# Project Manager's role in localization projects – Localizing SSH Secure Shell

Henri Rantanen
Pro Gradu Thesis
University of Tampere
School of Modern Languages and Translation Studies
Translation Studies (English)
May 2007

TIIVISTELMÄ

Tampereen yliopisto
Kieli- ja käännöstieteiden laitos

RANTANEN, HENRI: Project Manager's role in localization projects – Localizing SSH Secure Shell
Pro gradu -tutkielma, 101 sivua, suomenkielinen lyhennelmä 10 sivua.
Käännöstiede (englanti)
Toukokuu 2007

---

Tämän tutkielman tarkoitus on antaa kuvaus projektipäällikön työstä lokalisointiprojektissa ensin teorian tasolla ja verrata sitten teoriaa käytäntöön eli todelliseen lokalisointiprojektiin, jossa SSH Secure Shell -asiakassovellus lokalisoitiin ensimmäistä kertaa kolmelle kielelle.

Teoriapohjan projektipäällikön työlle antavat pääosin Bert Esselinkin vuonna 2000 julkaistu lokalisointiprojekteja käsittelevä A Practical Guide to Localization -kirja ja Scott Berkunin vuonna 2005 julkaistu ohjelmointiprojekteja käsittelevä The Art of Project Management -kirja. Esselink käsittelee kirjassaan lokalisointiprojektin eri osia ja työvaiheita tehtävä- ja roolipohjaisen lähestymistavan avulla, kun Berkun puolestaan keskittyy huomattavasti painokkaammin luoviin ratkaisuihin, johtamiseen ja kommunikointiin projektin liittyvien ihmisten kanssa.

Tutkielmassa kerrotaan lokalisoinnin ulkoistamisen hyvät ja huonot puolet, kuvataan lokalisointiprosessi lokalisoinnin tekevän yrityksen ja työn toimeksi antavan asiakkaan näkökulmista sekä kerrotaan SSH Secure Shell -asiakassovelluksen lokalisointiprosessin vaiheista alusta loppuun. Siinä kerrotaan prosessin hyvin toimineet vaiheet sekä annetaan esimerkkejä virheistä ja asioista, jotka projektipäällikkö joutui oppimaan kantapään kautta. Tutkielmassa myös kuvataan, millainen lokalisointiprojektin tulisi olla, jotta virheet minimoitaisiin.

Lisäksi tutkielmassa todetaan, että projektipäällikön tulisi tuntea ja osata tehdä lokalisointiprojektin jokaiseen vaiheeseen liittyvät toimet, jotta projekti olisi mahdollisimman sujuva. Hänen täytyy pystyä suunnittelemaan projekti ja viestimään erilaisten ihmisten kanssa. Jokainen uudessa yrityksessä tai ympäristössä käynnistetty lokalisointiprojekti sisältää erilaisia ongelmia, joiden ennakointi on erittäin vaikeaa tai jopa mahdotonta. Tämän vuoksi projektipäällikön on oltava valmis näkemään "ylimääräistä" vaivaa, ja hänellä on oltava erinomaiset ongelmanratkaisutaidot niin teknisten kuin ihmislähtöistenkin ongelmien ratkaisemista varten. Tutkielmassa todetaan myös, että projektin ja ihmisten johtamisen lisäksi projektipäällikön voidaan nähdä toimivan yrityksen eri kulttuurien–niin yrityksissä työskentelevien ihmisten kuin yrityksen osastojen tai toimintojen kulttuurienkin–välisenä tulkkina, joka kommunikoinnillaan ja toimillaan tekee projektista sujuvan ja auttaa sen jäseniä työskentelemään yhteisen päämäärän hyväksi parhaalla mahdollisella tavalla.

AVAINSANAT: lokalisointi, projekti, projektinhallinta, projektipäällikkö, johtaminen

## TABLE OF CONTENTS

# 1. Introduction

The role of a project manager in localization projects is by no means easy, and the tasks connected with it can vary greatly in different projects, as project sizes and types differ from each other, and also in different organizations. What makes it especially interesting, however, is that it can combine different roles as well. Globalization and localization expert Bert Esselink (2000) has authored a popular localization guide, and he has a very task-oriented approach to localization projects' processes and management. Scott Berkun (2005), a project management and product design consultant and a teacher of creative thinking at the University of Washington, on the other hand, stresses in his account on project management the human factor and relationships between people in the project teams. These form the main theoretical source of my thesis.

This thesis has a strong practical emphasis, and in it I examine how these views and descriptions of the task of project management and the role of a project manager work in a real-life software localization project. The viewpoint I am using of that of a localization manager working in an IT company for the company's very first localization project. I also describe an ideal localization process for the company in question based on the lessons learned in the first project of the company. When discussing working with localization agencies or companies, I will give two different viewpoints myself – from "both sides of the fence", the software company's and the localization company's – on various issues, processes, and the ways of thinking.

The project case study part of this thesis contains some detailed process and problem descriptions, which serve as examples on unforeseen issues a project manager might encounter when taking on a new localization project.

First I need to say a few words about my background to give the reader an overview of what I base my opinions on, i.e. where my practical experience comes from. I worked for roughly five years for a Finnish office of a large international localization company. My tasks in the localization company included team leading, project management, translating, technical proofreading, technical validation, testing, Desktop Publishing, compiling, building, and quality assurance. After that, I worked for SSH Communications Security as Localization Manager/Coordinator for about four years, launching their first localization projects and processes. I have also been doing freelance translation work for more than five years both while working for SSH and since leaving the company. In this thesis, I will build on this practical experience.

## 1.1 Globalization, internationalization and localization

When discussing software localization there are three main concepts that need some clarification: *globalization, internationalization,* and *localization*. In addition to those, *translation* – where the text is translated into another language without much modification as such – could also be added here, since my personal experience shows that all three concepts have been confused with translation and the other way around. According to LISA, the Localization Industry Standards Association (2007), the concepts are defined as follows:

Globalization addresses all of the business issues associated with making an organization truly global. For the globalization of products and services this involves integrating all of the internal and external business functions with marketing, sales, and customer support in the world market.

Internationalization is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for redesign. Internationalization takes place at the level of program design and document development.

Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold.

In this context, as Lingo Systems (2000: 4) defines it, the term translation is used of the “process of actually converting the written word of a source language into the written word of a target language. Translation is a crucial component of localization.” Translation requires that the full meaning of the source material is accurately rendered into the target language, with special attention paid to cultural nuance and style. The difference between translation and localization is that translation is only one of the activities in localization. In addition to translation, a localization project includes many other tasks such as project management, software engineering, testing, and desktop publishing. In localization, a stronger emphasis is placed on translation tools and technology than in the traditional translation industry. (Esselink 2000: 4)

Localization differs from translation mainly in scope. Localization, as defined by for example Esselink (2000), includes translation of, for example, manuals, user interface, help texts and error messages, but product names may have to be changed to avoid unfortunate associations in the target language. However, the process also requires some other skills besides just translation. Software dialog boxes and user interface text field lengths may have to be altered, date, time and currency formats changed, and so on. In the case of bi-directional target languages (such as Arabic and Hebrew) and double-byte character sets (such as those for Chinese, Japanese, and Korean), some programming is most likely required to ensure that localized text is displayed correctly on the target platforms. On the content side, programs often have to be changed to conform to national and cultural norms and standards. In multimedia applications the color, size, and shape of objects such as coins and notes, telephones, mailboxes, taxi cabs, buses, and ambulances, traditionally vary from country to country. Vehicles may have to drive on the other side of the road, dress codes will vary, and symbols may take on a new significance. Similarly, mainstream business applications, such as address databases and financial accounting packages, have to be adapted to the procedures and conventions applicable in their new environments.

Multiple-language Web content and e-Business sites increase the complexity of enterprise globalization as well as the localization process of products and services.

Internationalization, on the other hand, needs to be carried out before localization. It is the process of designing and creating a product which is culturally and technically as neutral as possible, and which can therefore easily be localized into a specific target culture or cultures. This reduces the time and resources required for the localization process, which saves money and improves time-to-market. Internationalization has reached the point where major software publishers can release 30 or more different localized versions at the same as the original version. (LISA 2007)

The Localization Industry Standards Association's Web site also mentions that often the product names are also localized. While this may happen when localizing more tangible consumer products which are aimed at a larger target group, such as books or food products, this is very seldom the case, for example, in business software localization. When localizing games for young children or other kinds of “low-level” multimedia products, it is often sensible to translate the name of the product. This is because the target audience expects an experience where they can adapt to the atmosphere easily. With regard to this type of products, the localization process often contains a lot of adaptation, for example in graphics, voices or the general content. For business software or applications, however, the names are often left in English, partly due to international communication where people from different countries and cultures often use and discuss the same tools and applications, and partly due to the trademarks and copyrights involved. This also gives the products a more professional and international feel. This is especially true when localizing some brand applications which are used all around the world, such as Microsoft’s Windows, Corel’s WordPerfect, and Autodesk’s AutoCAD.

Another concept which is confusing to some is translation tools, such as Translation Memory applications, which are often confused with machine translation. Machine translation development started already back in the 1950s, and it aims to undertake the whole translation process, but all current commercial and operational systems produce output which must be edited in order to attain publishable quality. Computer-based translation systems are not rivals to human translators, but they help them increase productivity in technical translation. Computer aids for translators, such as Translation Memory applications, which are a result of machine translation application development, in turn support professional translators. Translator workstations combine access to dictionaries and terminological databanks, multilingual word processing, management of glossaries and terminology resources, and above all they include a translation memory facility, which enables translators to create, align, store and access existing translations for later reuse or revision or as sources for translations. (Hutchins 2001)

## 1.2 Background of localization, needs and reasons for

As for example Layden (1997) argues, localization aims at providing a wider range of users with a product. This increases usability in the target market and, as experience has shown, usually the most important reason for the software companies – sales and profit.

According to Freivalds (2002: 14), Franz Rau, a top executive at Microsoft, predicted in 1997 that by 2005 most US companies would be localizing their documentation in 80 languages, up from the 20 most commonly used in 1997. Many believed that the number of languages localized would diminish rather than increase, since increased globalization would lead to wider use of English. However, the opposite has happened, as publishers are trying to capture new markets, and some countries have even laws to prevent the overuse of English in technical documentation.

For more usability-oriented reasons than just preventing overuse of English, in some countries there are laws which state that an end user product has to have localized instructions of use or that the whole product needs to use an official language of the country before the product can even be sold. This is to protect the consumer both financially and health-wise. Usually there is a minimum required set of instructions which needs to be in the language of the target user, for example the installation instructions or the safety procedures. This applies for example to Finland. (Kuluttajansuojalaki, Chapter 5, 12a §)

As Cho (2002) describes the situation in 2002, "Globalization and the growing complexities of localization, in both technical and non-technical spheres, have driven the growth of the localization industry. Estimates of the size of the industry range in from \$11 to \$30 billion, with strong returns on investment for localizing companies. The 20 largest IT companies annually leverage total localization expenditures of around \$1.5 billion to generate sales in excess of \$50 billion."

However, all too many companies still do not know what they are spending or making on localization, largely due to the invisibility and lack of transparency of the localization process within organizations. Additionally, these figures do not take into account the localization of websites into multiple languages, which is increasing exponentially as the number of non-English speaking Internet users increases. (LISA 2007)

When a software company targets the Asian markets, such as the company in the case study section of this thesis, their programs must be double-byte enabled, which is another internationalization process. The term “double-byte” means that most of the Asian characters need to be specified by two bytes per character in computer operating systems instead of just one as in the Western character set. When a product needs to be localized for Asian cultures, double-byte enabling is a process in which the product code is manipulated so that the program will be able to read and handle Asian characters. After the program supports double-byte, it may be localized many times which can provide greater savings in the long run. (Layden 1997)

# 2. Project management and the role of a project manager

Project management is a part of every project. While some projects can be very small and the only person managing them is the one doing them, there is still management involved in any process with tasks in it. It is clear that projects in different fields and areas can differ a lot from each other and that not even all localization projects are the same or contain the same workflow items. I will give first a generalized overview of the project management tasks as seen by Esselink (2000) and the role of a project manager in localization projects and then move on to some of Berkun's (2005) views on the human factor in project management.

Project managers are key resources in any localization project and serve as the central point of communication for the people taking part in a localization project. It is usually the project manager's responsibility to schedule and monitor all components and activities which the project consists of. Depending on the number and size of the projects, project managers can be dedicated to a single client, or to one or more projects. Most large localization companies have dedicated project managers to assign to their localization projects. (Esselink 2000: 427)

The same is true for major software development companies which have a steady localization workflow of their products into different languages and locales.

## 2.1 Qualities of a project manager

As Esselink (2000: 427-428) states, project managers usually have a wide range of tasks and responsibilities, which include creating quotes and proposals for new projects, coordinating project setup (or preparing projects), planning projects, monitoring project finances, managing resources and quality assurance steps, and change management. In many smaller localization agencies senior translators work also as project managers or localization engineers. He has put together the following list of skills most localization vendors expect project managers to have as a minimum (427-428):

- General project management skills
- Experience dealing with localization or multilingual translation project
- Excellent communication, reporting, and negotiation skills
- Strong organizational skills for production tasks
- Experience with planning, budgeting, resource management, project tracking, risk management, and quality assurance
- Understanding of localization processes and benefits and limitations of translation technology
- Flexibility and adaptability
- Foreign language skills (preferred)
- Technical skills (preferred)

Esselink (2000: 428) also claims that in order to complete their tasks effectively, the localization project managers need many different software applications, including standard office, spreadsheet and Internet applications. In addition to that, he continues, the project managers will most likely need a project management application, such as Microsoft Project, standard templates for status reports, change approval forms, tracking sheets, and also a human resources database application.

## 2.2 Tasks and responsibilities of a project manager

According to Esselink (2000: 429), the key tasks of localization project managers focus on three important goals in managing a localization project, which are time, quality, and budget. In order for a project to be successful, it needs to be completed on schedule, within budget, and according to previously agreed quality standards.

Most localization projects start with pre-sales evaluation phase where clients send out a request for quotation (RFQ) or a request for proposal (RFP) to multiple localization vendors in order to get the most competitive bid. Most large localization vendors have account managers or business development managers who create quotations or proposals for new clients. Project managers often create quotes for new projects from existing clients with already established business relationships. Professional RFQs and RFPs usually contain an introduction to the project, an overview of the project components, project scope and volumes or an estimation of them, expected services and deliverables, and deadlines. Clients with well-established processes for localization will generally send out full localization kits, but the project manager needs to discuss with the client what they actually expect from a quotation or proposal. Some clients might want to find out a general cost estimate and some general company information, while others might expect a very detailed breakdown of all project costs and procedures. If there are any unclear elements in the project or process in question, the project manager needs to communicate with the client to find our for example the following: the languages is the product to be translated into, the components to be translated, other tasks which need to be performed, possible updates or milestones in the translation cycle, requirements to create glossaries, requirements to use specific translation memory tools or other tools, expected final deliveries or output, special software or hardware requirements in the localization process, need for specialized knowledge or experience of the subject matter, provided reference material and the time frame for delivering the localized product. Providing general cost estimates without going through a comprehensive project evaluation would introduce a number of financial risks for both the vendor and the client. As an industry standard, Esselink also states, project management is involved as a percentage of the total project cost, which was 10% at the time of the publishing of his book. (Esselink 2000: 427, 429-430)

As Esselink (2000: 430) claims, the project evaluation task at the quotation state of a project is often underestimated or neglected by the localization vendors because there are not enough resources allocated to the task. All too often is the task performed by people working in some actual project in progress and who concentrate less on evaluation of projects which might never take place. The project manager needs to make sure that all translatable project components have been identified and all activities necessary to localize the product have been covered.

