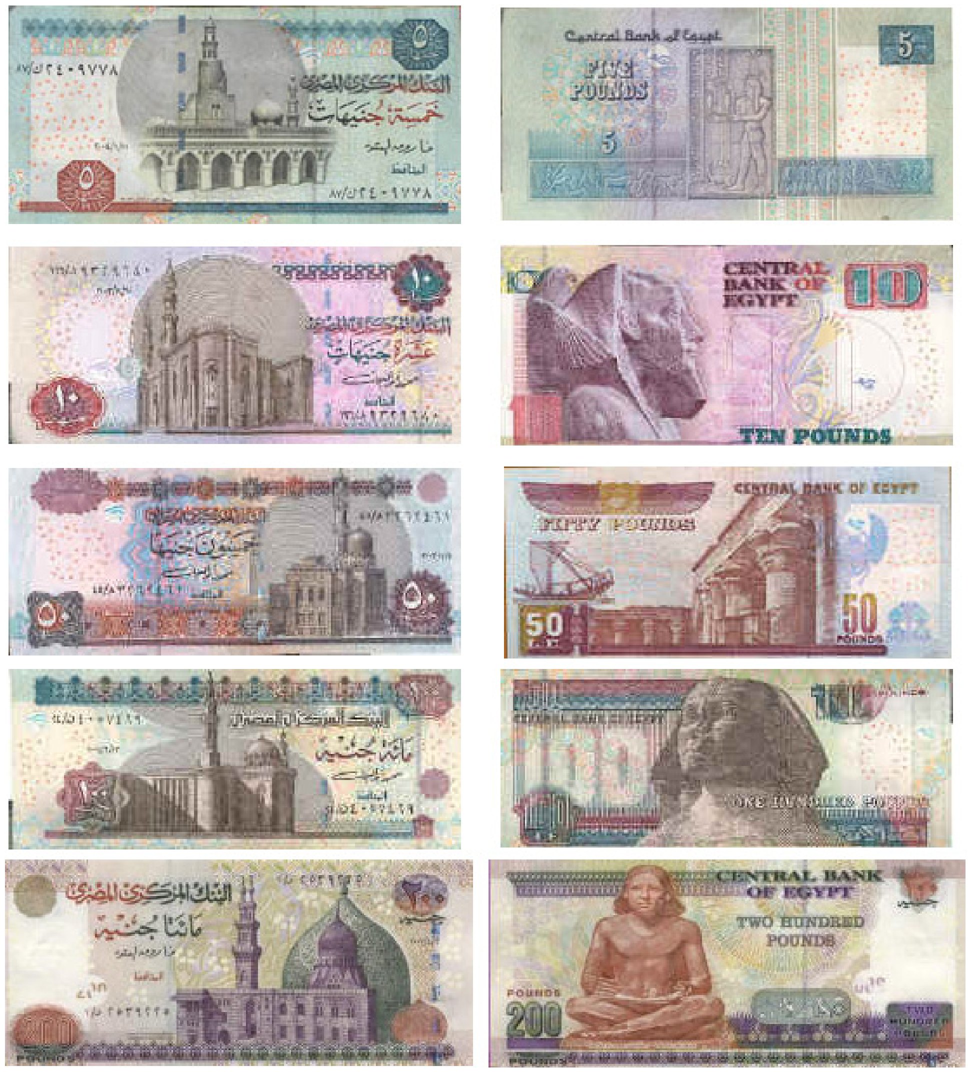
this one country ALONE justifies Arabic!

The justification for Arabic is because, exclusively looking at UAE, they have a population nearly the same size as Denmark and Finland (which we both each localize for), an average annual salary HIGHER than ANY country we currently localize into (Norway is #2 with $200 less per year for a pop 25% larger – both beat the US & Japan), yet compared with Hungary (another language we localize into) UAE has a higher GDP.

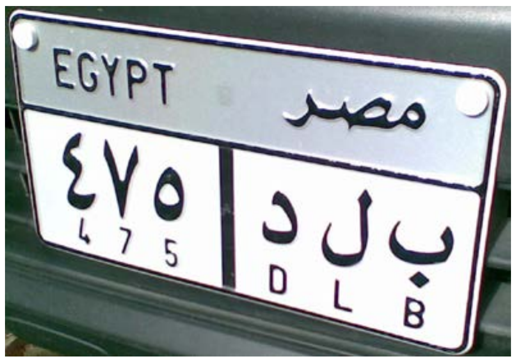
If you add in Qatar (far wealthier, per capita), Kuwait (roughly the same, although a smaller pop), and Bahrain (small pop, but salaries on par with other European nations), it is a complete mystery why we have not localized (or at least enabled) years ago – one language, larger combined pop than any Scandinavian countries, WAY higher per capita income…with the other 400 million potential users as “free potential income”.

Appendix Three

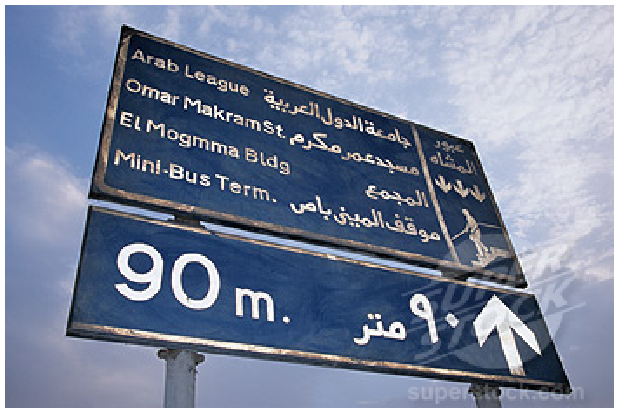
Numbers in an Arabic Context



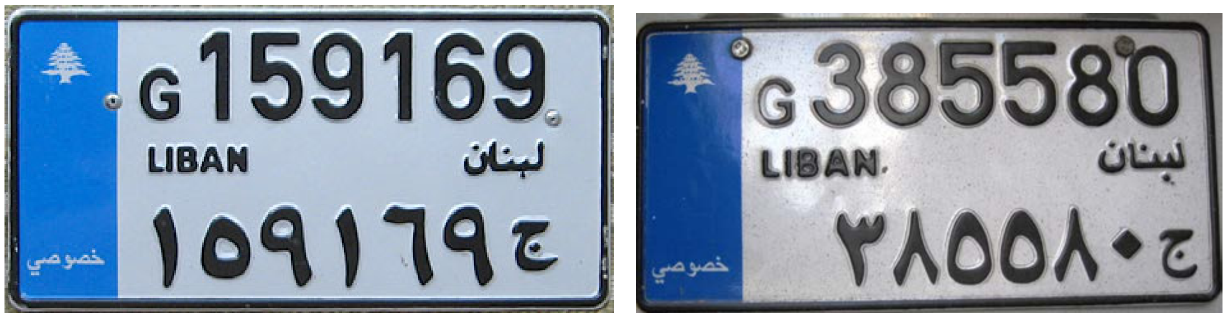
Egyptian Currency, displaying one language and numeric system on the front and another on the back.



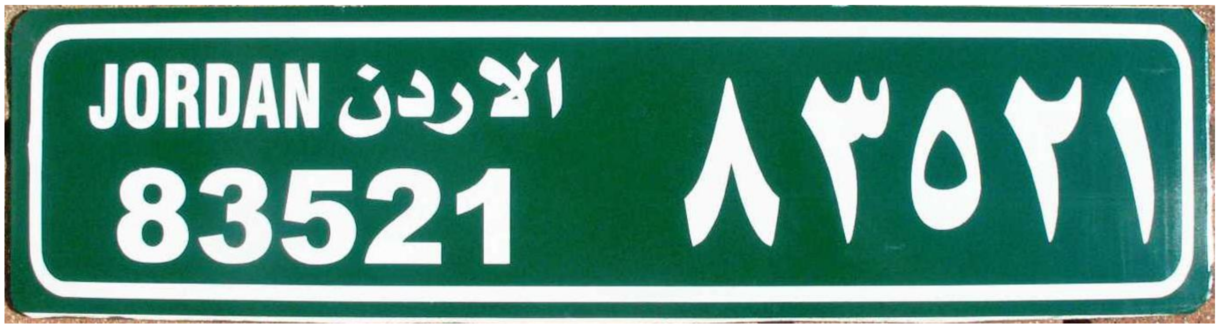
Egyptian license plate.