Project evaluation should provide the project manager with the word count volumes for all components, which should include the total word count, external matches, which means the leveraged or re-used words from a translation memory, and internal matches, which means the repeated words within the same document or material that only need to be translated once. Most of this information can be extracted from a translation memory tool's analysis features. The word counts should also be checked with at least one other tool, which can for example be the application used to create the source files. Estimating engineering and testing times for the material, such as online Help, Web pages, or software is most likely the most difficult task in evaluating a localization project, since many factors can influence the time required to build localized software applications or online Help projects. Those factors include file format, level of internationalization, level of testing required and so on. Many localization vendors base their estimations on an average turn-around time for standard online Help files, and that time is usually connected with the number of words in the component. The project manger should consult an experienced software engineer on the estimated time required for engineering a project to avoid miscalculations. The task of desktop publishing should also be taken into account when figuring out the costs and schedules of a project. (Esselink 2000: 431-432)

The project manager needs to work out the price estimate for the project. The estimate needs to be clear and competitive, because most software companies have a budget reserved for localized versions of their products, and thus software companies often ask several vendors for quotes, then compare the quotations and proposals and finally select the lowest bid. The project manager should create a professional quotation which has unit prices instead of general time estimates and a clear explanation which activities are included in each unit price in order to make it as clear as possible for the potential customer. The quotation needs to be very specific and define all the tasks and activities of the project but also leave room for modifications, changes and updates. (Esselink 2000: 432-433)

Esselink (2005: 438-440) states that when a project manager is assigned a new localization project, he or she should set up the project by organizing a project evaluation for all project components, create a project folder for all material, create a project schedule and a resource plan, organize preparation for all project components, create a project budget and a quality plan, and also create a communication plan. In most cases, project evaluation is carried out during the project quotation phase. When an actual awarded project is evaluated, project managers often need to extract more information than just estimated volumes and schedules, and identify potential problems or issues. The project folder is simply a well-organized central location in a network where all team members can access the source and target material of the project, and the project schedule and resource plan are based on the project evaluation report. The resource plan is closely linked to the schedule, and the project manager should have discussions with all the production line managers and make sure all the necessary resources, including translators, engineers and QA staff, are available during the time scheduled for each task in question.

The project preparation phase involves assigning the components of a localization kit to the individuals involved in the project and giving customized instructions for each of them. The project manager needs to make sure the translators are given the necessary information, which is typically a schedule detailing translation delivery dates, instructions on deliverables and services required, the source files to be translated, a running version of the application to be translated, instructions and details on the contents of the kit, possible non-standard tools required for completing the translation, reference material such as glossaries, and procedures. It might also be necessary to prepare kits for other outsourced tasks, such as engineering, desktop publishing, or testing. The project manager needs to make sure the each package is customized for each resource and related task. (Esselink 2005: 441-442)

Esselink (442) states that a communication plan is a critical part of a localization project, since it is the only way to make sure the project managers monitor projects effectively and highlight potential problems. It should outline the responsibilities assigned to each team member, identify resources responsible for resolving questions and sending files, and also have a clear description of the escalation path for any problems. Project managers should create communication plans both for internal use and external use. The plans need to be targeted to internal and external production staff working on the project and to the client, identifying the contacts at the vendor site.

For project planning, Esselink (2005: 443) argues, the first task required of localization project managers when creating a schedule is to define all tasks and activities. Most of this information should be generated from a thorough project evaluation, but detailed scheduling can only be carried out once the actual material has been received and analyzed. A project manager should create a time schedule at the very beginning of the project, outlining the start and end dates for all the activities in the localization project. Esselink describes the activities as follows:

- Terminology setup and multilingual glossary preparation, including client review
- Project setup and preparation, evaluation of source material and extraction of translatable text
- Translation of software, online Help and documentation, including all review steps and proofreading
- Engineering and testing of software and Help, including screen capture creation
- DTP of documentation, including screen captures and output to PostScript or online documentation format
- Quality assurance and final production

The project manager also needs to make sure there is time allowed for activities such as conversion, compilation, and preparation of files. When all tasks have been defined, the project manager should evaluate the workflow dependencies, sequence of activities, the workload and duration of each activity, and resource assignments. After all required information is collected, he or she will create a concrete schedule. Project management includes also scheduling and allocating resources, including translators, engineers and desktop publishers, in order to make the most effective use of the people assigned to the project. Most localization project managers do not manage corporate human resource activities, but they assign external contractors, external agencies, and production staff to projects and roles inside the projects. (Esselink 2000: 443-449)

Esselink (2000: 448,453) states that once the project work starts, the project manager needs to monitor the process by requesting regular status reports and time sheets. If the project at some point is at risk of going off track, the project manager should take corrective action by adding alternative resources or by adjusting the schedule to avoid negative impact on the entire project. In addition to monitoring the progress, the project manager needs to monitor the quality throughout the whole process and arrange for the quality assurance checks and corrections.

"Communication is critical to the management success of a localization project", argues Esselink (2000: 458) and continues to state that project managers are communicating with customers, translators, technical staff, linguists, reviewers, and all the participants of the project. The project manager need to keep external and internal communication separate and find the right balance in the amount of information in communication and communicate the right amount of information to the right people at the right time. He or she is in most cases the point of contact for schedules, finances, quality, deliveries, and project status. The project manager should actively encourage information exchange within the project team and ensure all team members know the latest developments, changes, or updates in the project but at the same time avoid overloading people with unnecessary information. He or she should also be honest in communicating with the client and tell about delays or other project problems as soon as they occur. The project manager is also responsible for reporting the project status according to agreed reporting schedule and items to report as well as dealing with possible complaints from the client. (Esselink 2000: 458-460)

All in all, at least what comes to larger localization projects, the project manager is usually not carrying out the actual translating, engineering or quality assurance tasks by himself or herself but rather working as the driving force of the project, planning the project flow from the start to the end and making sure all tasks will be carried out, the project is profitable, and the schedule agreed with the client is followed. He or she is following the progress of the project closely, communicating with the project staff and the client, and taking corrective or preventive action whenever necessary.

While this Esselink's task-oriented approach to project management is very important and should be paid proper attention, Berkun (2005) concentrates much more on communication and inter-personal relations in project management, which are more interesting and relevant in this thesis' viewpoint. Berkun's book deals with project management in a software development process and related projects, but the following are certain “human” issues and conclusions of relevance.

## 2.3 The human factor

According to Berkun (2005: 7-8), project management can be a profession, a job, a role, or an activity. Some companies have project managers whose job is to oversee entire 200-person projects while others use the title of project manager for line-level junior managers, each responsible for a small area of a large project. Depending on organization's structure, culture, and the goals of the project, project management can be an informal role or highly defined. Sometimes the absence of a dedicated project manager works fine, but other times there will be some dysfunction. If a person whose primary job is to guide the overall effort is not present in the project, individual biases and interests can derail the directions of the team. Strong adversarial factions may develop around engineering and business roles, which slows down progress and frustrates everyone involved. The different “cultures” inside the company can cause friction and misunderstandings, which can and often will result in less than optimal effort or outcome. The project manager will understand the business view of the project as well as the technical view, and he or she will help the team understand both when necessary.

Berkun (2005: 10-12) states that project managers need to maintain a balance of attitudes, since different situations require different behavior. A project manager needs to develop instincts for which traits are appropriate at which times, which contributes to the idea of project management as an art: it requires intuition, judgment, and experience. Berkun quotes Tom Peters' essay (1991), which lists the following traits of an ideal project manager, complemented by Berkun:

- Total ego/no ego. Project managers have a high emotional investment in what they are doing, and for many, this emotional connection is what enables them to maintain the intensity needed to be effective. At the same time, however, they have to avoid placing their own interests ahead of the project. They must be willing to delegate important or fun tasks and shade the accolades and rewards with the entire team.
- Autocrat/delegator. A project manager has to be confident and willful enough to take control and force certain actions onto the team. However, the general goal should be to avoid the need for these extreme situations. A well-managed project should create an environment where work can be delegated and collaborated on effectively.
- Leader/manager. Project managers are only as good as their teammates' commitment, energy and diverse skills. So project managers must be leaders: visionaries and invigorators. On the other hand, management means being expert at the mechanics. A good project manager has a passion for inspiring others and is also very interested in the nuts and bolts of the job.
- Tolerate ambiguity/pursue perfection. Controlled ambiguity is essential for good ideas to surface, and a project manager must respect it, if not manage it. However, discipline and precision are paramount. It requires wisdom to discern when the quest for perfection is worthwhile, versus when a mediocre or quick-and-dirty solution is sufficient.
- Oral/written. Oral skills are critically important to project management, since there will always be meetings, negotiations, hallway discussions and brainstorming sessions, but the larger the organization or the project is, the more important written skills and the willingness to use them become.

- Acknowledge complexity/champion simplicity. Effective project managers must juggle a thousand balls of differing (and ever changing) shapes and sizes. On the other hand, they must be "Keep It Simple, Stupid" fanatics, making sure that a few, essential values dominate the organization.
- Impatient/patient. Most of the time, the project manager is the person pushing for action, forcing others to keep work lean and focused, but in some situations, impatience works against the project. Project managers need to know when to force an issue and when to back off and let things happen.
- Courage/fear. A project manager must have a healthy respect for all the things that can go wrong, and see them as entirely possible, but he or she needs to match this respect with the courage necessary to take on big challenges. Project managers should also not hide from failures but instead see them as opportunities to learn something.
- Believer/skeptic. A project manager needs to have confidence in the work being done and see true value in the goals that will be achieved. At the same time, there is a need for skepticism about how things are going and the ways in which they are being done, bringing difficult issues to light.

According to Berkun (2005: 12-14, 17), a successful project manager needs to have both management and leadership qualities, since managing well requires leadership skills, and leading well requires management skills. Anyone involved in project management will be responsible for some of both. Also, in project management, there is a risk of confusing process with goals. If a project manager is unsure what to do or afraid to do things, he or she can pay an unnecessary amount of attention to charts, tables, checklists, and reports. It is possible that at some point the project manager begins to believe that the data and the processes actually are the project itself. He or she focuses on the less important things that are easy to work with, meaning spreadsheets or reports, rather than the important things that are challenging to work with, meaning the work effort or the schedule. Good project managers avoid too strict a division between project management tasks and the project itself. Adherence to checklists implies that there is a definitive process that guarantees a particular outcome, which is never the case. In reality, there are always just three things: a goal, a pile of work, and a bunch of people. Instead of focusing on processes and methods, project managers should be focused on their teams. Every team and project is different, and there are often good reasons to question old judgments and processes.

Project managers are not hired to contribute a linear amount of work like a worker is expected to do, but they are hired to amplify the value of everyone around them. No project-tracking system completely replaces the need for people to talk to each other about what is going on because social networks are always stronger and sometimes faster than technological ones, and the project manager has a critical role in making the information flow active and healthy. The project manager spends more time with each person on the team than anyone else, and his or her mood or outlook will rub off on anyone he or she encounters every day. Whatever project managers bring to the project, good or bad, will be contagious for the rest of the team. They need to be genuinely interested in helping their teammates and be successful at it more often than not. (Berkun 2005: 14-17)

Project managers enforce schedules, and all schedules serve three primary purposes. The first is to make commitments about when things will be done, and the second is to encourage everyone contributing to a project to see his or her efforts as part of a whole, and invest in making his or her pieces work with the others. If used properly by a project manager, schedules force everyone whose work appears on them to carefully think through the work they need to do and how it fits into what others are doing. The third purpose of a schedule is to give a team a tool to track progress and to break work into manageable chunks, which will give a clearer understanding of what tasks will be done and when, and each team member has a greater opportunity to ask good questions and clarify assumptions. (Berkun 2005: 17-18)

One of Berkun's (2005: 170) main thoughts is that if communication does not work, the project cannot succeed. Nowadays the speed of communication does not present any problems, thanks to E-mail, phones, and instant messaging. Instead of speed, the primary problem in communication is quality and effectiveness of communication. Also, there need to be effective and healthy relationships between the people who are working together. Since many decisions are shared, and much work is done collaboratively, without good relationships no amount of extra communication matters. Project managers spend a lot of time communicating with individuals and groups, so they have more responsibility for effective communication than other members of the team. (171) Management by walking around, meaning spending time dropping by in teammates' offices and discussing things in person in a more of an informal way, is described as a central quality for successful managers. (173) As Berkun (172) says, "dialogs are better than monologs."

Both Berkun (2005) and Esselink (2000) agree that if communication does not work in the project, the project will fail. Whereas Esselink concentrates very heavily on checklists, tools, processes, and procedures, Berkun gives much emphasis on the interpersonal relationships and in a way, emotional intelligence, which is described by Goleman (1998) as "the capacity for recognizing our own feelings and those of others, for motivating ourselves, and for managing emotions well in ourselves and in our relationships" and which helps the project managers to be conscious on how to communicate and act in different situations. Relationships between the project manager and any team members or of course also between different team members can either improve or hinder the progress of the project, depending on the situation. The project manager needs to recognize this in order to make the best of any situation or phase of the project. In addition to knowing the technical and business side of the project, project managers therefore need to have interpersonal skills to gap both the person-level and business function-level culture differences inside the company in the best possible way and make everyone work towards the same goal. As Berkun says: "Project management is about using any means necessary to increase the probability and speed of positive outcomes." (17)

# 3. Working with localization agencies

There are many localization agencies in the market to choose from. Some smaller ones concentrate on one specific market, field of expertise and/or language, while the larger ones function as multi-language vendors capable of localizing one product into several languages either internally or, most often, by taking care of the tasks connected with handing out the actual translation and localization work to other offices or vendors in other countries. Often large multi-language vendors are keen to offer their services as hub offices for the customer. There are two reasons for this: to make the customer's work easier and, more importantly, to make more money for the hub office. The hub is a central contact point for the customer wishing to have the product (or just a set of files) localized into several languages at the same time. Acting as a hub office means that the localization company provides the customer with a single contact point through which all localization work can be guided, no matter how many languages or related processes are needed. The hub gets the material from the customer, possibly converts it or handles it in some other way, and then sends it out to several other offices in other countries to be localized at the same time. The hub will also work as a reporting and solving point of problems, either with or without the customer's help.

It is worth considering to use different localization companies or even different offices of the same localization company directly for different languages without using a hub at all. Real life experience has shown me that when the amount of localization is not too large (in general, a large project is considered to consist of more than 50,000 words to be localized) and the project is not too complex (consisting of many different tasks and components), using a hub office for several languages brings no extra benefits, but quite the opposite.

Using a central point of contact can be of great use for a customer who does not have much experience in localization and who has little time for scattered or even non-professional communication. Usually the customer also gets a small discount for the total cost of work for using a hub office, but unfortunately this is where the problems tend to arise. My own experience has shown that the different country offices of a large international localization company compete against each other and strive to meet the financial goals set by the central organization. The different country offices usually have a certain set of "inter-company prices", which are considerably lower than the prices they use when working with direct customers, to be used when working together with other offices of the same parent company, and since the hub invoices the customer according to the public prices, even discounted ones, it therefore gets most of the money. In the hub office scenario, the other country offices have to carry out the same work they usually do, but they have to do it for less money than they usually make.

An example of the difference between inter-company prices and the direct customer prices from 2001 is as follows for English to German translations (the French were using exactly the same inter-company prices):

| Type | Software per word | Documentation per word |
|---|---|---|
| Inter-company | 0.21 euros | 0.14 euros |
| Direct customer | 0.25 euros | 0.22 euros |

The Japanese localization company was using the inter-company prices with SSH Communications Security at the time, mostly due to me having done business with them in my previous job and maybe also them not realizing at first that I had changed companies. They sent me the inter-company price list (extract above) which the 13 country offices were using, and technically I was really not supposed to receive that. However, they held on to those prices most likely in order to be polite or because of my previous relationship with the company and persons in question. As Berkun (2005: 171) states, the effective and healthy relationships between people who work together do make a difference. The prices were discounted to 0.37 euros per word for software and 0.26 euros per word for documentation, and there was an another offer from a different company, which was using 0.60 euros per word for software and 0.50 euros per word for documentation localization.

This all means that all the offices want to function as the hub office whenever possible and none of them wants to work for the hub if they are given a choice. In the above pricing example the hub office gets 0.04 euros for every word of software and 0.08 euros for every word of documentation (minus the possible discounts) for the localization work they do not actually do themselves but just forward to other offices and take care for the communication and possibly also some technical tasks. This leads to the unfortunate fact that the inter-company work is usually given less priority than the direct work and also inferior or even insufficient resources, which of course can and often will result in inferior quality and delays in schedules. The hub office gets the blame for the delays or the quality issues, so working as a hub is by no means an easy task either.

Maintaining direct contact with the people who actually carry out the tasks or who hand out the files directly to the people carrying them out gives better control of the events taking place and brings possible problems out in the open with no extra delay or "censorship". It has often been the case that localization agencies try to cushion the situation when communicating with the customer and try to sort out the arising problems by themselves or hope that they get solved in the process instead of warning the customer about possible delays in schedule. This is because large international localization agencies compete against each other and tend to try and win customers from each other in every possible way. They do not want to give their customers any unnecessary doubt about whether the agency is totally competent in carrying out the task or tasks. For some parties, talking about problems seems to be a sign of weakness or incompetence, but as Berkun (2005: 206) argues, it is natural to encounter problems and responding to adversity might contribute to success more than the ability of avoiding problems. When working with several offices instead of just one, the speed and quality of communication increase and the possible inter-company problems do not exist. Since the software company handing out the localization work has ownership of the files and also has developed them and will process them further after the localization work is done, they are also fast and competent in solving any problems with the files or tasks or answering the questions regarding them.

When a hub office is not used, all the possible problem-solving needs to be carried out together by the contacts on the customer's side and on the localization agency's side instead of having the hub organization work out some of the problems themselves before contacting the customer. As always, there are two sides to this issue as well. The problems will not get "filtered" before reaching the customer, the message will be clear and complete and the interaction fast. However, there are cases where the translation agency's questions can be irrelevant for the task, about internal problems or questions that they should solve by themselves. Esselink (2000: 460) argues that the problems should be communicated as soon as possible, but due to the competitive nature of the market, this is not often the case in real life projects.

## 3.1 Advantages and disadvantages of outsourcing

There are many reasons why it would be useful and sensible to outsource the localization and translation tasks to a professional localization company or even to a freelance translator or a group of freelance translators. First of all, the company which needs the localized product often has no qualified people in-house for carrying out the necessary tasks. As Esselink (2000: 6) states, most software companies nowadays outsource the translation and localization activities, since the companies more often than not do not have a large enough base of employees who are qualified for those tasks. According to my experience, however, many people, especially in the IT field, seem to think that anyone with even moderate skills in a language can translate any text into that language or at least proofread any text that is written in a language one knows. This belief can result in very poor proofreading with no or even false corrections, which of course lead to poor quality. Unfortunately this happens all too often.

Furthermore, more often than not the in-house people are heavily involved in other tasks beside the localization – which is to say that they are carrying out their normal tasks – and because their normal tasks are not necessarily connected with localization, the in-house translators-to-be might lack both the skills and the time necessary to do a good or even a satisfactory job. They can not concentrate on the tasks and often have to work in a hurry. If they need to prioritize between their own tasks – for which their work performance is most likely evaluated – and the extra tasks brought to them by localization – which most likely nobody even remembers to give them credit for – they will naturally choose their own tasks over localization tasks. Since small companies often have no need for full-time translators or localizers, they either have to use in-house people already carrying out other tasks or outsource the work to someone or somewhere else. Therefore outsourcing has become reality for an increasing number of companies for translation and localization tasks as well.

Using an in-house translator is seemingly much cheaper than buying the translations or the work from an external company. Outsourcing localization work will result in an invoice that needs to be taken into account in a company or department budget, while using in-house people often shows only in the company or department salary costs. However, the time spent on translating is often more valuable than the amount that shows on the external invoice, especially if the in-house translators or localizers should have been doing other work instead of translating or proofing translations. When resources are pulled from other tasks, those tasks suffer instead, and the whole concept of localization can often be seen as a burden instead of a positive addition.

If a company has professional in-house translators, they can leverage the company's other resources and information sources much better than a localization company's translators. Since in-house translators are often near the development process either physically or as a process, they can easily access all the material and interview all the people they need in order to solve problems with both the material they are handling and the process itself. Since they usually work inside the company, in-house translators usually have a lot of background information about the company and its products, which adds to the quality of localization and the ease of finding the necessary information. When the localization work is outsourced, the information of the company and in the company is usually not as easily accessible.

All in all, the best choice would often be to use professional in-house translators hired for only translating and localizing instead of outsourcing the work. In real life few companies can afford having or even have the need for people who do only translation and localization tasks in the company. Large organizations, such as Microsoft, IBM and Novell, naturally have their own internal localization experts and departments, but they are constantly developing a wide range of products for many markets and need the constant localization workflow in order to keep their international offices, subsidiaries and customers satisfied – and of course, their revenues flowing. They have committed themselves to providing their customers updates and security patches, and they need to have them quickly for their localized versions as well. Having a department which is able to take care of the translation, localization and, when required, outsourcing on very short notice is essential and even mandatory for their businesses. For smaller companies needing localization only occasionally it is usually more convenient and cost-efficient to use as much outsourcing as possible for all localization tasks.

Testing is closely connected with software localization, since the software regardless of language needs to be tested before releasing, and localization is one potential stage to introduce bugs or other errors to the product. Outsourcing testing tasks has two sides to it. On the one hand, the company's internal testing processes are most likely set up according to the needs of the testing of the particular product developed by the company and then translated. The hardware conforms to the requirements of the software, but the localized environments, such as Japanese Windows and especially Japanese or other double-byte enabled servers, are not necessarily available, or there might be lack of expertise in other language versions' or environments' requirements or specific features. The testers also know the tasks beforehand, since they have carried them out with the English version, but they might not have the language skills necessary to carry out the tasks well enough to notice all the truncations and mistakes. On the other hand, there is a danger that the in-house testers who have already tested the English version might become “blind” to some mistakes in the localized version, since they are so used to seeing the English version and carrying out mostly very technical testing. Therefore, using a new environment and a totally new set of testers who also know the language the software is localized into, might yield results which can also be leveraged for the core product: new testers sometimes find and report problems which exist in all language versions while testing the localized version. Localization testers are also able to concentrate on the relevant issues, which is further illustrated in chapter 4.11, where I discuss Secure Shell testing in more detail.

## 3.2 The process as seen from the perspective of a software company

When a software company makes a decision to localize a product it has developed, the goal for localization usually is to try and expand the company's market to areas which use different languages from the language the product was originally written in. The company has to take into account not only the tangible costs of the actual localization work but also the in-house effort and possible issues with the release schedules, not to mention the packaging, marketing, and sales of the localized products.

Localization enables a company to enter new markets and to compete effectively in the market environments. In many countries, language barriers and nationalism preclude end users from utilizing English-language software. According to Layden (1997), in China, for example, most end users do not have good enough skills in English and require software which uses their native language. In Germany, however, most people can read English as well but prefer their software to be in German. Products localized into German will have a better chance of competing against native German products. Localization also enables the company to build credibility in the target markets. By creating a product in a country's native language, the company shows that it respects the target culture's history, customs, and language, in addition to which it also demonstrates that the country's business is very valuable to the company and that the company is dedicated to meeting the needs of all of their customers.

If localization of a company's product is discussed at all, it usually indicates the need to expand the business of the company or an opportunity for doing so. A competent and successful software company releases software products only when they have been planned, coded, and tested with care and attention. They usually know what is expected and required from a software product. The company should see the benefits of localization and act accordingly, setting up the processes and channels, enabling the localization from the beginning and supporting it in marketing and sales as well. If all the parts and functions of the company do not support the localization and the localized product, the end results will be less than optimal.

### 3.2.1 Why localize?

As mentioned earlier, in some countries the local laws state that at least the instructions for end user software should be available in the local language or languages. (See for example Esselink 2000: 5) Also, depending on the target audience, which might be for example the military, hospital staff or factory workers, the software might have to be in the local language or languages in order for the possibly non-linguistically oriented users to be able to fully understand and operate the software safely. Some countries, such as France and Japan, are more proud of their language heritage than others, so if the software user interface and instructions are not written in the local language, users might regard the whole application as a second-rate product even though the technology the software is based on might be of superior quality. In addition to this, some countries and languages use special character sets, which in most cases calls for internationalization in order to make the products work in the target market at all. Since internationalization can be costly, troublesome and time-consuming, localization is not a large cost on top of internationalization nor does it in this case cause much delay in release schedules. As a result, the decision to localize software might not meet much resistance for those markets.

As also discussed earlier, localization can increase the company's revenue and make the company seem more interested in and committed to the area and culture it is operating in. This improves how the company is seen by the members of that particular culture. Localization can thus add to the credibility of the company’s non-localized products as well, which in turns can result in increased revenue – the goal for most companies.

### 3.2.2 Why not to localize?

Many Finnish software companies, such as SSH and F-Secure, write their software, including the documentation, in English, since Finnish is a minor language in the world and understood only by a small number of potential users of the software product. Writing the software in English lets the companies have a much larger market than just the domestic one without actually localizing their software. In addition to this, most IT professionals, who are usually the target group for the Finnish software companies, have good knowledge of English, so they can and will use English software developed in Finland. If a software product is developed for a company to use instead of an end-user to use, the consumer protection laws do not deem localization mandatory. Also, if localization becomes necessary later on, English is the most common language to localize the software from, and also the most cost-efficient one.

English is the major language in the IT industry worldwide, and using English software makes large companies’ inter-country offices’ communication easier than using multilingual software versions. This means that people using the same English software anywhere in the world can see the same user interface strings and error messages on their screens during a phone conversation, for example. The job of the IT departments becomes easier as well, since they can have one set of instructions and tell the end user which button to click, since the text on the button will be the same everywhere (unless the product is developed to use the buttons and some other strings directly from the operating system, which can be either localized or non-localized). When there are different localized versions of a product there is usually a need to have the support in understand or use the languages the product is localized into.

Sometimes it is no use for a software company to localize its products. It may not have resources for making the product available outside a certain market, which by itself makes it clear that there is simply no need for localization. The product might also be simply too market-specific to be localized, such as a program to help out to fill a tax report or to study for a drivers' exam, which vary from country to country.

### 3.2.3 The effect of localization on a company’s processes and finances

In most cases, localizing a software product consumes some of the software company’s resources even if many localization and translation tasks could be outsourced. Someone usually has to work as a middleman between the internal development team and the company or people carrying out the localization tasks. The files need to be prepared, proofreading and the final sanity check arranged and the testing coordinated. Of course all tasks can be handed out to an outside party to carry out, but generally software companies do not want to send out all the source code to anyone – and in some cases business deals and agreements do not even allow sending it out – nor do they want to lose all control of the process of preparing a product which carries their name and thus works as a showcase of the company. Bad localization brings bad publicity, which in turn works against the company, bringing down the positive brand recognition. Ultimately, this might also have a negative effect on the stock value, even though just bad localization is seldom seen as bad business by itself in a wider scale. Of course there are cases in which outsourcing all localization and tasks connected with it has worked perfectly, but it requires selecting an outside party totally capable of taking care of the localization tasks and understanding what localization is all about.

An important thing to keep in mind in localization and in all kinds of translation is to take into account the cultural differences and the fact that language does evolve over time because of social and cultural transformation and technology of mass communication. (McCrum, MacNeil and Cran 1992: 375) This goes hand-in-hand with the aspect of respecting the language and the experts of the language. As I mentioned earlier, I have experienced that software engineers and financial people often seem to think that anyone who speaks a language as his or her native language or even has some knowledge of a language through low-level studies or having been a resident of some country can translate or proofread all kinds of texts with ease. This is a good example on the cultural differences inside companies that Berkun (see 2005: 8) mentions. While the kind of a person just described carrying out the tasks might indeed have adequate skills for translating or proofreading a text, more often than not this is not the case. Proofreading is all too often assigned to people who have minimum language skills and often also minimum work ethics or interest in the task. Unfortunately small companies with not enough understanding of the linguistic issues also keep on using inadequate in-house personnel to translate texts into foreign languages, partly because it is much easier and cheaper, partly because the non-professional people seem to think they are good enough and might be offended if proved wrong. The company needs to set up a process with professional people in it for this task as well. I have found out that it is often useful to point out to the willing Finnish listeners that not all Finns can proofread Finnish nor write grammatically or even typographically correct Finnish, and that the same might be true for other languages as well. Pointing this out has helped to "teach" some of the software developers and other non-linguistically oriented people to agree that it might actually be a good idea to use professionals for this task as well, since in order to produce material which is regarded as being of high quality in the market an organization needs to be professional in all the levels visible to outside parties.

### 3.2.4 Deciding whether to localize or not

There is always a definite goal for software localization. In some cases, localization is mandatory in order to enter a specific market or to stay there, and in other cases localization is just a means to build up the company profile. However, the most common goal for localization is to make money. Market research could be a helpful tool when deciding the potential of a certain market with regards to releasing a localized version of a product. The research could be based on direct questionnaires or surveys, sales experience or on how other products in the same field and market area have been received and the reasons for that. For example LISA has carried out such generic research on the Chinese market (LISA 2003).

Experience has shown me that some companies that have multiple language versions of their products, such as Microsoft, have specific internal localization tiers for different markets. This means that there is a comprehensive release map where a certain set of languages is bundled together to form a tier, which consists of different language versions to be released at the same time. One example of a normal bundle or a tier is FIGS (French, Italian, German and Spanish) which all are major languages in the business world and thus very important and usually also highly profitable, but these tiers will vary depending on the company and product, and for example Adobe PDF Ifilter's tier 1 languages are English, French, German, and Japanese (Adobe 2006). Of course, the income from a localized language version should exceed the costs of localization and the related items, such as the internal personnel costs, project management, testing, advertising, packaging and transportation. Costs are, however, not the only factor to take into account when deciding whether localizing a specific product into a specific language or a set of languages is of any use and profit.

Company-specific factors may also affect the decision process. Many companies have some sort of processes and action plans according to which the companies decide if a product is to be localized, including when and how the localization decision is made. The product cycle, which means among other things how often a new version of a product is released, can be a deciding factor in determining whether all the versions including minor product versions are to be localized or if the localization should concentrate on the major product versions only. Major versions can be named for example version 3 and 4, and minor versions are usually indicated with a second number, such as 3.1. Some bug fix versions can also have a third number, such as 3.1.1. This varies from company to company. (See an article on version numbers on Wikipedia at http://en.wikipedia.org/wiki/Version_numbering.) Major versions usually contain new features and major bug fixes, whereas minor versions are usually released to provide the customers with a quick fix for a bug in the software code which can be considered serious or even critical. It is a common course of action that in case of critical bugs, a localized minor version of the product is usually released as quickly as possible. Usually a bug fix does not call for much translation, since the fixes usually consist of core code fixes only plus maybe a few lines of release notes, but no separate new code or new features as such and the user interface is seldom changed because of a bug fix.

Sometimes localizing a product just does not fit in the company's plans or the way it sees the market. The reasons for this are naturally company-specific. Localization might be financially justified in terms of sales figures, but scheduling or support issues might be problematic. The release dates should either follow the core version release dates very closely, or a separate release schedule needs to be implemented for the localized version or versions. The first case means that localization should be planned, scheduled, and started well in advance before the core version is released, depending on the localization effort. The problem for this method is that when the developers change the code and user interface or the technical writers change the documentation, which is normal and very common in the later stages of the software development process, extra effort will be needed in localization as well. This means that if localization starts for example a month prior to the release date of the product, there will be changes in the material which has already been localized, and thus some effort is wasted and new effort will be needed periodically or at the point the code and the documentation is frozen i.e. final. Using this method will result in extra costs, but the product can be released very close to the release date of the core product. (See also Esselink (2000: 22-23) for his view on processing updates.)