Arab League جامعة الدول العربية
Omar Makram St. مسجد عمر مكرم
El Mogmma Bldg المجمع
Mini-Bus Term. موقف الميني باص
عبور المشاة
90 m. ٩٠ متر

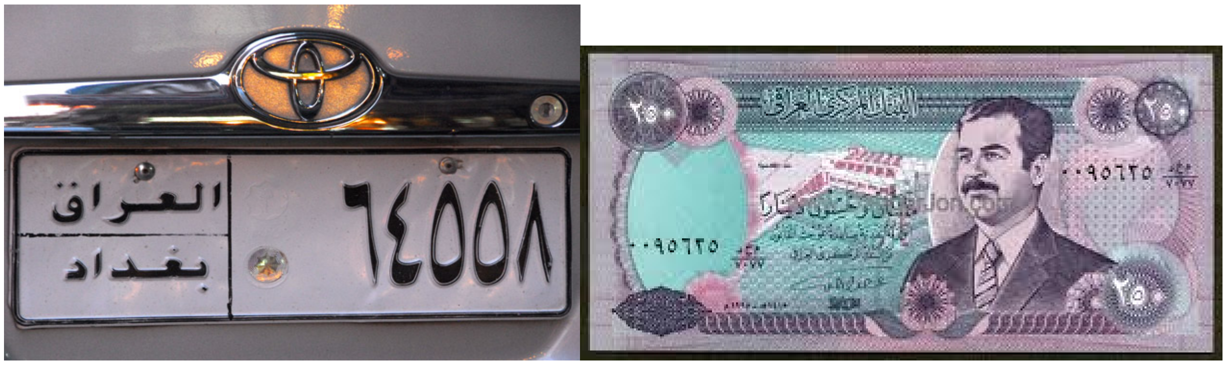
Egyptian Road sign in Cairo.



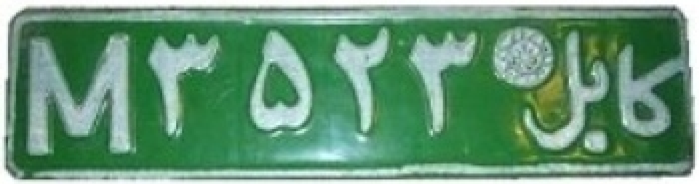
Lebanese license plates showing two language and two numeric systems.



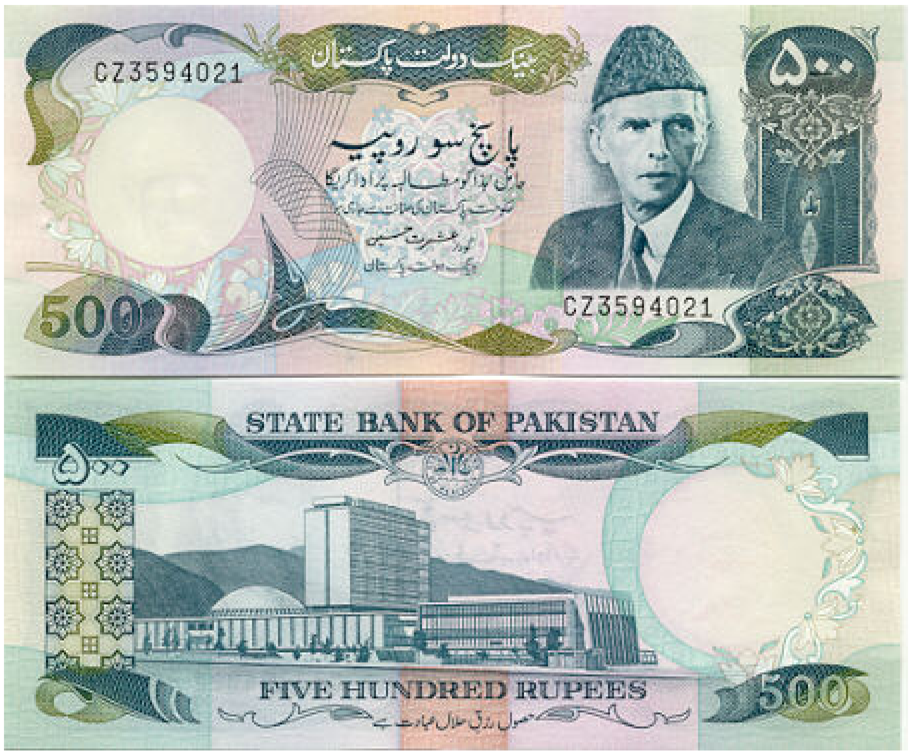
Jordanian license plate.



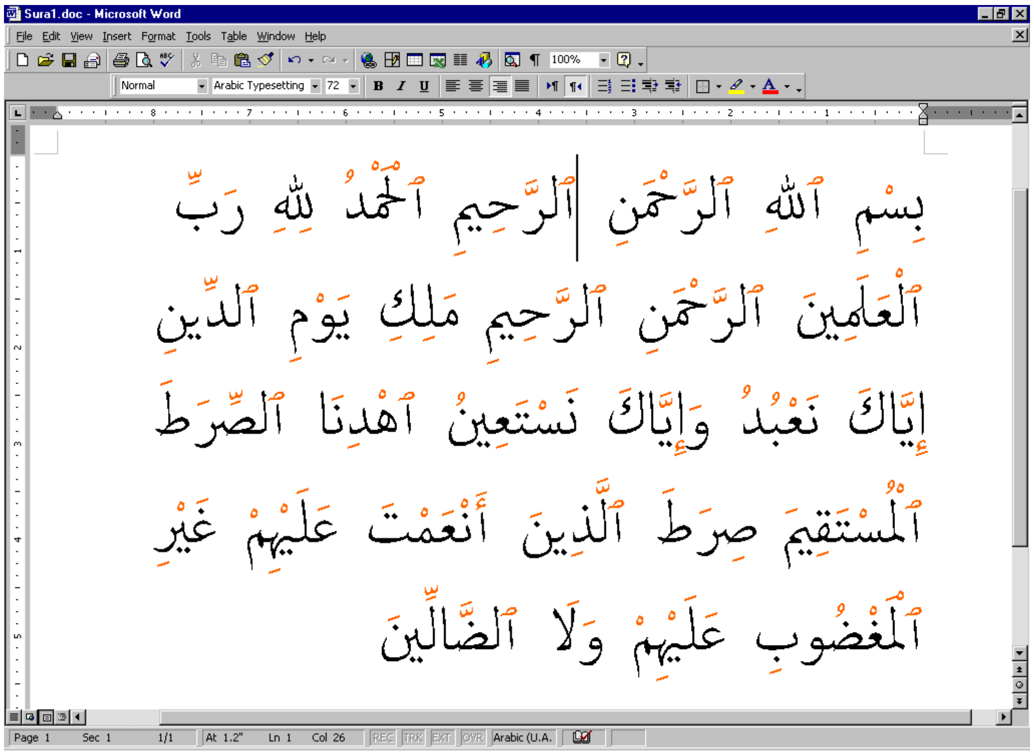
Iraqi license plate and currency, both showing only Arabic script and Eastern numbers.



Afghani license plate from Kabul, showing Farsi numbers.



Pakistani currency showing English and Arabic text, but Western and Farsi digits.


Sura1.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help

Normal | Arabic Typesetting | 72 | 100%

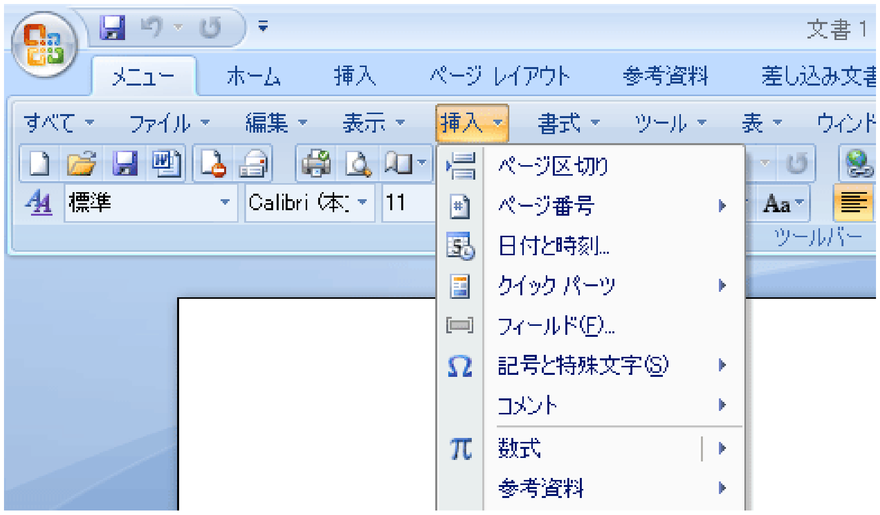
بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ ٱلْحَمْدُ لِلَّهِ رَبِّ
ٱلْعَٰلَمِينَ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ مَٰلِكِ يَوْمِ ٱلدِّينِ
إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ ٱهْدِنَا ٱلصِّرَٰطَ
ٱلْمُسْتَقِيمَ صِرَٰطَ ٱلَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ
ٱلْمَغْضُوبِ عَلَيْهِمْ وَلَا ٱلضَّآلِّينَ

Page 1 Sec 1 1/1 At 1.2" Ln 1 Col 26 REC TRK EXT OVR Arabic (U.A.