However, if the localization process can only be started after the core version has been released – or more preferably when the code and the documentation has been frozen – the localized version might only be ready and released relatively close to the release date of the next core version. This will cause problems for sales, since if a potential customer has a choice between purchasing, for example, a localized version 3.1 of a software product only about a week or a month before an English version 4.0 of the same product is released, the customer will of course weigh the options of having a version which uses a familiar language or a version which might have been very much technically improved, the localized version might suddenly seem to be less attractive an option unless the purchase agreement covers some localized releases in future or even the non-localized core version.

Another important issue to take into account is the product's technical support. If the core version of the product is, for example, in English, the company's support department also works in English and support requests are most likely written in English. However, if the same product is localized into German, for example, the user interface and the documentation will also be in German. This means that the support requests and contacts will in some cases also be written in German and they will most likely contain German error messages or references to the German user interface. Because of this, the support personnel need to be able to read and write German, which in most cases means that the company needs to invest into new support engineers familiar with the language into which the product is localized or that some of the support issues need to be outsourced. Using error codes with the error text will make the errors more universal to understand but it does not solve the whole problem. Of course it can be agreed that the support will be for example in English only, but if the customers are provided with proper support according to their language version, this will be an extra cost which is not easily visible in the localization costs.

Software products usually continue to be updated now and then until development is discontinued for one reason or another. Among those reasons are security updates, which means fixing possible security risks in the products, fixing errors in the software code, and adding or improving the application's features. It is common that when a customer purchases a software product, he or she is entitled to at least some software updates. For example purchasing version 3.0 of a software product might entitle the customer to receive free software updates until the next major version, which in this example would be version 4.0, is released. This means that when a version of a software product is localized, there either needs to be a new localized version in the future or the License Agreement or the Sales Agreement of the product needs to state clearly that there will not be any new localized versions available in the future without a separate agreement or payment, or that the customer has purchased rights only to the specific version that was available at the time of purchase, and that no localized software updates will be automatically provided to the customer.

## 3.3 The process as seen from the perspective of a localization company

During the five years I spent working in a localization company, there was a big change in the way the business was conducted. First the company was considered a small "family owned" company, and it employed only about 40 people. The company had two separate checking or proofreading stages, both technical and linguistic, which was to result in very high quality and high levels of customer satisfaction. In time, the company grew and was eventually bought by a large international localization company which sought to expand their offices and functions into the Finnish market in order to serve the large international software companies, which needed to have a full set of languages localized using a single contact point. With this change, the company became more and more profit-oriented, and the “unnecessary” costs were cut. An example of this was that the two-round-proofing was no more an automatic way of conducting business but rather a separate function for which the customers had to pay extra, which meant that only one round of proofing was offered to the customer for the price they had earlier paid for the two rounds. If the customer’s needs indicated that higher quality was required, the price went up and the second round of proofing was added. This reduced workload and made it possible to make better profits. The new, more profit-oriented approach of many large localization companies versus the more personal approach of some smaller companies is good to keep in mind when commissioning localization work and considering approaches to various localization tasks. Large companies have their benefits, of course, but they also have more financial pressure from the organization.

### 3.3.1 Quality

The linguistic quality of the translations is generally good when using a professional localization company, since the majority of localization companies have linguistic experts who proofread the translations. However, what might cause problems is the technical quality of the translations, since there is no way a translator could be a professional in every field, and localization companies often do not go to the trouble of finding real technical experts for every field or product.

Translators also learn by doing, which means that it is very useful to try and keep the same people translating the same type of material, i.e. it might be possible to have a team of translators who are specialized in translating telecommunications material and another who concentrate on, for example, Microsoft software and documentation. This would ensure that the work can be started very quickly without having to train the translators for the task in question. The quality will constantly improve and the turnover speed increase as the translators learn more and more about the material by working on it. As Esselink (2000: 449) says, it is a good idea to limit the number of translators working on a project in order to maintain consistency. Since a localization company usually has quite a few projects going on at the same time, the translators will move from one project to another all the time. This is very common in the field and it means that the same people who were available to translate for example Product 1.1 might be working on some other project and thus not available for the Product 1.2 or Product 1.1.1 update when it happens to take place.

Large customers, such as Microsoft or IBM, often have specialized teams inside the localization company, and the team members can be kept busy with some Microsoft localization tasks all the time, but they do not necessarily remain working on the same application or product. Usually it is possible to have the software translators stay in translating software and the documentation translators translating documentation. This results in quicker completion of tasks, since translation is often carried out by using separate tools for software and documentation and mastering those tools takes time. Experience has shown that a powerful tool in the hands of an inexperienced but daring user can and will do great and sometimes even irreversible damage. For example, it is possible to delete or mess up all translations in some auto-saving translation tools with a simple copy-paste operating gone wrong, resulting possibly in the loss of tens of thousands of translated words. Frequent backups and Translation Memories help to reduce the risk, of course.

Because the localization company needs to change the team doing the translation work more often than not, this often results in increased training or learning efforts. An example of a real life mistake which lead to a great deal of extra work is that for a certain Microsoft's localization project it was specified that the translators would have to use English version of Microsoft Word 2000. However, some of the project's translators were not properly instructed, and they used the localized Finnish version of the same application. This could have been fine, but in this particular case the document adopted localized style names from Microsoft Word, resulting in the separators in the style names being commas (,) instead of full stops (.). This in turn made compiling the Help system from the localized Word files cause hundreds of errors and eventually fail, since the style names were hard-coded in the compilation code. The result was that all text (without the paragraph marks which were the style placeholders) needed to be copied from the localized files to the English files, replacing the English text. A simple find and replace operation of styles would not fix the problem and the styles were embedded in the Translation Memory, so this safe and most reliable way to fix the issue amounted to many hours of extra work.

### 3.3.2 Scheduling

The more in-house people or freelance translators a localization company has available, the more flexible the company can be in terms of schedule. In general, it would be optimal to have only one translator for a single project in order to achieve maximum consistency (see Esselink 2000: 449), but usually the schedules do not allow this.

Real life experience has also proven that localization schedules set by the company handing out the localization work change quite often, for example due to an internal development task taking place before the localization needing more time than originally estimated. All too often the postponing of the localization starting date does not affect the deadline for localization or affects it too little. This is often because companies find it more acceptable and easy to set stricter schedules for outsourced tasks than internal tasks, not realizing that localization quality is often worse than when localization tasks are given enough time.

The calculations which were used for measuring the speed of tasks connected with translation in order to evaluate the need of resources and time by the localization company I worked for were as follows (they vary a little from company to company and in reality are somewhat higher these days):

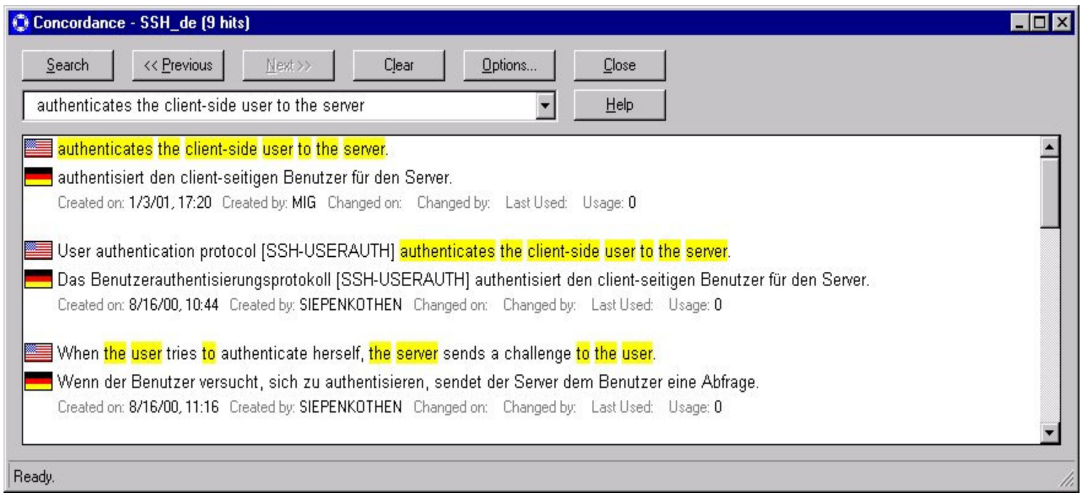
Manual and Help translation: 250 words per hour
Software translation: 150 words per hour
Technical validation: 1,000 - 1,500 words per hour (varies with the material type)
Linguistic validation: 1,000 - 1,500 words per hour (varies with the material type)
Desktop Publishing (DTP): 6 pages per hour
Screen shots: 2-3 complex screens and <10 normal screens per hour
Words per a full-text page: 250

This means that if a manual has, for example, 50,000 words, it should technically take 50,000/250 = 200 hours to translate. A manual of 50,000 words usually consists of about 200-

300 pages. Similarly, a software of 15,000 words takes 100 hours to translate. On top of that, there will be at least one proofing round and a DTP (Desktop Publishing) stage at the very end of the process. The DTP stage is where the layout of the document pages is finalized, pictures and callout boxes re-sized so that the manual looks similar to the original, non-localized manual. Sometimes the original manuals are somewhat flawed, so the localized manual can even be technically of even better quality than the original one. This work has to be done after all translation and proofing tasks are finished and screen shots (sometimes called screen captures as well) and other pictures inserted in the document, since any changes to the text or graphics will change the layout as well. Sometimes the graphics might need to be translated using an image editing application or some screens might need to be captured from the up-and-running software.

### 3.3.3 Translation Memories

Using a good Translation Memory will make the actual translation time shorter especially when the material to be translated has repetitive sections or sentences. (See also Esselink's take on different Translation Memory tools and their functions (2000: 362-378).) A Translation Memory enables the translator to easily re-use previously translated material to translate new material with similar sentences or segments and also to do concordance searches and find out how a specific term or string has previously been translated. When a source text sentence or a group of words is selected and a concordance search is carried out in a Translation Memory tool, the tool offers the user sentences or groups of words which are similar or somewhat similar to the original ones. When a new sentence or a segment is translated, it is added into the Translation Memory used while translating, after which it will be usable in future translation.



authenticates the client-side user to the server

authenticates the client-side user to the server.
authentisiert den client-seitigen Benutzer für den Server.
Created on: 1/3/01, 17:20 Created by: MIG Changed on: Changed by: Last Used: Usage: 0

User authentication protocol [SSH-USERAUTH] authenticates the client-side user to the server.
Das Benutzerauthentisierungsprotokoll [SSH-USERAUTH] authentisiert den client-seitigen Benutzer für den Server.
Created on: 8/16/00, 10:44 Created by: SIEPENKOTHEN Changed on: Changed by: Last Used: Usage: 0

When the user tries to authenticate herself, the server sends a challenge to the user.
Wenn der Benutzer versucht, sich zu authentisieren, sendet der Server dem Benutzer eine Abfrage.
Created on: 8/16/00, 11:16 Created by: SIEPENKOTHEN Changed on: Changed by: Last Used: Usage: 0

Image: An example of German Secure Shell's Trados Translation Memory's concordance search.

If a file has been translated before without using a Translation Memory and if both the source and the target files are available, they can be used to create a Translation Memory with Trados' WinAlign tool. This will save money when implementing Translation Memory usage in a company when translating material, since almost all old previously translated material can be re-used in translating and searching for terms.

Trados, a leading Translation Memory application provider, described the concept of Translation Memory as follows:

> Translation Memory is a type of software that has the ability to store and reuse translations produced by human linguists and translators. Human translators always produce the translations. But the software remembers and empowers organizations to reuse those translations as content goes through revisions and updates. Translation Memory should not be confused with machine translation, a type of software that produces the translation itself and can be used to produce draft, but not high-quality, translations. By contrast, translation memory stores and reuses already created human translations. (Trados 2004)

### 3.3.4 Processes

Most localization companies have a very specialized structure of employees. Some of the employees are managers or people taking care of the "normal" business functions of any company, others project managers, translators, engineers, DTP specialists, proofreaders and so on. (See also Esselink 2000: 13-14.) This usually means that there is also a structured workflow in place, with specialists at each stage carrying out the tasks suitable for and assigned to them. The management and sales people will establish contacts with customers who hand out the work to the assigned project managers. The project managers coordinate the work by either preparing the files themselves for the translators or assigning an engineer or a translator for the task. Preparation of files can consist of, for example, converting the files to a format which supports using a Translation Memory tool or some other tool necessary for translating the files or pre-processing the files so that only the relevant sections of the textual content can be translated or touched at all. The translators translate the files and hand them back to the project manager, who first makes that sure all the text to be translated in all the files has been translated and checks that no files or tags have been broken in the process, and then forwards the files to the technical and linguistical proofreaders either as they are or after preparing them, depending on the tools used in translating the files. When the files are proofread, the project manager hands back the feedback and corrections in either paper or electronic format to the translators, who make the necessary changes and corrections and hand the files back to the project manager. Then the project manager makes sure that the corrections have been implemented and that the files are not damaged in any way, after which he or she converts the files back to the original format as required by the customer.

If the files need layout formatting, the project manager assigns a person to take care of the DTP tasks before the final handback to the customer. Building the software and compiling the Help system can also be a part of the localization company's workflow. For example Microsoft assigns building tasks to the localization companies by supplying them with the technical and process requirements, build environment files, tools and necessary settings to be used in the process. The localization company sets up the build environment for as many languages as needed, builds or compiles the files and usually also carries out both functional and visual testing of the built software. The software building workflow varies from project to project and is not required by many customers, but the Help compiling is quite a common task and it is carried out by using a Help compiler. The compiler creates a single standalone Help file from all defined source files and other components of the Help project. It determines what the Help system contains and how it looks and functions. (Help Basics: 9)

This, of course, requires a team of testers and engineers, whose tasks include building, functioning as a test lead, testing, fixing the bugs and carrying out regression testing after the bugs have been fixed. Webopedia (2004) defines regression testing as selective retesting of a modified software system to ensure that any bugs have been fixed and that no previously working functions have been broken as a result of the repair work.

Sometimes it is also necessary to grab some screen shots from the built software or, which is just as common, “fake” the screen shots, if a built software version cannot be used. Faking the screen shots means using an image editing application and replacing the original text strings with the localized counterparts. This requires using a Translation Memory or a list of localized terms in order to get adequate results, since the screen shots must be using the same strings the localized software is using. Unfortunately, it is often the case that the localized and built software is not yet ready to be used when the screen shots need to be inserted in the material. A good example of this is a press release or an advertisement of the up-and-coming localized software product. Very often a press release or an advertisement is the first thing to localize in a project, since it is usually published before the actual software has even been handed out to be translated or even released in English. Therefore translating such material often includes much guesswork, there being next to no context available for the translators or proofreaders, and the results can be misleading.

## 3.4 Freelance translators vs. in-house translators

As mentioned earlier, there are basically two types of translators; freelance translators and in-house translators. Both of them have distinct advantages. I will give a short description on the differences between these two types of translators, how they are seen by localization companies and also what should be known about them in order to understand what might be going on in the localization process. Usually the problems with translators will and have to be taken care of by the localization company – the representatives of which more often than not do not tell the customer the whole truth about any problems in their internal workflows – but it is good for the customer to know what can be expected and what can be going on behind the scenes.

In-house translators usually have fixed or semi-fixed working hours, and their work schedules are generally known and set by the localization company which employs them. They usually specialize in a certain area of work or work in the projects of a certain customer or a group of customers. However, when a new project or a new customer comes about, they often have to function just as temporary help in such projects, since their main area of work might require their work efforts in much more constant way in future. Usually the in-house translators are not as flexible as freelance translators in terms of schedule requirements, since they more often than not are working for a set amount of working hours per month. The in-house translators will most likely not have much problems in communication and file transfer.