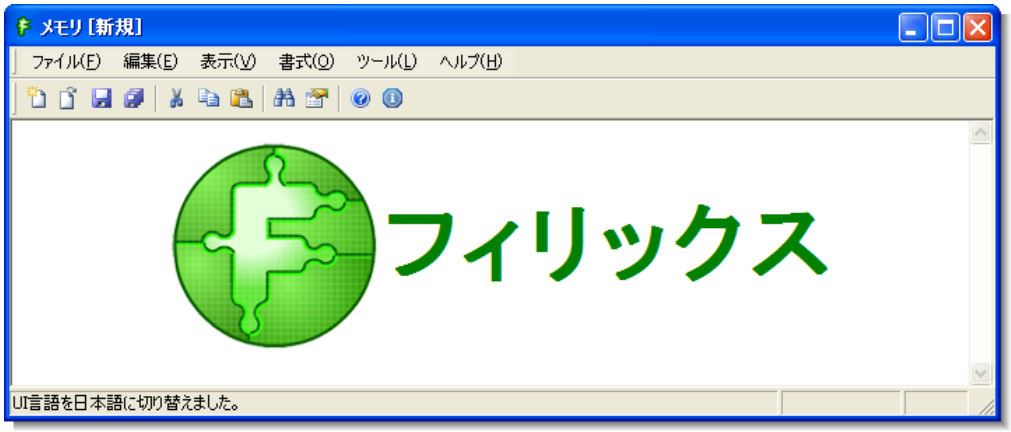
Sample text showing colored diacritics.
This cannot be done in Microsoft Word and shows a third-party solution (Wilhelm 2013).

Appendix Four

Non-English User Interfaces and Keyboard Shortcuts

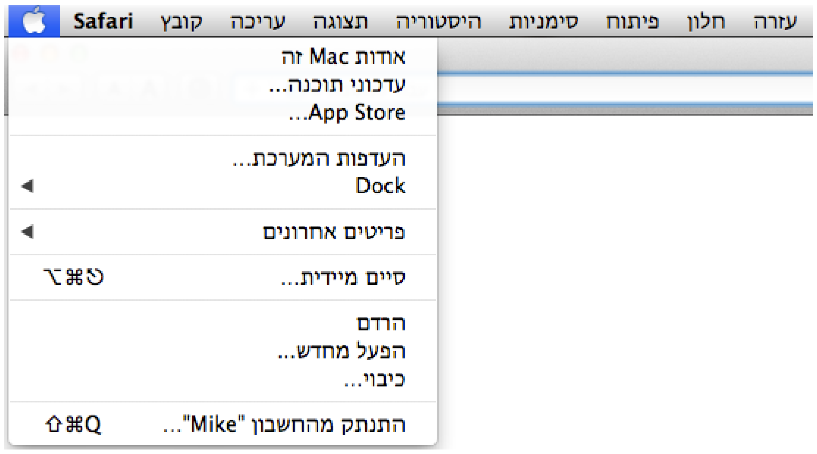


Japanese version of MS Word. Note the underlined English letters; these are accelerator keys.

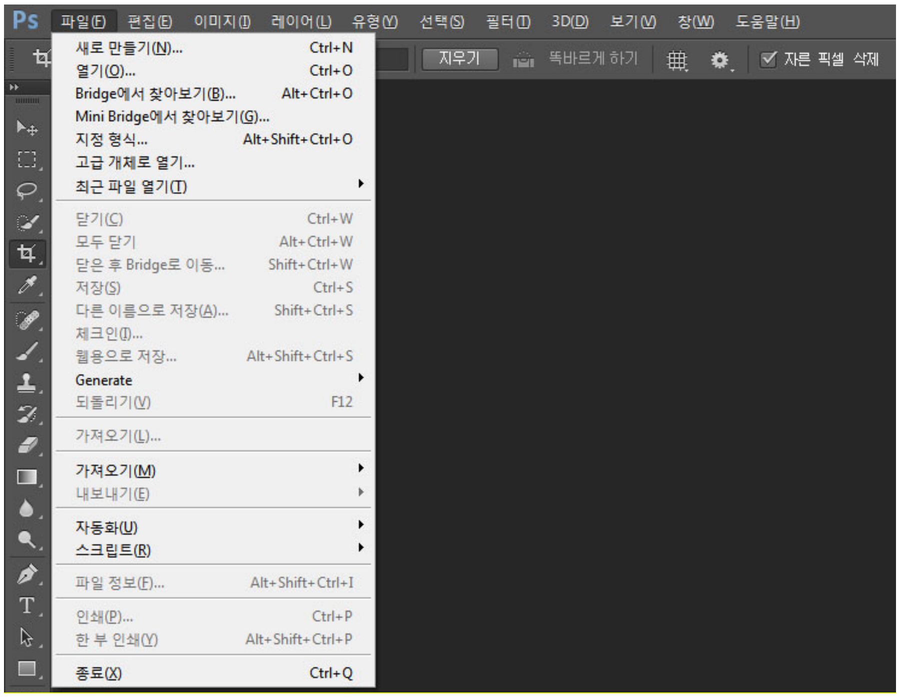


UI言語を日本語に切り替えました。

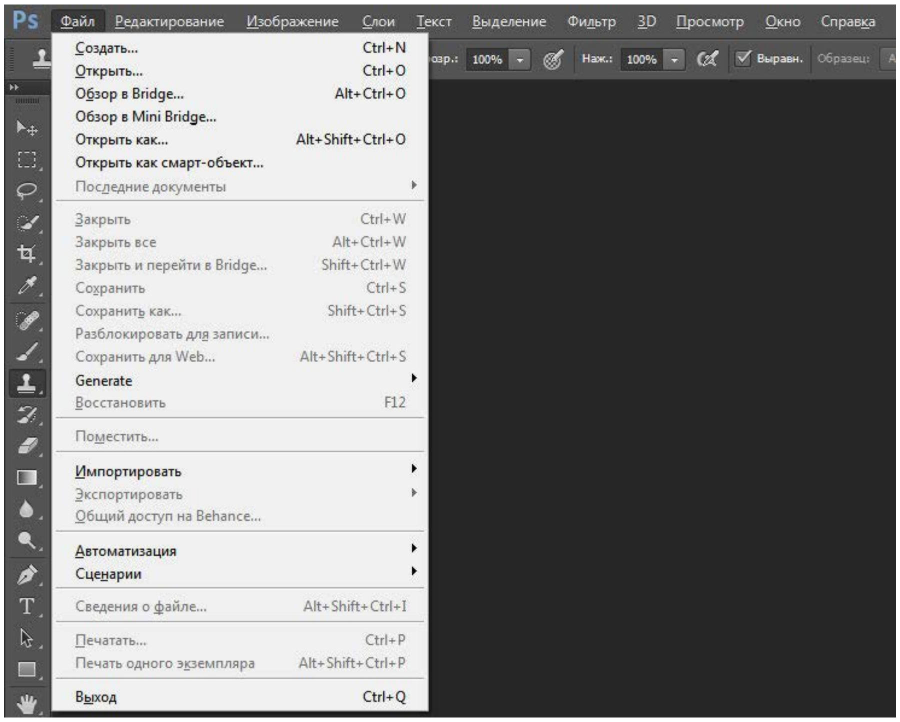
Japanese version of Felix. Again, the underlined English letters indicate accelerator keys.



Hebrew version of Safari. Note the shortcut key for Quit is in English.


Ps
파일(F) 편집(E) 이미지(I) 레이어(L) 유형(Y) 선택(S) 필터(T) 3D(D) 보기(V) 창(W) 도움말(H)
지우기
똑바르게 하기
자른 픽셀 삭제
새로 만들기(N)... Ctrl+N
열기(O)... Ctrl+O
Bridge에서 찾아보기(B)... Alt+Ctrl+O
Mini Bridge에서 찾아보기(G)...
지정 형식... Alt+Shift+Ctrl+O
고급 개체로 열기...
최근 파일 열기(T)
닫기(C) Ctrl+W
모두 닫기 Alt+Ctrl+W
닫은 후 Bridge로 이동... Shift+Ctrl+W
저장(S) Ctrl+S
다른 이름으로 저장(A)... Shift+Ctrl+S
체크인(I)...
웹용으로 저장... Alt+Shift+Ctrl+S
Generate
되돌리기(V) F12
가져오기(L)...
가져오기(M)
내보내기(E)
자동화(U)
스크립트(R)
파일 정보(F)... Alt+Shift+Ctrl+I
인쇄(P)... Ctrl+P
한 부 인쇄(Y) Alt+Shift+Ctrl+P
종료(X) Ctrl+Q

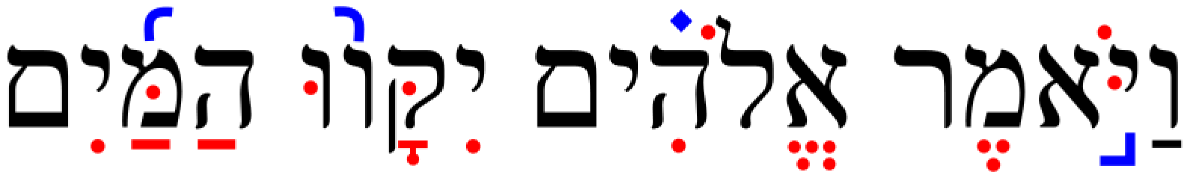
Screenshot of Korean version of Photoshop. Note that in addition to accelerator keys, there are also a number of shortcut keys displayed, such as Ctrl+N or Ctrl+O.


Ps
Файл Редактирование Изображение Слои Текст Выделение Фильтр 3D Просмотр Окно Справка
озр.: 100%
Наж.: 100%
Выравн.
Образец:
Создать... Ctrl+N
Открыть... Ctrl+O
Обзор в Bridge... Alt+Ctrl+O
Обзор в Mini Bridge...
Открыть как... Alt+Shift+Ctrl+O
Открыть как смарт-объект...
Последние документы
Закрыть Ctrl+W
Закрыть все Alt+Ctrl+W
Закрыть и перейти в Bridge... Shift+Ctrl+W
Сохранить Ctrl+S
Сохранить как... Shift+Ctrl+S
Разблокировать для записи...
Сохранить для Web... Alt+Shift+Ctrl+S
Generate
Восстановить F12
Поместить...
Импортировать
Экспортировать
Общий доступ на Behance...
Автоматизация
Сценарии
Сведения о файле... Alt+Shift+Ctrl+I
Печатать... Ctrl+P
Печать одного экземпляра Alt+Shift+Ctrl+P
Выход Ctrl+Q

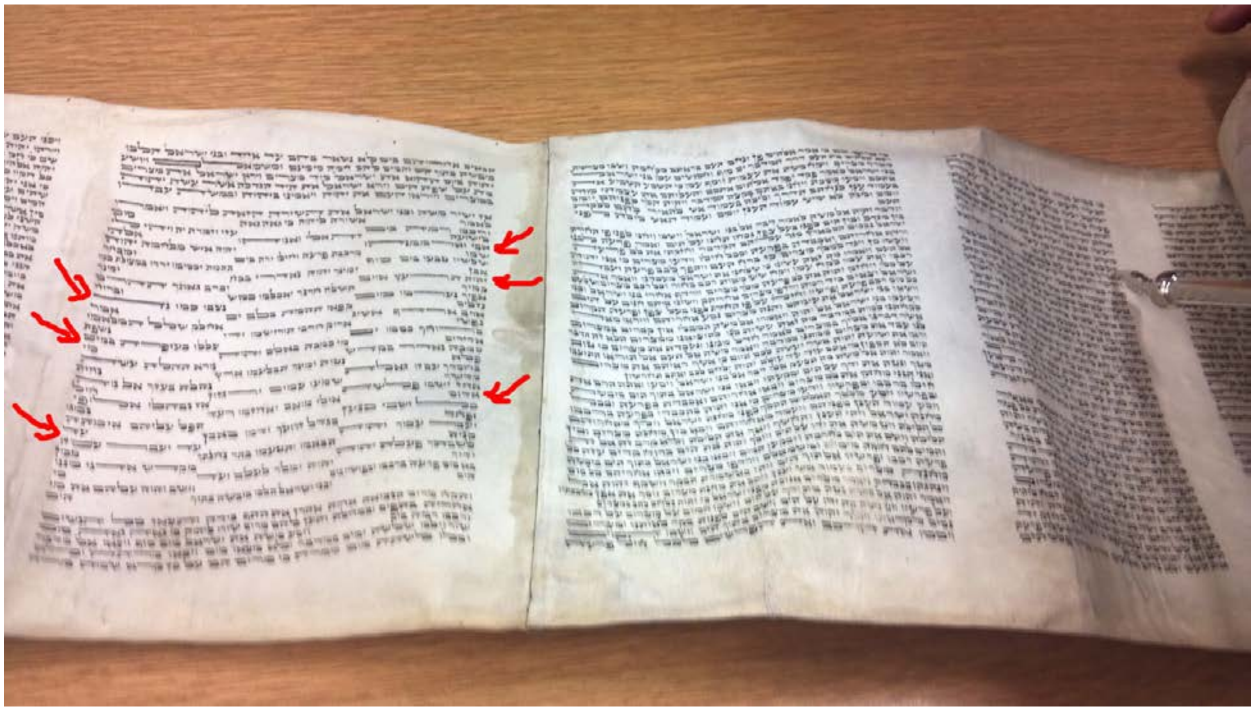
Screenshot of Russian version of Photoshop. Even through Russian uses Cyrillic script, Latin shortcut keys are used in the menus.

Appendix Five

Examples of Hebrew Typography



Genesis 1:9: *God said, "Let the waters be collected."*
The Hebrew letters are in black, the diacritical marks for vowels (and consonants) are in red, and the cantillation marks are in blue.


400 year old hand-written Torah with elongated Hebrew letters, similar to Arabic taṭwīls.

Bibliography

Abufardeh, Sameer and Kenneth Magel
2009 Software Internationalization: Crosscutting Concerns across the Development Lifecycle. Presented at the 2009 International Conference on New Trends in Information and Service Science. Stable URL: http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=5260815

Aldahleh , Ayman
2010 Tailoring the Unicode Bidi Algorithm. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20th, 2010.

Allawi, Adil
2012 Socializing Bi-Di. Presented at the 36th Internationalization and Unicode Conference, Santa Clara, California, October 24th, 2012.

Bordeau, Jessica
2010 The Beauty Of Typography: Writing Systems And Calligraphy, Part 2. http://www.smashingmagazine.com/2010/06/22/the-beauty-of-typography-writing-systems-and-calligraphy-part-2/ accessed October 11th, 2013

Burns, Diana
2006 InDesign and World Languages, Part 3: InDesign ME speaks Arabic and Hebrew—fluently! InDesign Magazine, August/September: 19-25.

Chapman, Christopher J.
2010 A Beginner's Guide to Indic Scripts. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20th, 2010.

Collins, Robert
1992 Japan-Think Ameri-Think: An Irreverent Guide to Understanding the Cultural Differences Between Us. New York: Penguin Books.

Collins, Rosann Webb
2002 Software Localization for Internet Software: Issues and Methods. IEEE Software March/April: 74-80.

Cornelius, Craig and Brian Kemler
2013 Burmese – Challenges and Lessons Learned from an Early Adopter. Presented at the 37th Internationalization and Unicode Conference, Santa Clara, California, October 23rd, 2013.

Cunningham, Cara A.
1993 Software Profits Speak Many Languages. Computerworld, Oct 4: 30.

Damian, Daniela and Deependra Moitra
2006 Global Software Development: How Far Have We Come? IEEE Software. Volume 23, Issue 5.

David, Vee
2006 The Kanji Handbook. North Clarendon, Vermont: Tuttle Publishing.

Diessner, Herwig
2006 Farbtreue Fotos: Geräte kalibrieren, Bilder justieren. Computer & Literatur. Böblingen: Germany

Dohler, Per N.
1997 Facets of Software Localization: A Translator's View. Translations Journal 1(1).

Ebbertz, Martin
2002 Das Internet spricht Englisch … und neuerdings auch Deutsch (The Internet speaks English … and lately, also German). Stable URL: http://www.netz-tipp.de/sprachen.html accessed August 18th, 2013

Editorial

1970 American's Moment of Truth. Time Magazine. October 26th, 1970. Retrieved December 18th, 2011.

Effective Language Learning
2013 Language Difficulty Ranking. Stable URL: http://www.effectivelanguagelearning.com/language-guide/language-difficulty accessed August 18th, 2013

Eldawy, Mohamed
2010 Tashkeel -- A library for automatically adding diacritics to undiacritized Arabic text. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20th, 2010.

Etherington, Darrell
2011 iPhone 4S: Siri's international limitations. http://gigaom.com/2011/10/14/iphone-4s-siris-international-limitations/ accessed October 12th, 2013

Ethnologue
2013 Urdu, The Language of Pakistan. Stable URL: http://www.ethnologue.com/language/URD/***EDITION*** accessed October 19th, 2013

Forbes
2013 The World's Biggest Public Companies. Added search criteria for domain of "Software Programming" Stable URL: http://www.forbes.com/global2000/list/#page:1_sort:0_direction:asc_search:_filter:Software%20%26%20Programming_filter:All%20countries_filter:All%20states accessed October 17th, 2013

Forbes, Shoshannah and Li, Shanjian
2007 Gmail BiDi Enabling - A Case Study from Two Perspectives. Presented at the 31st Internationalization and Unicode Conference, San Jose, October 17th, 2007.