Freelance translators are “free”. Emphasis on that word. While this is not to say or mean that they do not cost anything, they are quite inexpensive and usually do not have any set schedules or even a clear-cut set of customers to work for. The only cost to the localization company when using a freelance translator is the price per word or per task that the freelance translator charges the company. There are no hidden costs, such as equipment or workspace costs, very little management costs and no healthcare, insurance or related costs. Most localization companies expect the freelance translators to sign a contract with them indicating that they are not to try and get any job assignments directly from the customers the localization companies have them working for. This means that if a localization company gives a freelance translator, for example, some work which originates from Microsoft, the freelance translator is not to do any translation work directly for Microsoft for a set amount of time after he or she translates the last piece of Microsoft work for the localization company. This is very sensible for the localization company, since it might easily lose work if they get pushed to the side and the translator or translators will be directly employed by the customer which earlier commissioned the work to the localization company, because it could be more cost-efficient for the customer to have a direct contact with the translators in most cases. In large projects and for demanding customers, this is less likely to happen anyhow, since the large projects are usually given to a central entity, a localization company, which takes care of many tasks, including training, Translation Memory work, proofing, testing, DTP, assembling a team of translators and so on, and working with a number of individual translators would only increase the time and effort necessary for coordinating the tasks. Using a single translator is generally of use for a customer who needs simple translation or proofing tasks carried out by outside parties.

In terms of schedule and workload, experience has shown that using freelance translators can be much more productive in terms of translated wordcount than using in-house translators. There are usually no extra tasks, such as everyday communication inside a company, meetings and other company-related interruptions of work, and the working hours are usually very flexible, since the work is often carried out at home or any location using a mobile computer. Since the translator is usually paid per word instead of by hour, the pressure to maintain the level of efficiency is harder than that of an in-house translator. Freelance translators sometimes translate workloads which are many times the calculated industry amount per day. This means that their working hours can be and often are long and irregular. Freelance translators have no paid vacation time, so a little extra effort is required to make the ends meet. The quality of the translations is most likely not as good as it would be during a normal "from nine to five" day, but the customers seem to indicate a very fast turnaround time is desired and even required. Of course the localization companies do have internal proofing, so it should not be a problem as such, but there is some extra cost for more effort in proofreading.

Communication and file transfer, however, might present problems for the freelance translators. The project manager usually communicates with a freelance translator via telephone and E-mail only, and there is little other communication with the project team. Since the translators might be doing the translations after spending a normal working day with some other company or studying and only open their E-mail or file transfer connections late in the day, the questions and problems might not arise during the working hours of the people at localization companies. In worst cases, this might mean that a day's amount of work is lost for both parties, including a day's pay for the freelancer, since he or she is to be paid per translated word, and no translations would be produced for the company during the hours the translator intended to work. For example, if a freelance translator cannot open a set of files for one reason or another or does not know how to use the tool or even have the tool required by the customer or the translation company, he or she might not get help or a new, uncorrupted set of files before the next day begins for the office of the translation company. The same is true for the deliveries. If, for any reason, an E-mail is not getting through or a transferred file set gets corrupted or if even a wrong set of files is returned, the localization company might not get the translations when needed. Transferring large files might present a problem as well, especially since some freelancers, who might be non-technically oriented or living in remote locations, might have slow connections, and file transfer might literally take hours. With cable modems or other broadband connections transferring large files will not be a problem. Of course, the localization company can send the files to the freelance translator after burning the files to a CD or a DVD via regular mail, but two-directional file transfer via regular mail means that there will be at least a couple of days lost in the process, and in a normal localization process and project there are no days to spare. A courier company could of course be used as well, but it can be both expensive and somewhat difficult. There is definite advantage for using freelance translators from different time zones. If a translator is on a time zone which is, for example, six hours behind the localization company, the translator can be given work when the working day is about to finish for the localization company and the work can be handed back by the beginning of the next working day.

Most localization companies have set buffers of time in the translation process in case problems arise. This means that if the localization company has, for example, 10,000 words to translate and the customer has given them five days to translate it in, the company might give the freelancer only three days, since they need to reserve about one day for proofreading and also like to have one extra day for buffer just in case something goes wrong. While this makes total sense for the company, it puts more pressure on the translator schedule-wise. Delivering the material in batches, i.e. staggered deliveries, would help but they are not always possible to arrange.

As a summary, both types of translators have their advantages and disadvantages. All major localization companies are leaning more and more towards using freelance translators, since they are not a potential cost to the localization company when the company has no work for them to do. When a localization company is using in-house translators, they are usually working for a fixed monthly salary or guaranteed a certain amount of work hours or words to translate per month, and the localization company has to pay them a fixed minimum even if the translators have no work to do. Freelance translators, however, get usually paid for exactly the amount of work they do, which makes them very useful employees, since they only generate profit for the company they are working for. In addition to this, freelance translators do not need to have computers or working rooms set up by the localization company, and it is fairly easy to assemble an army of translators for a project of any type and size via telephone and internet communication and very fast file transfer. The quality might suffer somewhat in favor of speed, but the tendency nowadays seems to be to focus more on quantity instead of quality.

# 4. Localizing the SSH Secure Shell client

The first decision to localize the SSH Secure Shell client was made in March-April 2000 by the Project Manager (who also acted as the Product Manager) of the product. I joined the company in May 2000 in order to set up the localization processes for the product and manage the localization, and there were no other assigned internal staff for the localization work nor did SSH have any other contacts in the localization field. The SSH Secure Shell product consisted of both client and server, but only the client was to be localized, at least to start things off. The version of the client was at the time 2.3, but since version 2.4 was to be released before the very first localization round would be over, the first released localized version was 2.4 for both scheduling reasons and common sense. The first localized language versions were French, German and Japanese, since the product management saw those languages as the most useful and profitable languages at the time. They also considered Spanish in the very beginning, but decided to leave it out at least from the first batch of languages to be localized into. Localizing the SSH Secure Shell client into Spanish was mentioned a few times over the years that followed, but the localization never progressed further than having the quotations for prices from a Spanish localization company and having them sign the Non-Disclosure Agreement (NDA). The following is a description of the process of decision-making for localizing SSH Secure Shell client version 2.3 and a description of the actual localization process of Secure Shell client version 2.4.

## 4.1 SSH Communications Security

SSH Communications Security's Web site stated the following in early 2003, which is probably the best way to describe the company and its background and goals at the time:

> Founded in 1995, SSH Communications Security is the world-leading supplier of Internet security software for demanding network security solutions.
>
> SSH provides best-of-breed cryptography and authentication technologies and products for secure Internet communications.
>
> The company's SSH® Secure ShellTM application has become the de-facto standard for secure logins, and now has millions of users in over 80 countries.
>
> SSH® IPSEC ExpressTM is the world's leading toolkit for IPSec (Internet Protocol Security), IKE (Internet Key Exchange) and X.509 certificate management system solutions.
>
> Numerous Internet infrastructure providers, OEMs, and system integrators have licensed SSH® IPSEC ExpressTM to incorporate high security into their products. These customers include many IT market leaders such as Compaq, CoSine, Ericsson, Lucent Technologies, Nokia, and Sun Microsystems, among others.
> (SSH Communications Security Web site 3/2004)

The company had offices in Helsinki, Kuopio, and Japan (Tokyo). The Japan office concentrated only on sales and marketing, while the Kuopio office's main function was development of the SSH Sentinel product. The Helsinki office had sales, marketing, main organization, and also development of several products, one of which was SSH Secure Shell. Just like all other IT companies, SSH Communications Security also suffered from the weakened market situation. The revenue and profit decreased and were at the time of the writing of the SSH section of this thesis (early 2004) in a small upswing. Growth was very slow, and the company decided to focus on a single product family, the core competence and the core products of the company. In November 2003, SSH Secure Shell was renamed and re-packaged into SSH Tectia, and several other products that were localized or about-to-be-localized, had either been sold to other companies or discontinued altogether. This means that SSH Secure Shell as such is no longer developed, but this thesis is based on and mostly written during the period of time when it still existed. (See a good technical description of the Secure Shell protocol on Wikipedia at http://en.wikipedia.org/wiki/Secure_Shell.)

## 4.2 What is SSH Secure Shell?

The following is taken from the documentation of SSH Secure Shell client 2.4, and it describes the product quite well:

> The SSH Secure Shell for Workstations Windows client (SSH2 client) is a program that allows secure network services over an insecure network.
>
> The SSH Secure Shell for Workstations Windows client replaces other, insecure terminal applications, such as Telnet and FTP. It allows you to securely remote login to remote host computers, to execute commands safely in a remote computer, and to provide secure encrypted and authenticated communications between two untrusted hosts. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel, expanding Secure Shell 2's usability even further.

In short, SSH Secure Shell client, which uses the SSH2 protocol, replaces Windows' default insecure tools such as Telnet (see description on Wikipedia at http://en.wikipedia.org/wiki/Telnet) and File Transfer Protocol (FTP) (see description on Wikipedia at http://en.wikipedia.org/wiki/FTP) so, as the need for secure communication was and is increasing, there was definite potential in the product to enter a wide market. The product had a lot of potential to bring in some profits, so localizing it for use in non-English markets did sound sensible.

## 4.3 Preliminary work, inspecting the material

After the decision to localize SSH Secure Shell client version 2.3 was made and I was hired by the company to work as the Localization Manager and manage the process as the project manager for localization, the first thing I needed to do was to prepare a budget estimate for the project. The project budget would consist of only outsourced tasks; no tasks that were carried out in-house were to be included in it. Esselink (2000) concentrates more on localization project manager's tasks in a localization company, but my work as the Localization Manager and the project manager of localization did include some of those tasks as well.

The Project Manager of Secure Shell presented me a comprehensive demo of the product, consisting of how the product works and some of the basic concepts and ideas behind the product and its technology. Much of this information was available from SSH

Communications Security's Web site at www.ssh.com. Only the Windows version of SSH Secure Shell was to be localized; there were several other supported platforms as well, but the end users needing the localized version the most were thought to be Windows users. The other platforms the product ran on were generally non-localized and in English, so the users of those were thought to do well enough with the English version. This made the task somewhat easier than first expected, since it is a great deal less demanding to concentrate on only a single platform. Now, when the nature of the product and the required tasks seemed to be clear, the next step was to find out which components and files actually made up the product in question.

Naturally there was the software component itself, with all the User Interface strings and error messages the user was to see on-screen at one point or another. Then there was a standalone version of the Online Help in single HTML (HyperText Markup Language) format files in a separate file and folder structure. The software had also a context-sensitive Help, which was the standalone Online Help system converted to HTML files which had more simplistic layout and structure, and which were then compiled into CHM (Compiled HTML) format to form a single file usable in Windows. There were also some screen shots which had to be either grabbed or faked.

After finding out what the components which made up the complete project were, the translatable source files had to be delivered for localization by the Secure Shell development team. I instructed the developers through the Project Manager of the product to make sure all the localizable strings – meaning all the text the user will or might see at some point when using the software – would be in those .rc files, since all of the source code would and could not be sent to the localizers for various security reasons. Due to the localization workflow being a new item in the developer team's tasks, I needed to get everything done through the Project Manager because they at first saw localization just as an unnecessary extra chore. As Berkun (2005: 8) states, there might be friction between the different factions of the company. By going to the developers' offices and talking with them a lot and letting them have their opinions and expertise heard instead of just sending out instructing E-mails I managed to make improve our co-operation and smoothen the processes. (See also Berkun

2005: 172-173) on how relationships enhance communication.) The developers handed out a comprehensive bundle of software files and the technical writer of the product handed out the documentation files. (See also chapter 4.5. of this thesis.) The software was created using Microsoft Visual C++ for Windows. In practice, this usually means that all the strings to be localized should be included in .rc format resource files and no other files needed to be touched. There were other files in the directory structure which were used by Microsoft Visual C++ for Windows to open the files correctly so that the layout would be shown visually by the tool and of course to build the software. The localizers would receive only the .rc files and the set of corresponding support files. There was no easy way to make sure all the strings were indeed included, but it was to be found out later in the process; in the testing stage at the very latest.

Then there was the Help system. The Help actually consisted of a bundle of SGML files (18 of them) and some common support files, which were used in the internal file conversion process. This process – which utilized a set of internal UNIX *sshdoc* tools – converted the 18 SGML files into 231 HTML files using a set of common support files to generate, for example, the headers and footers on each page. In addition to those files, compiling the CHM Help system required support files for Index and Table of Contents generation and information used when mapping the software's internal context-sensitive Help system to the separate HTML files. I will discuss this process with its requirements and problems in chapter 4.9. All in all, the SGML files needed to be converted twice using two sets of settings and *sshdoc* support files, since the layout of the context-sensitive HTML Help system was different from the standalone HTML Help system. The good thing was that even though there were two separate Help systems for the user, both systems were using the same set of SGML source files, which therefore needed to be translated only once. In addition to this, the PDF format manual would have had to be created somehow, and since the PDF file itself would be very hard to translate in that format and the original way to create the file would be to convert it from the SGML files, there would have been a need to create a workaround.

The conversion was tested with the English files, and the whole conversion process was very smooth and error free because the files had already been prepared and tested in the English product. The screen shots of the Help systems seemed to be quite easy to grab in-house, which meant that it was decided that to save time and money the grabbing was not to be outsourced – not only did the company have the right equipment and platforms but also had the knowhow on where and how the screens were originally taken.

After this I analyzed the files for word counts and prepared a small table of the workload and tasks in question. The work consisted of 4,837 words of software files originally in .rc format and later to be converted into RTF format for easy and relatively error-free translation, 24,143 words of Help system in SGML format and a couple of thousand words of documentation in small miscellaneous files in both HTML and RTF format. Of the total content of SGML help, 3,015 words were repetitions, meaning that from the 24,143 words 3,015 words were repeated as similar sentences or word groups, and even when using even an empty Translation Memory, that would result in a small discount for the commissioning party, i.e. us at SSH Communications Security. There would have been no discount for the repetitions if I had not brought it up, since the localization companies usually do not mention these types of discounts in the beginning of the process unless the customer knows about the possibility for them beforehand.

In comparison, it would have been possible to translate the HTML files as well, which would have eliminated the need for conversions and some of the hassle connected with it, but since a lot of words get added to each HTML file from the small support files, the cost of localization would have been over two times as much as by translating the SGML files and doing the conversions, and on top of that, there would have literally hundreds of more files to translate. The re-usability of the translation would also have been considerably lower.

The above evaluation phase, the material inspection part of it at least, fits right in Esselink's (2000: 430-432) description on the analysis of the material and the finding out of the tasks needed in the project workflow. The difference was that I needed to decide myself which tasks we, mostly I, could do in-house, while the localization company's project manager usually has a clear description of the tasks which the company was meant to carry out.

## 4.4 Selecting the processes, tools and workflows

After it was quite clear which material made up the localization project, the next step for me was to decide which process and tools to use for the project. Of course, in most cases the goal is to have things done as easily, efficiently and cost-efficiently as possible. The translation task can basically be carried out in two ways; using a translation memory or not using one at all. If the source file formats do not support using a translation memory, for example, due to using a non-standard set of tools in the development stages, then it usually is much easier and possibly also cheaper to just translate the material using the native development and writing tools instead of specialized translation tools, if the native tools are made available for the translators. The same is also true when the word count of the material to be translated is relatively small. The Translation Memory tools are constantly developed and updated to add support of an increasing number of development tools and file formats, so it will be possible to re-use the existing translations and memories even more widely in the future projects, no matter what format of files were originally translated or on which platform the work was carried out.