Guo, She-Sen
2003 Learning from Software Localization. British Journal of Educational Technology 34 (3): 372-374.

Hall, Edward T., and Mildred Reed Hall
1987 Hidden Differences: Doing Business with the Japanese. Garden City, New York: Anchor Press/Doubleday.

Hardin, Garrett
1968 The Tragedy of the Commons. Science, December 13

Herbsleb, James D. and Deependra Moitra
2001 Global Software Development. IEEE Software, March/April: 16-20.

IBM
2013 IBM Globalization Services Stable URL: http://www-01.ibm.com/software/globalization/services/index.html accessed October 17th, 2013

Jewish Virtual Library
2013 Vital Statistics: Jewish Population in the United States, Nationally, 1654 to Present. Stable URL: http://www.jewishvirtuallibrary.org/jsource/US-Israel/usjewpop1.html accessed August 18th, 2013

Hillman, Dave
2009 India has now joined the world's depressed newsprint market. http://www.glgroup.com/News/India-has-now-joined-the-worlds-depressed-newsprint-market-32869.html, accessed December 8th, 2010

Ishida, Richard
2010 Extending Bidi Support on the Web. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20th, 2010.

Jantunen, Sami, Kari Smolander, and Donald C. Gause
2007 How Internationalization of a Product Changes Requirements Engineering Activities: An Exploratory Study. Presented at the 15th IEEE International Requirements Engineering Conference. Stable URL: http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4384179&tag=1

Kaneko, Atsushi
1999 Software Localization on Windows. Computing Japan 6, no. 5, May: 17.

Kaplan, Michael
2010 The Past, Present, and Future of Tamil in Unicode. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20th, 2010.

Kay, Russell
1994 State of the Art Developing Software Overseas. BYTE, June 1: 90.

Khalifa, Mohamed
2011 International Marketing Mistakes Related to Culture. Master's presentation. ESLSCA Business School, Egypt. Stable URL: http://www.slideshare.net/levi22usa/international-marketing-mistakes-related-to-culture accessed May 12th, 2013

Kuivalainen, Olli, Jani Lindqvist, Sami Saarenketo, Toivo Äijö
2006 International Growth of Finnish Software Firms: Starting Points, Pathways, and Outcomes. Journal of Euromarketing, 16: 7-22.

Lanier, Clinton R.
2005 Linux and the Appeal to Cultural Values. IEEE Technology and Society Magazine, Winter: 12-17 & 42.

Metiu, Anca and Bruce Kogut
2004 Distributed Knowledge and Creativity in the International Software Industry. MIR: Management International Review, Vol. 44, No. 3. Innovation and Internationalization.

Microsoft
2013 Description of the international features and the multilingual features in Word 2000. http://support.microsoft.com/kb/212400 accessed October 17th, 2013

Microsoft
2013 Globalization Step-by-Step. MSDN online reference. Stable URL: http://msdn.microsoft.com/en-us/goglobal/bb688153 accessed October 17th, 2013

Microsoft
2013 OFF98: Not All Languages Listed in Language Dialog Box are Available. http://support.microsoft.com/kb/181965/EN-US accessed October 17th, 2013

Microsoft
2013 Proofing tools that are available for each language. http://office.microsoft.com/en-us/mac-word-help/proofing-tools-that-are-available-for-each-language-HA102927381.aspx accessed October 17th, 2013

Microsoft
2013 Word features for East Asian languages. http://office.microsoft.com/en-us/word-help/word-features-for-east-asian-languages-HP005258566.aspx accessed November 1st, 2013

Microsoft
2013 Word features for right-to-left languages. http://office.microsoft.com/en-us/word-help/word-features-for-right-to-left-languages-HP005258567.aspx accessed November 1st, 2013

Milo, Thomas
2012 Arabic Typography. Presented at the 36th Internationalization and Unicode Conference, Santa Clara, California, October 23rd, 2012.

Milo, Thomas
2011 Arabic vs. Eurobic – Part I: The Role of Dutch Arabic Typography in Middle Eastern Printing. Presented at ATypE, Reykjevik, Iceland, September 16th, 2011. Stable URL: http://www.youtube.com/watch?v=1lJsfUQ-qqw

Milo, Thomas
2005 A Model for Handling the Arabic Script. Presented at 27th Internationalization and Unicode Conference, Berlin, Germany, April, 2005. Stable URL: http://www.academia.edu/2448322/A_model_for_Handling_the_Arabic_Script_2005_

Milo, Thomas
2002 Arabic Script and Typography: A Brief Historical Overview. Stable URL: http://www.academia.edu/910536/Arabic_script_and_typography_A_brief_historical_overview_2002_

Moeran, Brian
2005 The Business of Ethnography: Strategic Exchanges, People and Organizations. New York: Berg.

Mockus, Audris and James Herbsleb
2001 Challenges of Global Software Development. Software Metrics Symposium, 2001. 182-184

Mohr, David
2006 CE Enablement Testing. Presented at the 2006 Adobe Technology Summit, San Jose, California, February 10th, 2006.

Mohr, David
2012 How to access Arabic and Hebrew features in Photoshop CS6. YouTube video with stable URL: http://www.youtube.com/watch?v=gfuw7m5mVTk

Nakakoji, Kumiyo
1994 State of the Art Crossing the Cultural Boundary. BYTE, June 1: 107.

National Virtual Translation Center
2007 Language Learning Difficulty for English Speakers. Stable URL: http://web.archive.org/web/20071014005901/http://www.nvtc.gov/lotw/months/november/learningExpectations.html accessed August 18th, 2013

New World Encyclopedia
2013 Urdu. Stable URL: http://www.newworldencyclopedia.org/entry/Urdu accessed October 19th, 2013

Nogueira, Danilo and Semolini, Kelli
2010 How to Drive Your Translators Crazy without Really Trying. Translations Journal 14(4).

Nydell, Margaret K.
2006 Understanding Arabs: A Guide for Modern Times, 4th Ed. Boston: Intercultural Press, Inc.

OpenOffice
2013 Globalisation in Software Design. Apache Software Foundation. Stable URL: http://www.openoffice.org/specs/collaterals/guides/I18n_in_Software.html accessed August 18th, 2013

PewsResearch
2011 The Future of the Global Muslim Population. Stable URL: http://features.pewforum.org/muslim-population-graphic/#/United%20States accessed August 18th, 2013

Phillips, Addison
2013 How to Spot a Flying Unicorn: Analyzing User Interface Designs for Localizability. Presented at the 37th Internationalization and Unicode Conference, Santa Clara, California, October 22nd, 2013.

Purvis, Martin, Peter Hwang, Maryam Purvis, Nazim Madhavji, and Stephan Cranefield
2001 A Practical Look at Software Internationalization. Journal of Design and Integrated Process Science 5(3): 79-90.

Rafii, Farshad and Sam Perkins
1995 Internationalizating Software with Concurrent Engineering. IEEE Software, September: 38-46.

Reddy, Deepa S.
2013 Ethnography is Everywhere. UX Design Newsletter, July 2013. Stable URL: http://www.humanfactors.com/downloads/jul13.asp?utm_source=newsletter&utm_medium=email&utm_campaign=jul2013 accessed August 18th, 2013

Romney, A., W. Batchelder, and S. Weller
1986 Culture as consensus: A theory of culture and informant accuracy. American Anthropologist, Vol. 88, No. 2.

Seria
2013 Cantillation. http://www.seraia.com/seraiauk/Seraia.htm accessed October 8th, 2013

Sharma, Alolita
2013 Keynote Presentation: Enabling the Next Billion Users for Wikipedia. Presented at the 37th Internationalization and Unicode Conference, Santa Clara, California, October 22nd, 2013.

Sharma, Alolita
2013 Making Agile Work for Global Software Development i18n Teams @Wikipedia. Presented at the 37th Internationalization and Unicode Conference, Santa Clara, California, October 23rd 2013.

Silverman, Anav
2012 Jews Less than 0.2% of World Population. The Jewish Press.com. September 20th, 2012. Stable URL: http://www.jewishpress.com/news/jewish-news/jews-less-than-0-2-of-world-population/2012/09/20/ accessed August 18th, 2013

Stephan, Kurt, ed.
2003 Developing International Software. Redmond, Washington: Microsoft Press.

Sunderland, Patricia L. and Rita M. Denny
2007 Doing Anthropology in Consumer Research. Walnut Creek, California: Left Coast Press Inc.

Takezaki, Noriko
1999 Localization. Computing Japan 6, no. 5, May: 17

Tchernoousko, Iouri
2011 Globalization Project Management. Presented at Monterey Institute of International Studies. April 27th, 2011

Tidwell, Jennifer
2011 Designing Interfaces. Second Edition. Sebastopol, California: O'Reilly Media.

Treffis
2013 http://www.trefis.com/ using search criteria ADBE, accessed October 11th, 2013

Voxy
2011 What Are The Hardest Languages To Learn? [INFOGRAPHIC] Stable URL: http://voxy.com/blog/index.php/2011/03/hardest-languages-infographic/ accessed August 18th, 2013

Welcher, Laura
2010 The Rosetta Project - Wiki of All Human Language. Presented at the 34th Internationalization and Unicode Conference, Santa Clara, California, October 20, 2010.

Whorf, Benjamin
1941 The Relation of Habitual Thought and Behavior to Language. *In* Language, Culture, and Personality: Essays in Memory of Edward Sapir. Pp. 75-93. Menasha, WI: Sapir Memorial Publication Fund.

Wikibooks

2013 Wikibooks:Language Learning Difficulty for English Speakers. Stable URL: http://en.wikibooks.org/wiki/Wikibooks:Language_Learning_Difficulty_for_English_Speakers accessed August 18th, 2013

Wikipedia
2013 Mac OSX Lion. Stable URL: http://en.wikipedia.org/wiki/Mac_OS_X_Lion accessed October 18th, 2013

Wikipedia
2010 Internationalization and Localization. Stable URL: http://en.wikipedia.org/wiki/Internationalization_and_localization, accessed November 28th, 2010

Wikipedia
2010 Software Industry. Stable URL: http://en.wikipedia.org/wiki/Software_industry, accessed December 9th, 2010

Wilhelm, Robert
2013 The FreeType Project: OpenType Support. Stable URL: http://www.freetype.org/opentype/

Wilson, Susan
1998 Culture Shock: Egypt. A Guide to Customs and Etiquette. Singapore: Times Editions Pte Limited.

Windfuhr, Gernot
2009 The Iranian Languages. New York: Routledge Language Family Series.

W3Techs
2013 Usage of content languages for websites. Stable URL: http://w3techs.com/technologies/overview/content_language/all accessed August 18th, 2013

Yourdon, Edward
1994 State of the Art Developing Software Overseas. BYTE, June 1: 113.

---

[1] An executable is the core of a software program, performing some task according to programmed instructions. Unlike source files, executables cannot be read by humans as they have been compiled in machine code. On the Windows platform, executables usually have the extension .EXE, such as Word.exe or Photoshop.exe.

[2] Even something as apparently simple as converting a person's name into kanji can be very problematic. Japanese is full of homophones and every name has a detailed meaning revealed only through the glyphs which make up its kana. The name, Michiko, for instance, can be written 13 different ways, each with a unique significance. "Katakana is reserved for foreign words. Here the function of Katakana shows the cultural consciousness to draw a line between what is Japanese, and what is not…Kanji characters must be learned to differentiate between the large bodies of Japanese homophones; words that sound alike though the meanings may differ. A separate Kanji character exists for all the homophones whose meaning would be unclear if written in purely Hiragana or Katakana text…In addition, as the usage of Japanese language is more defined by the regiments of Japanese culture than by conventional reason, one must gain a basic understanding of the Japanese way of life. As such, one cannot isolate the study of the language from the confines of Japanese culture itself." (David 2006, xi)

[3] As the author points out, a "business meeting" in Japan serves a very different purpose than one in the United States. In North American culture, this is a time to discuss ideas, brainstorm, innovate, analyze, critique, bargain, and hammer out a plan. In Japanese culture, it is a time to announce and publically support a plan from management which has already been defined, usually in a series of smaller, more informal discussions, often at a restaurant or other offsite social venue. As a result, software which enhances the exchange of ideas and side-conversations among remote attendees has little usefulness in the Japanese context; such capabilities do not enhance the Japanese meaning and construction of a

"business meeting" and as a consequence, are not features which are valuable to members of this community.

[4] The French government fines any and all violations. All software sold in France must be completely translated. This legal requirement does not apply, however, to other French-speaking locales, such as Belgium or Switzerland.

[5] For example, in Arabic, the word "he wrote" is كَتَبَ (pronounced "kataba") while "it was written" is كُتِبَ (pronounced "kutiba"). The only differentiation between the known and unknown past (what we in English call the active and passive voice) is the through the changes to the short vowels, written in red and not usually included in non-religious text except to explicitly avoid confusion. And yet plugging this word with or without vowel markers into an online Arabic dictionary gives "books" for both forms (pronounced "kutub"). It is all about context and the diacritics; modern software has not reached the level of technological sophistication to correctly spell or even define Arabic words automatically and reliably. The best we can hope for is limited pattern recognition (Eldawy 2010). Meanwhile, to illustrate the difficulty in capturing these words and their diacritical marks, it is worth noting that the Arabic words in this footnote are actually graphical images edited in Photoshop and pasted into the document; I am unaware of any non-specialized, commercially-available word-processing application which can handle this simple and much needed task in Arabic or Hebrew.

[6] Google did attempt to create an exhaustive, Arabic dictionary along the lines of the Western paradigm. The project, code named Tashkeel, was presented to the public at the 2010 Unicode Conference in Santa Clara after years of work. A team of software engineers and linguistic experts, led by Mohamed Eldway, scrubbed the web for Arabic references and constructed an elaborate database and related algorithms to determine vowel type and location based on context and usage. The design seemed promising and would solve many rudimentary problems, from simple machine translations for non-speakers to allowing computers to give vocal approximations for Arabic words, as can be obtained for nearly all other languages (Eldway 2010). At the 2012 Unicode conference, it was unofficially announced that Google had shelved Tashkeel due to technical obstacles.

[7] A common failing of globalization engineering during the late 1990s and early 2000s was to view globalization as simply another feature which could be added just before shipping, rather than a fundamental aspect of the entire software design process. While localization needs to be done near the end of the development process, internationalization must be handled from the very outset (Collins 2002, Kaneko 1999, Dohler 1997, Rafii 1995, Kay 1994, Nakakoji 1994, Cunningham 1993).

[8] My first experience working with localization and internationalization in the late 1990s had such problems. While progressive in design, the PhotoDeluxe team encountered numerous, serious design flaws in the product which resulted in a long and costly delay-to-market.

[9] The Middle East and North Africa (MENA) would be one example while India another. In the MENA region, many nations are former French colonies, but in the last generation or so have strongly turned toward English in the professional sphere. As a result, outside of Arabic usage, there is an age-based linguistic divide among users with regard to supplementary support materials. To compound the problem, as in India, usage of a European language, especially English, in the context of a high-tech field has an in-culture connotation of greater sophistication or expertise. As a result, Help and training documentation in the MENA region must be in Arabic, Hebrew, English, and French while in South Asia it must be in English, Hindi, and the local language, such as Malayalam, Tamil, Bengali, Urdu, etc.

[10] The User Experience team (XD) is responsible for a product's workflow and software interface with focuses on the various aspects of human-computer interaction.

[11] At that time, as today, the importance of different languages and their markets are evaluated based on a number of criteria. Prior to the implementation of simultaneous releases, new versions of the product were rolled out in order of these various tiers. Our most important markets are referred to as Tier1 and include (in order): English, Japanese, French, and German; only bugs which affect these languages can hold up shipping the product. Tier2 languages are less important that Tier1 and roughly of equal priority to one another; they are: Danish, Dutch, Finnish, Italian, Norwegian, Portuguese, Spanish, and Swedish. Tier3 languages are less important still, including Simplified (Mainland) Chinese, Traditions (Taiwanese) Chinese, and Korean. Tier4 languages, introduced when Eastern European markets were embraced include: Czech, Greek, Hungarian, Polish, Romanian, Russian, Turkish, and Ukrainian. This project marks the inclusion of Tier5, Hebrew and Arabic, which is broken into a generic (English-focused) and North African (French-focused) version, although at this time, the product is not localized into the Tier5 languages.

[12] I cannot speak for other teams at Adobe or elsewhere, but on the Photoshop team, each engineer, whether developer or quality, has at least one designated backup for each area of responsibility for the product. The backup will often attend meetings and may actively contribute, but generally it is the primary individual who does the majority of work as well as takes responsibility for the final result.

[13] In many ways, Photoshop and Illustrator are sister products. In addition to being two of the oldest applications Adobe offers, the two are flip sides of the same tool. Photoshop operates on raster files and Illustrator operates on vector files.

[14] These would include Save For Web, Zoomify, KEVLAR, Picture Package, Contact Sheet, and Kular as well as brand new features such as Behance.