Using a Translation Memory improves the quality and consistency of the translations by allowing the translators to re-use the existing translations for the segments of the material which are repeated and perform concordance searches to find out how a certain term or sentence had been previously translated. When a Translation Memory is used, the time required for translating material is shorter and the cost of translation will be lower than translating using some other tools. If the word counts are very small, it is generally not cost-efficient to use a Translation Memory, since the localization companies usually charge a fee for creating, maintaining and using Translation Memories. Sometimes the TM fee is counted as hours, sometimes as a certain percentage of the total of the cost of translated words. This percentage is generally around 5-10%, and the amount of hours usually varies between 1-4. Of course, the localization companies will try and get money from anywhere they safely can, and it is only recent development that Translation Memory work is even invoiced from the customer, since using Translation Memories should be a benefit to all parties involved and the companies themselves recommend using Translation Memories to cut down costs. Before, when most localization work was carried in-house at the localization companies' premises, it was easy to place the Translation Memory in a central location in the company's network, and the Memory would be automatically updated as the translators translate. The translation becomes available for the other users as soon as the translator closes and saves the translation unit he or she is working on, but on the negative side, the translation is available before it has been validated, so any errors will be easily duplicated. Many people could use the same Translation Memory at the same time and get constantly updated strings for their searches. Translation Memory work in large projects has now presented new problems, since the translations are often carried out by several people in different locations even all over the world, and a central point – the localization company – needs to make sure the Translation Memories are as up-to-date as possible at all times and constantly hand out the updated Memories to all translators. This is generating extra work for the company, but at the same time, the extra work is actually caused by the localization companies themselves and the ways they handle localization work these days. Some companies do use public servers (with password) which host Translation Memories, but due to the number of people using them and network traffic their performance is often not as good as it should be.

Since the project was quite normal in terms of the source files and their native formats, the obvious choice to make was to convert the files to a format which supported using a Translation Memory when the conversion was necessary. The most widely used Translation Memory in the field and of the world is Trados Translation Memory, but there are dozens of other Translation Memory tools to choose from, for example, SDL International's SDL Suite, Star Transit and Multilizer. Practically all localization companies and IT-oriented freelance translators have at least the Trados Translator's Suite, so using it was an obvious choice, especially since experience has proved that it works really well for most localization tasks. Esselink (2000: see 20-21, 59-60, 166, 290-291) also strongly supports using Translation Memory tools for all the components but he does not recommend any specific all-purpose tool (see Chapter 11 of Esselink 2000 for more information) but correctly stresses that the tool need to be chosen according to the project in question.

The best tools to use with Trados Translation Memory were Microsoft Word and Trados TagEditor. This meant that the files needed to be converted into either of those formats.

Microsoft Word is the obvious choice for most straight-forward translation work, but translating HTML, SGML or other tagged format files is easier and best of all safer with Trados TagEditor, since TagEditor allows skipping and protecting the document's internal tags which are not meant to be translated or touched at all. The tool is still somewhat cumbersome when the tags need to be modified, changed or moved, but it is still very usable.

## 4.5 Preparing the material

The software files needed to be converted from .rc format into .rtf format in order to make the translating easier by using Microsoft Word together with the Trados Translation Memory. The .rc files can be easily converted into moderately protected .rtf format with a Microsoft Word macro TW4WinPrepareRC, which is created by Trados. This helps avoid overtranslation and breaking of the support codes. If the files are translated using a Translation Memory tool without using the macro, the word count would be 8,546 with 1,647 repetitions, while the macro would prepare the files so that the word count is 4,837 words with 296 repetitions. The difference here is that the control codes and other non-translatable strings would also be counted as translatable words even though they are not to be touched. Esselink also supports this workflow but does not discuss the money and effort the approach saves (2000: 59-60). For example, the following code sample would amount to 13 words with just 2 repetitions without the macro preparation, since every line would be a translation unit of its own, and "MENUITEM SEPARATOR" is the only repeated string. When the macro is used, there are just 3 words to translate. This is technically unfair to the translator, since "&Connect…\tEnter" is counted as one word only, but the following short example illustrates the importance of macro preparation:

```
MENUITEM SEPARATOR
MENUITEM "&Connect...\tEnter",          ID_FILE_CONNECT
MENUITEM "&Disconnect",                 ID_FILE_DISCONNECT
MENUITEM SEPARATOR
MENUITEM "E&xit",                       ID_APP_EXIT
```

The macro preparation had some problems. The conversion macro does not mark certain strings correctly, and it incorrectly cuts off software strings containing a hash (#) character and leaves the rest of the string outside the translation unit. The strings with the hash character needed to be checked manually and the styles changed in order for the whole string to be translated.

There was no good or definitive way to check out at this stage whether all the strings to be translated had indeed been included in the .rc files, but when the translated .rc files are compiled with the rest of the software and when the compiled software is tested, the non-localized strings will easily be located. These non-localized strings are strings or words that do not exist in the translatable files for one reason or another and thus are not available for translation. They are often called “hard-coded strings". It is quite common that hard-coded strings will be found in the software when the testing and the whole localization process progresses. These findings usually result in an update round or several update rounds containing some new material to be translated, and this usually takes place near the end of the project.

The documentation which was in SGML format did not need to be converted into any format when using Trados TagEditor for translating. The tool can read SGML files correctly if it is provided with a right .dtd support file which tells the tool how the different tags should be handled. The only thing that was needed to enable SSH's SGML documentation to be translated using Trados TagEditor was to change the default DTD file for SGML files to contain the line DOCTYPE=sshdoc under [GeneralSGMLSettings].

In general, the fewer times the files are converted in the localization and translation processes, the better, since more conversions lead to more potential errors in the process. One of the localization companies tried to invoice a couple of hours for SGML file conversion later in the process, but quickly admitted their “error” when I told them that no conversion was needed, since it had explicitly been made sure that they received the files in formats they could and should handle directly and easily with their translation tools. This is one example of the situations where it is useful for the project manager to know the workflow himself or herself, or otherwise these kinds of “errors” may be unnoticed and money will be lost.

When I had chosen the file formats to be used and prepared the files for translation, it was possible for me to analyze the material to get accurate word counts to be submitted to the localization companies for an exact quotation for the translations. Of course the localization companies re-analyzed the files at their office to make sure our calculations and word counts would be similar. This is normal practice; better safe than sorry applies to localization, too.

## 4.6 Workflow and quotations

Since the tasks I chose to be outsourced included only straight-forward translating and related internal tasks, I sent E-mails out to several different localization companies in Germany, France, and Japan asking for unit prices for software translation, Help translation and other tasks, including DTP, project management, updates, and Translation Memory management. My previous experience helped to know which companies were producing good quality work and which were not, which made it easy to pre-select the potential localization companies to be used for the project and concentrate just on them. In general, the companies Microsoft uses for their operating systems and application suites translations are good choices, since Microsoft has quite developed screening processes, and the companies who work for Microsoft generally have a good amount of contacts and a good number of employed people, in addition to being accustomed to use Microsoft glossaries, which are the basis of Windows application localization anyhow. After receiving the unit prices I chose the three localization companies to be used, one for each language, and validated the decision with the Project Manager of the product. I decided not to go and hand all the languages right away to a single localization company or office, since it would have resulted in less control on what was going on, and because the workload was relatively small, it was quite easy to handle the project using separate offices for the languages in question. The option to use a single contact point in the future still remained open, since all the chosen localization companies offered to function as a hub office for all the necessary and desired languages and also, as expected, stated there would be a discount if we decided to use a hub office for the projects in future. After dealing with the different companies it would be easier to decide which one would be the most qualified for the hub task.

There were separate proofreading contacts for German and French, and SSH's Japan office had promised to take care of the Japanese proofreading by having their people go through all the material. Again, experience had shown that due to competition in the localization business, if an another localization company was used to proofread text produced by some other localization company, the results were not necessarily adequate or even fair to the company which did the translation in the first place. When some localization companies are asked to carry out proofreading of texts translated by other localization companies it is common practice to be extremely strict with comments and corrections, since the companies try to win over the translation work from the customer in question instead of getting to carry out only the proofreading tasks. For them the best way to win the work seems to be to show the customer that the other company is not offering as good quality as the proofreading company would be able to offer. In order to avoid the possibility to encounter the trade politics between different localization companies, I decided to use qualified independent parties (meaning people whom I knew beforehand to have the skills) to carry out the proofreading for German and French, emphasizing that the proofreaders did not have to try and find mistakes where there were no mistakes, but rather concentrate on the real errors or areas for improvement.

Since the material to be translated was cutting-edge security software and because SSH Communications Security and F-Secure had a contract disagreement on the Secure Shell product, we needed the localization companies and all the parties involved to sign a Non-Disclosure Agreement (NDA) before any product material could be handed to the companies or proofreaders. While signing a NDA is normal practice for some parties which did sign it and send it back within an hour after receiving it, others wanted their lawyers to read through the agreements before signing them. The agreement in question was just to make sure no files or information would leak outside the localization company or translators, and if it did, there would be a price to pay for the mishap. The work in question did not consist of any non-public trade secrets as such, but as a precaution and for possible future projects, we needed those agreements signed in order to start working, so that was taken care of. When the parties had first faxed me the signed agreements – this was to save some time; they would send them to us via regular mail at the same time – we could go on with the project.

One of the localization companies asked us to arrange for on-site training at SSH's Helsinki office so that we would have paid for the flights, hotel, working hours, and other costs. While training is always of use and preferable, we did not have the funds or the resources for it. Also arranging all that for such a small project was not really sensible.

The workflow to use was simply to first prepare a small glossary containing the most technical and product-specific terminology for each language, have them translated and proofread, and then send out the files to the localization companies accompanied with the verified glossary. The main glossaries to be used were the Microsoft glossaries, since the product's platform was Microsoft Windows and the localization industry uses Microsoft glossaries for all Windows-related products or products running on Windows, when possible. This is general practice in the business among other reasons because consistent terminology makes it easier and faster for the end users to learn to use new software products. This also saves coding resources and effort, because most programs use some dialog boxes directly from the Windows operating system, for example, for file opening and file saving, and it would look and feel unprofessional to use different terminology elsewhere in the software for the same functions and items. Another plus is that Microsoft glossaries are also freely downloadable, and a terminology translations file with 12,000 English terms and translations in 59 different languages is available at the time of the writing (4.5.2007) for example from the Web site http://www.microsoft.com/globaldev/tools/MILSGlossary.mspx.

When the localization companies had the files, I set a sensible deadline, and an actual quotation on the work in question and commitment to the schedule were requested from the companies. Even though I had already prepared and analyzed the files, the localization companies naturally needed to analyze the files themselves before they could agree to any set prices or deadlines, as Esselink (2000: 438-439) states. The localization companies analyzed the files and sent me quotations on the work. It was a little surprising to find out that all quotations differed from each other in some aspects; some quotations did not have any Translation Memory work and the project management portion was different for all languages.

An example on the pricing negotiation is that for English-German translation the chosen localization company estimated that the price for average software would be around 0.25-0.30 euro per word and between 0.22-0.25 euro per word for the average SGML. After a short E-mail negotiation and letting them know the material was quite normal IT industry material, they offered 0.25 for software and 0.22 for the translation of the Help. When the project actually started and the project was assigned a Project Manager from the localization company – who was a different person from the one who negotiated the initial prices – the prices offered were suddenly 0.01 euro more expensive per word and all kinds of unnecessary extra costs were included, such as file conversion and handling. As stated earlier, this is unfortunately by no means uncommon when dealing with a customer who is thought to have no real knowledge on the processes, and if the project manager of the customer knows the processes and tasks, these can be avoided. Again, I told the localization company that the files had been prepared beforehand thus making conversions and tasks which amounted the extra work totally unnecessary. After this the localization company admitted having made a small mistake in the calculations and the process, and corrected the figures.

## 4.7 Translating

When all the scheduling and financial issues were in order, the next step was getting the "real" work done; translating the files. When the localization companies received the files from me either by FTP or by E-mail in the form of PGP (Pretty Good Privacy, more information available from www.pgp.com) protected Zip (more information available from www.winzip.com) packages, they unpacked the material. After this they analyzed the files and either prepared the files in the way they preferred – no preparation was actually needed or paid for, so it made no difference – or just handed them on to the translators as they were; with the glossaries, of course. To put it short, the translators translated the files, the localization companies' proofreaders read the files, the mistakes were corrected by the translators or by the company, and the files were handed back to me in the same format I sent them out in. The next step was to carry out some sanity checks on the files, making sure everything that should be translated was translated and that not much formatting or internal codes were lost in the process. Usually at least some code gets damaged for one reason or another, but in general that is easy to fix.

## 4.8 Proofreading, feedback

When the files were in order for technical aspects, I handed them out to the proofreaders with the glossaries and non-localized English counterparts of the files. The proofreaders I used were independent parties for German and French, and for Japanese I used our Japan office. For German and French, everything went quite smoothly, but Japanese presented quite a few problems. The original plan was that the people at the SSH Japan office would be the ones carrying out the proofreading, since they were both in the business and natives of the linguistic area. First their linguistic skills caused some concern, but it was found out that the proofreaders-to-be had linguistic backgrounds as well. After the proofreading process started, the SSH Japan office had hired an outside proofreader who only worked part-time on the project instead of using the in-house people.

Of course, using an outside party to carry out the proofreading was not a bad choice as such, and exactly the same was being done for French and German, but there were deadlines to keep, and the job was proceeding a lot slower than expected. One thing the Japanese language had in common with, for example, Finnish is that the terminology of the information technology and computing field was not totally unified yet. IT English has new compounds, neologisms and other new forms, which are sometimes difficult to research. In Japanese the source term can be recreated phonetically, such as English Ka-n-se-ru for English Cancel. (Handbook of Terminology Management 2001: 523)

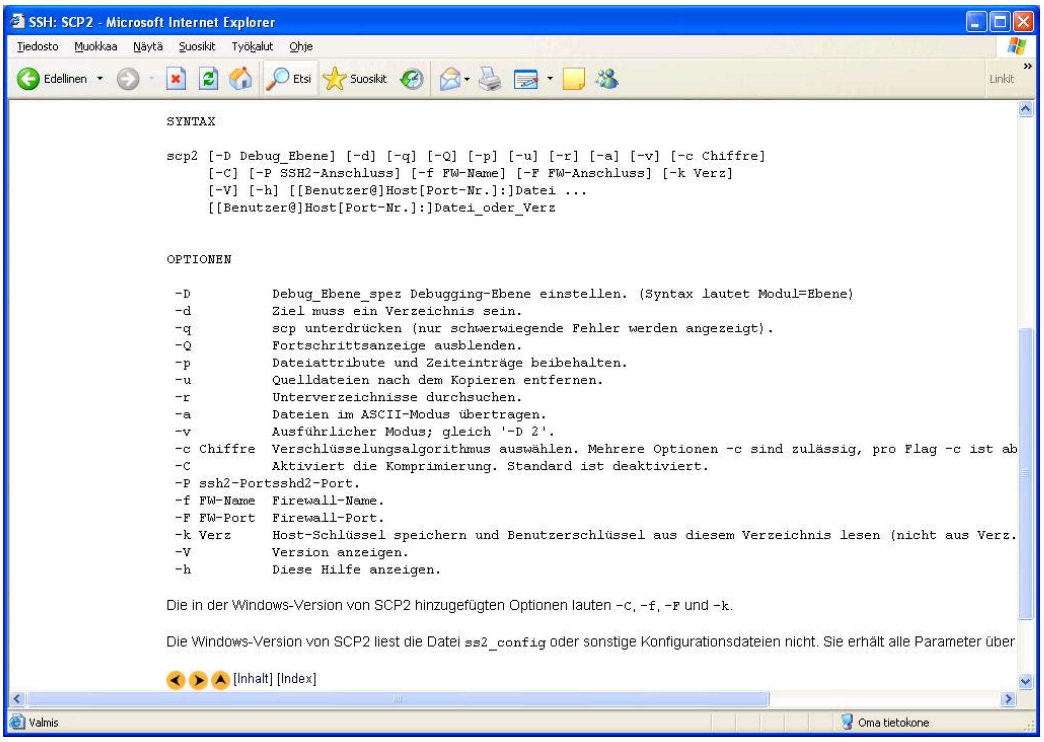
On top of that, there were a total of 3 people, 2 in Japan and 1 in Finland, involved in the proofreading process, which meant that there were a lot of disagreements since everyone had slightly different opinions on what terminology to use for even some of the quite basic concepts. Since the Japanese culture has a strong commitment to politeness, none of the proofreaders clearly said that something was wrong somewhere, but they rather said that perhaps some words or terms could be globally changed into something else. The other proofreaders sometimes agreed or suggested even more changes. This was somewhat frustrating for me, since I had absolutely no knowledge on the Japanese language or the conventions used there, but received comments in the cc: E-mail loop from three people suggesting changes or discussing changes. When the files were finally proofread once and handed back to the localization company with comments for corrections and modifications, one of the proofreaders sent me and the other proofreaders an E-mail suggesting that there could be some more general terminology changes to be made. Of course, this was too all too late. This is a good example on the cultural differences a project manager will have to deal with when managing projects. Because I had only very little knowledge on Japanese customs and habits, I did not know how to read between the lines and take firm enough action to steer the proofreading process better and just trusted their own workflow to be effective enough.