[15] Apple is well-known in the industry to make similar decisions. Not only do they lack even an internationalization discussion group on their user forums (https://discussions.apple.com/), but any internet search of internationalization deficiencies in Apple products will return many examples, such as the business locator, driving directions, and maps do not work on iPhones outside the US, even though the Western world has been digitally canvased for some time. The English version shipped on time, but the international version is still being developed (Etherington 2011).

[16] This is difficult to cite as Apple regularly updates their web-content, removing references to previous releases once a new version is available to the public. Given the work the MENA-enablement entailed, however, I was aware of these release dates as well as Arabic and Hebrew support on both platforms. The best reference I can offer for this is Wikipedia's article on OS 10.7 at http://en.wikipedia.org/wiki/Mac_OS_X_Lion

[17] As with Apple, citation for Microsoft's products is also problematic because historical information is usually replaced with data on the current release. In this case, however, Microsoft does maintain some of their earlier content. For details, see the entries for Microsoft's "OFF98: Not All Languages Listed in Language Dialog Box are Available," "Description of the international features and the multilingual features in Word 2000," and "Proofing tools that are available for each language".

[18] That is resolved in some terminal fashion, either as fixed or deferred. Bugs which were implemented in a later release are not included in this count.

[19] The current backlog for my sub-team would probably take two or three years to implement if that alone were all we did.

[20] This is based on a brief survey I did of all 704 web-generated bug reports from November 29th, 2010 to November 10th, 2009. 106 of these reports originated from a non-English-speaking internet-domain (outside of .com, .org, .biz, .us, .ca, .uk, .au, .nz, etc.) and yet were in English, the vast majority detailing issues in the localized application. 64 reports were in a language other than English, with three

appearing to be submitted without sufficient language skill to understand the required fields. A large percentage of non-English bugs were in German (25), with only Spanish (10), French (9), and Russian (8) having more than a handful. There were no bugs reported in Japanese, Chinese, or Korean, although one bug was reported from a Chinese-language source, although in English.

[21] Localized versions of software are generally much more expensive to purchase than English versions. Typically, the localized product costs between 50% and 95% more. With the release of the Creative Cloud suite this year, Adobe has brought the cost of all versions in line, but this happened nearly two years after the product described in this report.

[22] This event actually triggered a shift in international bug awareness. Starting after this development cycle, the Photoshop team has begun doing regular beta-test programs of our Tier1 languages (Japanese, German, and French) in order to capture such embarrassing bugs before the public does. I am unaware of any other team that makes this effort outside the Japanese market.

[23] As noted in #20, the number of complaints from German-speaking users is nearly as great as the combined number from Spanish, French, and Russian users, who make up the next three largest populations of user bugs. While the German-speaking market is one of Adobe's Tier1 markets and therefore particularly large, the sales of German and French versions are comparable, implying that the nearly three-fold increase in number of reported bugs is not due to the number of customers in this market, but other factors.

[24] Apple often sets trends in UI design which other companies follow. In 2009, the Adobe corporate user experience team (XD) mandated adopting the capitalization paradigm of Apple for all of Adobe products. Only the Photoshop team, one with embedded globalization expertise, immediately and very vociferously protested to this new "standard." After a series of very political battles and a unanimous agreement between the development team and the globalization team, the product team lobbied for and managed to drop this formatting for their interface. Only one other team did the same, and only after Photoshop already received corporate approval, and without explicitly stating to the user experience team in advance that they would – in short After Effects just did not comply at the 11th hour. Other Adobe applications have subsequently also dropped this mandate.

[25] Actually, although words and sentences flow right-to-left in each of these languages, technically, they are both bi-directional because numbers are written the same as in English, that is, left-to-right. While this might sound like a trivial point, it is actually a huge headache; Microsoft, in fact, recently unveiled a new bi-directional algorithm precisely because the existing model is, to quote Adil Allawi, "really an academic solution to a real-world problem that has never met user needs" which will continue to suffer from design flaws because it fails to capture the dynamism of a living language and the evolving, technologically-focused social conventions this entails (October 24th, 2012).

[26] For example, the Arabic letter baa carries the same sound as "b" in English, but has four forms. ﺑ at the beginning of a word, such as باب (/bāb/ or"door"), ﺒ in the middle of a word, as in مرحبا (/marḥaba/ or "howdy"), ﺐ at the end of a word, such as كلب (/kalb/ or "dog"), and finally ﺏ when isolated, either because it stands alone, or more frequently, because it cannot be connected to the preceding character due to the language's typographic logic, as in باب, where the final baa cannot be linked to the alif (long a).

[27] While all the documentation from that time refers to these as Central European languages, we subsequently corrected this to Eastern European languages in later materials. Please keep this in mind when reviewing the terminology.

[28] Those languages were Russian, Polish, Czech, Hungarian, Turkish, and Greek. Ukrainian and Romanian were added later.

[29] Cyrillic, the alphabet used to write Russian as well as Ukrainian, Bulgarian, and Serbian, among others, and Greek.

[30] FSI, the Foreign Service Institute of the US State Department has compiled a list of world languages, grouped by how difficult they are for an English-speaker to master. Hebrew is in category 2, along with Russian, Turkish, and Greek, requiring approximately 1100 classroom hours to achieve proficiency. Arabic is in category 3, which is reserved for it, Japanese, Korean, and various dialects of Chinese, requiring 2200 classroom hours for the same level of ability. (Effective Language Learning 2013, Wikibooks 2013, Voxy 2011, National Virtual Translation Center 2007)

[31] My managers have certainly been pleased with my work. Moreover, I have been called upon on multiple occasions (Tech Summit, Inter-team Loan, guest lecturer at MIIS) to shared what I have learned on the topic with others.

[32] As I write this project report, I am currently working on the features and bugs in post-CS7 Photoshop. Much of my current work is language-agnostic and yet this tension between how users see things vs. how engineers believe users see things remains an endemic struggle and source of design failures, as seen in the (lack of) adoption of type styles among web designers as well as conflicts in workflow philosophy and models for who transformed text should be handled.

[33] The world of Internationalization is small and most of the long-term players know one another by name, face, reputation, history, and skill set. In order to protect their identities, I have intentionally misrepresented my informants as much as possible without altering relevant characteristics so as to obscure their identities and protect their anonymity. When I use both first and last names, however, then I am referring to real names.

[34] GB18030 is a Chinese government standard for language support which includes not just Mandarin or Simplified Chinese but also Traditional or Taiwanese as well as the writing systems of various ethnic minorities of China, such as Mongolian, Tibetan, and traditional Uyghur scripts as well. Today, software companies wanting to do business in China must meet the requirements of this standard in order to sell their product.

[35] Until Adobe hired our California-based, Egyptian-born Project Manager, however, to the best of my knowledge and despite repeated attempts, I could not find a single Arabic-speaker anywhere in the company. This is both in terms of the MENA project or working on something else. I did work somewhat with a Farsi-speaking engineering manager, but she had no direct involvement in this effort and other than teaching me a few Persian pleasantries and a bit of grammar, she was not involved. It was not until the enablement work was already underway that I even began to have dealings with the Arabic-speaking contractors in Tunisia, Syria, and much later, India.

[36] Due to technical limitations in Microsoft Word, I cannot adjust the diacritical marks in my second example. Trust me that while I have chosen to use the Tahoma font in this report due to its support of relatively clear English, Arabic, and Hebrew characters, the implementation of diacritics in Arabic for this font leaves a bit to be desired and was the subject of many bug reports and user complaints. The following example looks better because it uses Adobe Arabic, a font designed for this language, but the text needs to be expanded to nearly twice the point size in order to see the missing detail: أهلاً و سهلاً It is issues such as this which Arabic typographers have to deal with, much of which are based on technical limitations coupled with limited choices and non-ideal implementations. Such deficiencies would never be tolerated in English, Japanese, or Chinese, but as a relatively new and unknown market, Arabic typography has many such examples.

[37] I do not mean to imply that Persian and Arabic culture completely overlap or that Muslim and Arabic culture do either. Each, however, provide insight into the collective whole.

[38] Five letters, the Kaf, Mem, Nun, Pe, and Tsadi, have different final forms or sofit when they occur at the end of a word. These are כ vs ך, מ vs ם, נ vs ן, פ vs ף, and צ vs ץ.

[39] Numbers for Arabic populations vary greatly, depending on source, sampling method, and potential political motivation. Some sources, such as the CIA Fact Book, place the total number as low as 250 million native speakers, even though the population of the countries in the Arab League alone is greater than 360 million. Conservative estimates, however, place the number of Arabic speakers between at least 350 and 450 million. That said, theoretically all of Earth's Muslims are expected to have at least a rudimentary command of the language. Since there are more than 1.6 billion Muslims, this would be approximately 23% of the human population.