Eventually the Japanese software and the SGML documentation was proofread and corrected so that all the parties had agreed to the result for the time being. All in all, the Japanese proofreading took more than three times as long as for the other languages. In retrospect, I had given the Japanese proofreading too much freedom with the schedule and choices, but since Japan was the most important market of the target languages, I discussed this issue with the product management and we decided to let them have a longer workflow than the other languages. They were supposed to have “a product of their own” and have as much to say as possible on how the product would turn out to be, since they were language and business experts in the area, and the more they would have to do with the product, the better they would adopt it. They were the ones responsible for selling and marketing the product after all. We did not have enough experience on the cultural area or about the potential problems the different habits and also computing systems might cause, so the process was seen to be far more simple than it actually was. During the proofreading discussions one Japanese employee commented on the Japanese software needing to be using the specific Japanese character sets, but we did not understand what was actually asked, partly because of some language problems in communication and partly – most part, actually – because we (meaning the development team, Project Manager, and myself) did not have any experience on the requirements of the Japanese computing environment. We simply thought that it was just an issue of using the right fonts or something similar to that. I made a major mistake there by not doing enough research beforehand on the requirements for Japanese localization. The project manager should find out about the specific requirements and communicate them with the development team or escalate the issue further.

## 4.9 Compiling, building, processing the localized material

When everything was proofread and corrected, the material needed to be compiled, built and processed further. This meant that I would convert – or rather, try to convert – the translated SGML source files into HTML files for the first time with the internal *sshdoc* tool. This resulted in me having to fix quite a few errors in the conversion process. The internal tool had a built-in reporting on which line the conversion failed, and the files needed to be fixed using that information. Usually the conversion failures were quite straightforward, and the error was found in the same line which the tool reported, but at times the error did not present itself that easily, since the *ending* tag might have been on that reported line but the actual problem was with the *starting* tag which sometimes was literally hundreds of lines before the ending tag or even in a totally different file.

When the internal document tags were in order and the conversion process could be finished without the tool stopping at some point of the process, I needed to check the resulting set of HTML files for layout problems. There were quite a few small items which needed correcting, since, for example, some of the tags did not allow automatic and dynamic line wrapping in the browser but rather set a fixed line length. For example, German text is usually much longer than the English counterpart, so the result was a HTML file which did not fit in the screen.

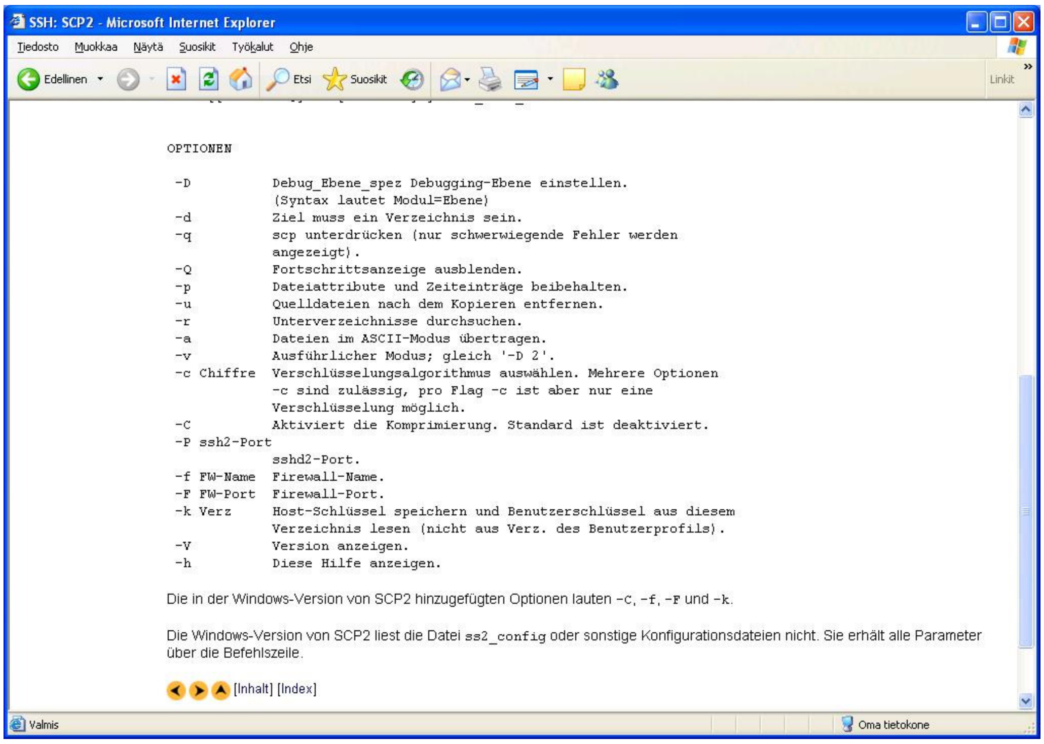

SSH: SCP2 - Microsoft Internet Explorer
Tiedosto Muokkaa Näytä Suosikit Työkalut Ohje
Edellinen
Etsi
Suosikit
Linkit
Valmis
Oma tietokone
[Inhalt] [Index]

```
SYNTAX

scp2 [-D Debug_Ebene] [-d] [-q] [-Q] [-p] [-u] [-r] [-a] [-v] [-c Chiffre]
     [-C] [-P SSH2-Anschluss] [-f FW-Name] [-F FW-Anschluss] [-k Verz]
     [-V] [-h] [[Benutzer@]Host[Port-Nr.]:]Datei ...
     [[Benutzer@]Host[Port-Nr.]:]Datei_oder_Verz


OPTIONEN

 -D           Debug_Ebene_spez Debugging-Ebene einstellen. (Syntax lautet Modul=Ebene)
 -d           Ziel muss ein Verzeichnis sein.
 -q           scp unterdrücken (nur schwerwiegende Fehler werden angezeigt).
 -Q           Fortschrittsanzeige ausblenden.
 -p           Dateiattribute und Zeiteinträge beibehalten.
 -u           Quelldateien nach dem Kopieren entfernen.
 -r           Unterverzeichnisse durchsuchen.
 -a           Dateien im ASCII-Modus übertragen.
 -v           Ausführlicher Modus; gleich '-D 2'.
 -c Chiffre   Verschlüsselungsalgorithmus auswählen. Mehrere Optionen -c sind zulässig, pro Flag -c ist ab
 -C           Aktiviert die Komprimierung. Standard ist deaktiviert.
 -P ssh2-Portsshd2-Port.
 -f FW-Name   Firewall-Name.
 -F FW-Port   Firewall-Port.
 -k Verz      Host-Schlüssel speichern und Benutzerschlüssel aus diesem Verzeichnis lesen (nicht aus Verz.
 -V           Version anzeigen.
 -h           Diese Hilfe anzeigen.
```

Die in der Windows-Version von SCP2 hinzugefügten Optionen lauten `-C`, `-f`, `-F` und `-k`.

Die Windows-Version von SCP2 liest die Datei `ss2_config` oder sonstige Konfigurationsdateien nicht. Sie erhält alle Parameter über

Image: An example of German text too long to fit in screen.


SSH: SCP2 - Microsoft Internet Explorer
Tiedosto Muokkaa Näytä Suosikit Työkalut Ohje
Edellinen
Etsi
Suosikit
Linkit
Valmis
Oma tietokone
[Inhalt] [Index]

```
OPTIONEN

 -D           Debug_Ebene_spez Debugging-Ebene einstellen.
              (Syntax lautet Modul=Ebene)
 -d           Ziel muss ein Verzeichnis sein.
 -q           scp unterdrücken (nur schwerwiegende Fehler werden
              angezeigt).
 -Q           Fortschrittsanzeige ausblenden.
 -p           Dateiattribute und Zeiteinträge beibehalten.
 -u           Quelldateien nach dem Kopieren entfernen.
 -r           Unterverzeichnisse durchsuchen.
 -a           Dateien im ASCII-Modus übertragen.
 -v           Ausführlicher Modus; gleich '-D 2'.
 -c Chiffre   Verschlüsselungsalgorithmus auswählen. Mehrere Optionen
              -c sind zulässig, pro Flag -c ist aber nur eine
              Verschlüsselung möglich.
 -C           Aktiviert die Komprimierung. Standard ist deaktiviert.
 -P ssh2-Port
              sshd2-Port.
 -f FW-Name   Firewall-Name.
 -F FW-Port   Firewall-Port.
 -k Verz      Host-Schlüssel speichern und Benutzerschlüssel aus diesem
              Verzeichnis lesen (nicht aus Verz. des Benutzerprofils).
 -V           Version anzeigen.
 -h           Diese Hilfe anzeigen.
```

Die in der Windows-Version von SCP2 hinzugefügten Optionen lauten `-C`, `-f`, `-F` und `-k`.

Die Windows-Version von SCP2 liest die Datei `ss2_config` oder sonstige Konfigurationsdateien nicht. Sie erhält alle Parameter über die Befehlszeile.

Image: The previous example, fixed.

Only after converting the localized files for the first time I found out that each language needed also a set of localized support files, which were used to generate the headers and footers of each page. This was not clear in the beginning, and no instructions stated that not all the on-screen text was placed in the SGML files handed out to be localized. When I tested the conversion process with the English files it was not clear just where each piece of text actually came from, but that was just one of the things that are learned in the process and hard to predict.

Underneath is an extract of the support file text that needed to be localized; specifically the text *SSH Home page* had to be localized for each language. All Help pages had that text on top of the page. No other part of the example code should be touched, and the safest option was to just extract all localizable text from the code and send just the necessary words to the localization companies for translating, since there is a great danger that an inexperienced translator touches the words he or she sees as localizable text but which actually are just code, such as words align, border, width and navigation in this example.

```
<TR>
<TD VALIGN="TOP"><IMG align=textTop alt="" border=0 height=16 src="gfx/box.gif"
width=15></TD>
<TD><span ID="navigation"><A href="http://www.ssh.com/">SSH Home page<BR>
</A></SPAN></TD>
</TR>
```

Cutting and pasting the actual text is much faster than checking the whole code and correcting the possible over-translations, not to mention that if the words are repeated, they only need to be translated once and all of them would be consistent throughout all the support files. Anyhow, a small text file had to be sent out to the localization companies for translation, and since there were only very few words, they invoiced us the minimum amount, which is somewhat equivalent to translating two full pages of text. This means that translating small updates is not at all cost-efficient, especially if there are numerous of small updates in the process. This is why it is useful to plan just a couple of collected update rounds in the process instead of sending out all missing strings and other items as they are found.

The French version of this extract was as follows:

```
<TR>
<TD VALIGN="TOP"><IMG align=textTop alt="" border=0 height=16 src="gfx/box.gif"
width=15></TD>
<TD><span ID="navigation"><A href="http://www.ssh.com/">Page d'accueil SSH<BR>
</A></SPAN></TD>
</TR>
```

When I inspected the localized files, I found out that both German and French translators had missed some text while translating the SGML files. This is illustrated by the non-localized text in the following example of command line examples from the client Help, where the translator had completely left everything untranslated:

```
ssh-keygen2 [-b bits] [-t key_algorithm] [-c comment_string] [-e file] [-p passphrase] [-P] [-\?] [-
h] [-q] [-1 file] [-i] [-V] [-r file] [-F file] [key1 key2 ...]

scp2 [-D debug_level] [-d] [-q] [-Q] [-p] [-u] [-r] [-v] [-c cipher] [-C] [-P ssh2-port]
[-f fw-name] [-F fw-port] [-k dir] [-V] [-h] [[user@]host[#port]:]file ...
[[user@]host[#port]:]file_or_dir
```

These types of omissions are quite common with inexperienced or uncertain translators, since it is often hard to be sure whether a command line example is actual text output – which is often not localized – or an example for illustration purposes. In this case this was the latter, so I sent the extracts of unlocalized text to the localization companies and they translated them. This was not an update but a correction, so SSH naturally did not have to pay anything for it. If the localization company's translator had doubts, they should have asked us. The correct translated version in French was as follows, with the translatable parts in *italics*:

ssh-keygen2 [-b *bits*] [-t *algorithme_de_clé*] [-c *chaîne_de_commentaire*] [-e *fichier*]
[-p *mot de passe*] [-P] [-\?] [-h] [-q] [-1 *fichier*] [-i] [-V] [-r *fichier*] [-F *fichier*]
[*clé1 clé2 ...*]

scp2 [-D *niveau_débogage*] [-d] [-q] [-Q] [-p] [-u] [-r] [-v] [-c *chiffrement*]
[-C] [-P *ssh2-port*] [-f *nom_pf*] [-F *port_pf*] [-k *répertoire*] [-V] [-h]
[[*utilisateur@*]*hôte*[#*port*]:]*fichier ...* [[*utilisateur@*]*hôte*[#*port*]:]*fichier_ou_répertoire*

After conversions, I found out that the *sshdoc* tool itself also contained some built-in text which needed to be somehow changed in the resulting files. One option was to modify the tool itself, but that would have been both time-consuming and potentially dangerous, in addition to which the modifications would have needed doing again for every new language in future. After some discussion and requests, the developers decided to add simple functionality which allowed using a command-line switch to specify a separate language definition file to use with the conversion. An example of such file was lang.fr for French. The file defines a simple replacement strings to be used when the files are converted and the text is inserted. On the left there is the original string and on the right, after the equals to (=) sign there is the target string. The file was similar for all languages, only the text on the right was changed. Underneath is an example of the body of the French language definition file.

```
"SSH Secure Shell 2 Windows Client Help" = "Aide de SSH Secure Shell 2 Windows
client"

"Contents"              = "Sommaire"
"Front page"            = "Page de couverture"
"Index"                 = "Index"

"Contact Information"   = "Contacts"
"Support"               = "Support"
"Feedback"              = "Feedback"
"SSH Home Page"         = "Page d'accueil SSH"
"SSH Products"          = "Produits SSH"

"Copyright &copy;"      = "Copyright &copy;"
"SSH Communications Security Corp<br>\nAll rights reserved.<br>" =
"SSH Communications Security Corp<br>\nTous droits réservés.<br>"

"Copyright Notice"      = "Note de Copyright"
"Figure"                = "Image"
```

Easily the biggest unexpected or unaccounted problem at this stage was to do with the names of the resulting HTML files. Our internal UNIX *sshdoc* tool split the large SGML files into small HTML files so that each page or Help topic of both Help systems was a separate HTML file. This created a total of 231 HTML files from 18 SGML files. The tool took the names for the files directly from the text of chapter or page titles, which naturally did not create any problems with the English files since this was the process which has been in use for years, but since the translated titles had some special non-English characters, Japanese of course consisting of almost nothing but them, the tool could not handle them correctly and the filenames looked strange. An example for this was that in the German Help system the topic *Spezielle Schaltflächen auf der Symbolleiste für die Dateiübertragung* generated a filename of *Spezielle_Schaltfl_chen_auf_der_Symbolleiste_f_r_die_Datei_.html*, which was both very long and had the non-English characters replaced with underscores (_), making the filename more than somewhat non-professional. Also, some of the non-standardly named Japanese files could not even be copied to the Windows file system at all from the tool's native UNIX file system. The easiest way to solve the problem at that time was to use command-line switch when running the tool. When this switch was used, the files were renamed in the conversion process to a universal neutral format, which used names such as *part-3-3-4.html* and *part-4.html* instead of, for example, *Manually_Editing_the_Authorization_File.html* and *Operation_Menu.html*. In addition to this, the Help compilation information contained the original English filenames for binding the set of HTML files into specific context-sensitive Help items and for the Contents of the Help, so the files had to be modified to make the information point to the right files.