[40] To be clear, both Arabic and Hebrew use diacritics to represent vowels, but Hebrew also uses diacritics to differentiate certain letters, such as שׂ and שׁ, which is the difference between "s" and "sh." Even though some may consider these distinct letters linguistically, the standard method by which these characters are typed using a keyboard defines them as one character with or without an added diacritical mark. In contrast, the analogous Arabic س and ش are distinct keys on the keyboard; no diacritical mark is needed and no complex user workflow is required.

[41] Of course, this assumes there was never a concerted human effort to standardize the language into its present logical and algebraic format. Such an effort appears to be supported by history and is worthy of further research, but it is also quite outside the scope of this project-report.

[42] This attitude remains even though there are words of Greek and Persian origin in the Qur'an as well.

[43] التمويه الضبابي is the accepted term in Arabic. It is made up of two words التمويه (camouflage or disguise) and ضبابي (blur). Similar examples include "de-interlace" (تشابك) which actually translates to "entangle" or "snarl" as well as "matting" (مادة لصنع الحصر) which translates to "article of the limited making."

[44] We discovered Russian and other users passionately support and defend this digital ecosystem when it quit working in Photoshop CS6 due to an architectural change on the part of Apple. Although we did not cause the problem, Adobe's user forums came alive with Russian users complaining that Photoshop was no longer following these undocumented "standards."

[45] An earlier study shows that the among of English content on the web today is only ~1% less than in 2002, when German was the second most used content-language at 7.7% (now 5.6%) and Arabic was so rare as to not be considered in the survey. Today, Russian is in the position of second most used content-language at 6.3%, up from 1.7% (Ebbertz 2002)

[46] One (admittedly simplistic) example was to compare the cultural ambivalence of Arabic users toward a localized product with the cultural ambivalence of Indian users toward the same. Both have the same in-culture perception of English language usage being more technical and modern. This was helpful when talking with one of the engineers on the team with a deep and personal background on the Subcontinent. Another example I used to make the case was to compare the transliteration of English terms into Arabic with the transliteration of such terms into Japanese hiragana. As anyone familiar with the language will tell you, foreign words MUST always be written in katakana. Arabic lacks this kind of distinct writing system for technical, legal, and foreign words and generally prefers to either use the original, untranslated word or a domestic equivalent. Since there are very few such words in the domain of digital imagery, that only really leaves English for such terms. A third example I used was to compare the translated user interface in Arabic with the attitudes of educated Scandinavian consumers toward translating English-language movies. While it is done for children's shows and films on TV, generally there is a perception that something is lost in the translation, resulting in a reduced experience. As a

result, those who can understand English (or French or German, etc.) usually prefer to see the film in the original version. This point was particularly useful given that Photoshop is understood as a professional-level application, suitable mainly for those with a specific, educated background rather than the general public.

[47] I actually regularly work with authors in foreign languages as well as our third-party translators to help clarify the meaning of various terms, instructions, explanations, and concepts. Highly specialized product areas, such as 3D modeling and High Definition Range (HDR) imagery, are among the most problematic, so much so that some non-English users actually prefer to use English terms whenever possible. Along these lines, I have worked with several well-known, native-language, How To writers in German, Swedish, French, and Spanish who all use the English language version of Photoshop as much as possible, even when writing their native-language books, but occasionally must resort to the localized version of Help for clarification. These non-Adobe colleagues then inform me of mistranslations or non-parallel explanations (or even just bad grammar) when they encounter it so I can have it corrected in the next iteration.

[48] To support North Africa means to support several countries which were former French colonies. In these nations it is normative to use French in lieu of English as the technical language of choice. This is a topic which did lead to further complications, as I will detail later.

[49] Except as noted, all Arabic in this report is Modern Standard Arabic (MSA) using the transliteration method from the library of Congress.

[50] We will return to this concept later.

[51] While this feature is important to Arabic typographers, the ability to add taṭwīl is not supported well or at all in many word processors. The text in this example is actually a graphic imported from Photoshop.

[52] The البيت (al-bayt or “the home”), for instance, should not have the leading alif (ا or long ā sound) separated from the rest of the word in the third example from the left.

[53] The type engine is the software that lays out how letters are grouped in a run of text. It determines spacing of characters and words, flow of text from line to line, which Open Type attributes can be applied, etc.

[54] In the Arabic-speaking world, there are people whose job it is to correctly add all of the vowel marks to formal texts before the work is read aloud. It is grammatically technical and rather specialized work, much like that of a professional speech-writer, para-legal, or medical proof-reader. It is not a skill that all Arabic speakers have, since generally only television personalities speak in Modern Standard Arabic (MSA); everyone can understand this formal register, but most people speak using a local, regional dialect.

[55] We will return to this idea when we discuss Hebrew, later.

[56] Very few German-speakers are familiar with what an “inch” or “Zoll” is, for instance, unless it is part of their work. The same is actually true for most French and Spanish speakers as well.

[57] At least as an ethnic population, there are more Arabs living in the United States (~3.5 million) than Djibouti, Bahrain, Qatar, Oman, Kuwait, or Mauritania. Europe’s Arab population is even higher, around 5 million. Given the non-Arabic-speaking, non-Muslim context most American and European Arabs live in, separate groupings for these populations seems quite logical.

[58] To completely confuse the matter, in India, there are several additional sets of digits used for specific languages such as Malayalam and Tamil. Most of the Sub-Continent, however, uses a

combination of Western digits and Devanagari digits (१,२,३,४,५,६,७,८,९,०), which are clearly different than what Arabs call Hindi or Indic digits.

[59] The Agile development model actually embraces this practice and has been adopted by my most software firms precisely because it helps deal with issues that the Waterfall method of software development does not. Although not fully implemented during the timeframe of this report, it is now used across the company today and has become a very common planning technique in the software development industry. Under the Agile model, work is divided into Sprints. Each Sprint will then have work that is required to be completed within the sprint with additional feature work or bugs that can dealt with if time remains. Everything else is kept in a one-to-n backlog for a future sprint.

[60] It takes 12 letters just to say "hi" in Russian, Здравствуйте. This is one of many such examples.

[61] This is actually quite logical as Far Eastern input methods (IMEs) are elaborate and complex, requiring several keystrokes to achieve a single character. In Japanese, for instance, to achieve the kanji (pictogram) for "I" a user would need to type "w" and "a," be offered the opportunity to change those two (English) characters into the corresponding Japanese syllabic character わ, the "t" and "a," and again the opportunity to get the hiragana syllable character た, and finally "s," "h," and "i," which elicits the option to get the hiragana syllable し, which then can be converted into the actual kanji glyph 私。

[62] I am actually a big fan of this whole series of books and have used them to both learn and brush up my Russian, Italian, and Spanish. As a fairly accomplished linguist, the series gives me the tools I need to get a basic understanding of grammar and lexicon. In this case, I was looking for how letters are used, in particular.

[63] I fully realize that the idea of any language having a pure, unevolved root is the result of carefully constructed cultural narrative rather than reality, since both culture and language naturally change over time as they encounter novel events. That said, for those meticulous linguists who carefully labored to restore the dead, liturgical Hebrew to life as a vibrant, living tongue using Yiddish as a frame, Arabic was their method to restore the Semitic flavor to many words and thus recapture some aspect of what Abraham, Moses, and King David spoke.

[64] As with nearly all Arabic letters, a few Hebrew letters also have different shapes depending on where they appear in a word.

[65] There are analogous (recitation) marks in Arabic as well, used when reciting the Qur'an in tajwīd or tartīl.

[66] This is the YouTube video page which I mention below, found at http://www.youtube.com/watch?v=gfuw7m5mVTk. Due to the comments section, it has become a long-standing alternative way to reach out to Arabic- and Hebrew-speaking users.

[67] The page has received more than 25,000 hits at the time I write this and is the number one response on Google for "Arabic Photoshop CS6" or "Hebrew Photoshop CS6" while it is on the first page of responses for each when "CS6" is omitted. Of the votes on the video page, 286 are positive and only one negative, while the only negative comments deal with either unrelated crashes or why InDesign or Illustrator do not also share this universal functionality. Based on our marketing group's internal criteria for interpreting social media feedback, these results are amazing. Normally, 60% positive and 18% negative comments are typical on Facebook or YouTube, while the results for my MENA video are far over 95% positive, even using the most pessimistic interpretation. Meanwhile, in comparison to other employee-created YouTube pages, this page receives far more traffic; of the 63 YouTube videos for all aspects of 3D features, most have less than 10,000 hits and only one has a count as high as 27,000. None have anywhere near as many positive votes or comments.

[68] Never had I participated in a program with more than 100 non-English prerelease users. The thought that this number would be too small never occurred to me until near the very end of the project.

[69] One example would be Intuit's redesign of Quicken for German customers. The other would be Adobe's redesign of InDesign for Japan. Although the work on InDesign was done more than a decade ago, none of these additional tools have been incorporated in any other versions of the product and continue to be supported through a series of additional modules which are incompatible with the core application.