The original English topic binding information looked like this:

```
ID_FILE_QUICKCONNECT=Quick_Connect.html
ID_PROFILES_EDIT=Edit_Profiles.html
ID_PROFILES_ADD=Add_Profile.html
ID_VIEW_CUSTOMIZE=View_Menu.html
ID_VIEW_RESETALL=View_Menu.html
ID_VIEW_PROFILESBAR=View_Menu.html
```

And the localized Help system was using a format like this after the file names had been changed to point to the right localized files:

```
ID_FILE_QUICKCONNECT=part-3-1.html
ID_PROFILES_EDIT=part-3-2-2.html
ID_PROFILES_ADD=part-3-2-1.html
ID_VIEW_CUSTOMIZE=part-7-4.html
ID_VIEW_RESETALL=part-7-4.html
ID_VIEW_PROFILESBAR=part-7-4.html
```

Technically this meant that the next step was to go through all the file references in various support files, check out which *part-x-x.html*-format file the reference was actually supposed to point at and then change the references. There was no easy, or actually fast, way of doing it, so the English Help system had to be compared against the localized Help system and make sure the right filename references were used. It was fortunate that all three localized versions used the same support files and all the file references were the same for all the localized languages, so the same support files could be used for German, French and Japanese Help systems. This problem was later fixed by adding a separate tag in the SGML files for the HTML filename, saving quite a lot of manual work in the process. Fortunately this only needed to be done once.

The CHM Help system had also a Content file which Microsoft HTML Help Workshop used when compiling the CHM Help file. Here is a short example of a part of such a file.

```
<OBJECT type="text/site properties">
        <param name="ImageType" value="Folder">
</OBJECT>
<UL>
        <LI> <OBJECT type="text/sitemap">
                <param name="Name" value="Introduction">
                <param name="Local" value="Introduction.html">
                </OBJECT>
        <UL>
                <LI> <OBJECT type="text/sitemap">
                        <param name="Name" value="Network Security Risks">
                        <param name="Local" value="Network_Security_Risks.html">
                        </OBJECT>
                <UL>
                        <LI> <OBJECT type="text/sitemap">
                                <param name="Name" value="Security of Internet Protocol">
                                <param name="Local"
                                value="Security_of_Internet_Protocol.html">
                                </OBJECT>
                </UL>
        </UL>
</UL>
```

It is easy to see from the above example how the chapter titles were used to generate filenames. The localized file below shows the parts which needed changing in *italics*. Both the chapter title and the filename had to be changed to correspond the French Help system for both content and format.

```
<OBJECT type="text/site properties">
        <param name="ImageType" value="Folder">
</OBJECT>
<UL>
        <LI> <OBJECT type="text/sitemap">
                <param name="Name" value="Introduction">
                <param name="Local" value="part-1.html">
                </OBJECT>
        <UL>
                <LI> <OBJECT type="text/sitemap">
                        <param name="Name" value="Risques en matière de sécurité
                        sur les réseaux">
                        <param name="Local" value="part-1-1.html">
                        </OBJECT>
                <UL>
                        <LI> <OBJECT type="text/sitemap">
                                <param name="Name" value="Sécurité de l'Internet
                                Protocol">
                                <param name="Local" value="part-1-1-1.html">
                                </OBJECT>
                </UL>
        </UL>
</UL>
```

The Help project file needed to be localized as well. The project file is in text format and it contains information about the files which are compiled, the title of the Help, the language settings of, for example, sorting rules and some other information on how the Help should function and how the visual elements should appear. It contains the mapping information, which dictates how the different Help topics are called from the software. It is safer to modify the file by using the Microsoft HTML Help Workshop application than to modify it directly with a text editor, even though it is somewhat easier to use a text editor, since it is faster to use and gives a little more control if one knows what to do. Here is an example of the beginning of the English SSHClientHelp.hpp file:

```
[OPTIONS]
Auto Index=Yes
Compatibility=1.0
Compiled file=SSHClientHelp.chm
Contents file=SSHClientHelp.hhc
Default Window=ssh2winclienthelpwindow
Default topic=index.html
Display compile progress=Yes
Full-text search=Yes
Index file=SSHClientHelp.hhk
Language=0x409 English (United States)
Title=SSH Secure Shell 2 Windows Client help

[WINDOWS]
ssh2winclienthelpwindow="SSH Secure Shell Windows client
help","SSHClientHelp.hhc","SSHClientHelp.hhk","index.html","index.html",,,,,0x63520,,0x387e,
[485,0,1235,600],0xb0000,,,,,,0
```

After this text the file contains a list of the HTML files the CHM Help will consist of after compilation and various mappings. All in all the definitions in the .hpp file are quite obvious, and there are only a few items to be localized. The corresponding French file shows the modified relevant parts underneath in *italics*:

```
[OPTIONS]
Compatibility=1.0
Compiled file=SSHClientHelp.chm
Contents file=SSHClientHelp.hhc
Default Window=ssh2winclienthelpwindow
Default topic=index.html
Display compile progress=Yes
Full-text search=Yes
Language=0x40c French (Standard)
Title=Aide de SSH Secure Shell 2 Windows client

[WINDOWS]
ssh2winclienthelpwindow="Aide de SSH Secure Shell 2 Windows
client","\SSHClientHelp.hhc",,"index.html","index.html",,,,,0x63520,,0x387e,[485,0,1235,600],0
xb0000,,,,,,0
```

The language of the CHM format Help system needs to be set for sorting purposes and the title of the help needs to be translated. In this example it is useful to notice also that the .hhk keywords file was left out from the French Help system because there were some major functionality problems with the keywords, and since the keywords weren't too relevant for this Help, this was not seen as a problem. It was easier to just leave out the keywords completely than to solve the problems connected with them for this version. I found out later that using a native French Windows with French language settings in the compilation computer would have solved the problem quite easily, but at the time even that was not easily accessible. This was a problem that was hard to solve, since the compilation tool, Microsoft HTML Help Workshop included the language setting to be used in the Help, which should have been enough by itself, but the non-documented fact was that only a native environment would have solved the problem.

The problems mentioned above are examples on the technical problems a project manager might need to solve while managing a project. Esselink (2000) does not concentrate on non-standard tools or workflows or their and the potential problems and their troubleshooting while Berkun (2005) gives proper emphasis on damage control and problem management as well, showing a more practical and realistic approach. I feel that any localization or project management guide should prepare the reader for handling problems at least in theoretical level, so the potential problems are not met totally unprepared and a proper buffer is created and preparation for extra effort is made.

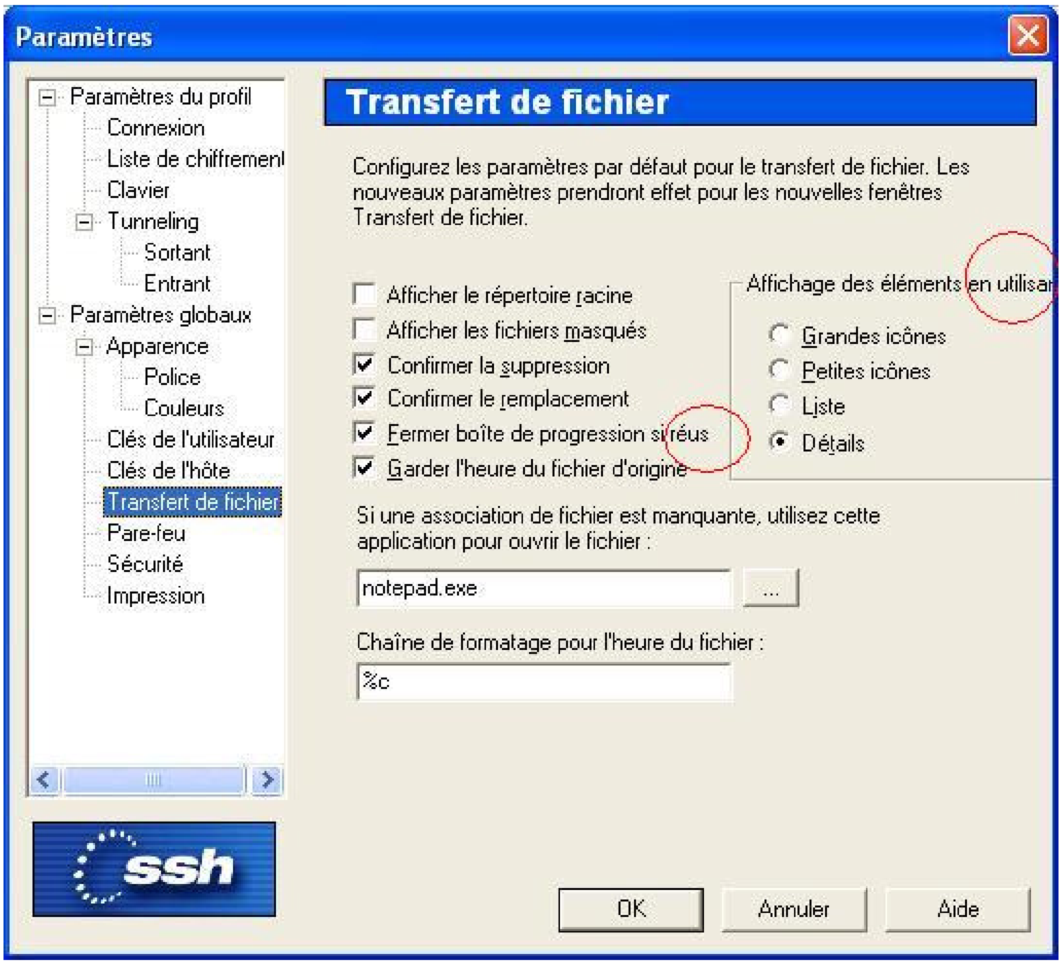
The support files were used only by the CHM format Help system, so no adjustments were needed to the standalone HTML Help system's support files. When the filename references were fixed to correspond to the actual filenames, and the internal Help language setting was configured for each language, the next step was to compile the Help system using the English screen shots at this time in order to see what the Help actually looked like when it was built. Fortunately there were no real problems at this time, and everything seemed to work out fine.

Then I checked the localized .rc files and fixed a couple of over-translations before the screen shots could be grabbed. Underneath is an example of a case where the words *dynamique* (*dynamic*) and *Statique* (*Static*) were over-translated. Those words are not on-screen text but internal properties which instruct how different elements were handled in the software. Unless the translator specifically knew that those were just control strings, he or she could not help but translate them – in this project no separate instructions were given to the localization company, since handling the controls was quite common knowledge in the field of software translation and the company should have possessed that information.

```
IDD_SETTINGS_SESSION DIALOG DISCARDABLE  0, 0, 357, 276
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Paramètres"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,188,257,50,14
    PUSHBUTTON      "Annuler",IDCANCEL,244,257,50,14
    PUSHBUTTON      "Aide",ID_BUT_HELP,300,257,50,14
    CONTROL         "Tree1",IDC_TREE_CATEGORY,"SysTreeView32",TVS_HASBUTTONS |
                    TVS_HASLINES | TVS_LINESATROOT | TVS_SHOWSELALWAYS |
                    WS_BORDER | WS_TABSTOP,5,5,90,228
    LTEXT           "dynamique",IDC_LABEL_SETTINGS_TITLE,107,5,245,16,
                    SS_CENTERIMAGE | SS_SUNKEN
    CONTROL         "",IDC_SHEET,"Statique",NOT WS_VISIBLE | 0x4,105,9,153,
                    150
    CONTROL         169,IDC_STATIC,"Statique",0xe,8,240,83,30
END
```

These over-translations caused the Microsoft Visual C++ application to fail parsing the .rc file correctly, and the application even refused to open the software project at all until the errors were corrected in a text editor. Since the translator had used a Translation Memory while translating, all the references of over-translated properties were translated in the same way, which made it easy to un-translate the strings by doing a couple of simple find-and-replace operations. This is yet another example of a situation where the project manager has to have technical knowledge on the process or tools or a qualified technical validator at his or her disposal quickly in order to solve the problems efficiently. Esselink (2000) does describe what for example Microsoft Visual C++ files look like and what they consist of, but does not, in my opinion, give nearly enough emphasis on the technical problem-solving.

When the necessary items were un-translated and the files could be opened in Microsoft Visual C++, I modified the layout of the dialog boxes in the application. The dialog boxes are simply windows of the software, containing textual and graphical items. When the application was used, it was possible to open the dialog boxes and check out the layout of the dialog box items, making sure there were no truncations or overlapping or misaligned items. The application allowed for automatic checking of duplicate hotkeys (mnemonics), which speeded up my pre-testing. Underneath is an example of truncated dialog box items.


Paramètres

Transfert de fichier

Configurez les paramètres par défaut pour le transfert de fichier. Les nouveaux paramètres prendront effet pour les nouvelles fenêtres Transfert de fichier.

- [ ] Afficher le répertoire racine
- [ ] Afficher les fichiers masqués
- [x] Confirmer la suppression
- [x] Confirmer le remplacement
- [x] Fermer boîte de progression si réus
- [x] Garder l'heure du fichier d'origine

Affichage des éléments en utilisar

Grandes icônes ( ) Petites icônes ( ) Liste [● Détails]

Si une association de fichier est manquante, utilisez cette application pour ouvrir le fichier :

notepad.exe

Chaîne de formatage pour l'heure du fichier :

%c

Image: Truncated dialog box items

Checking out the Japanese software was of course a little more difficult than German and French versions, since the truncated text segments were much harder to spot. This was because I had no knowledge of the language, so technically the only parts I could be absolutely sure of ended with a full stop. The items of the dialog boxes which were not full sentences and looked suspicious had to be checked separately by stretching the text frame of the string or auto-sizing it to make sure that no text was hidden. Also, dialog boxes and input fields within them will become around 25-30% wider when displaying Japanese characters, since they use more space than Western characters. (Unitech Systems' Unikaihatsu Software 2007) This made the software look somewhat different to the other language versions.

After I had made some corrections to the .rc files, I handed the files to the SSH Secure Shell development team. The developers built the localized software using an English Windows for German and French and a Japanese Windows for Japanese. After the software was built, the very first localized SSH Secure Shell clients were available for use. The next step was to check out the layout once more and fix some more layout inconsistencies, truncations and bugs in the .rc files and then hand back the .rc files for a rebuild.

Now there were quite “clean” versions of the localized SSH Secure Shell client which were good enough for grabbing the screen shots for the Help systems and the manual. The above examples, lengthy as they are, present some real life examples about the tasks and possible errors in a normal localization QA and fixing process. If the localization project manager does not have an available engineer in his or her team to check for errors and also to fix them, these items fall on the project manager's list of tasks and responsibilities.

## 4.10 Screen shots

When the software was built and had a clean and correct layout, it was time to grab the screen shots from the software to be used in the Help system and also in the manual. The graphics folder contained a total of 55 graphics files in .gif format. Those files were used for compiling the English Help system. When I examined the files, I found out that two of them contained no text at all. Since only the textual content of the software was changed when localizing the software, this meant that there were 53 graphics to grab or fake. That was the next step of the process. Since working with the software, checking out the layouts and using the software in actual work had given me some knowledge of the product, it was easy to locate most of the right dialog boxes and create the corresponding situations quite easily. The easiest way to grab a screen shot without specific tools for the task is to open the corresponding non-localized screen shot in an image viewer, open the software, browse to the right dialog box or screen, create the situation illustrated in the screen shot, press Alt+Print Screen, open a image editing application such as Microsoft Paint, and then paste the screen shot in the application by pressing Ctrl+V (or selecting Paste from the Edit menu) after which all that needs to be done is to save the screen shot to a different folder using the same name, file format and size the original screen shot was using.

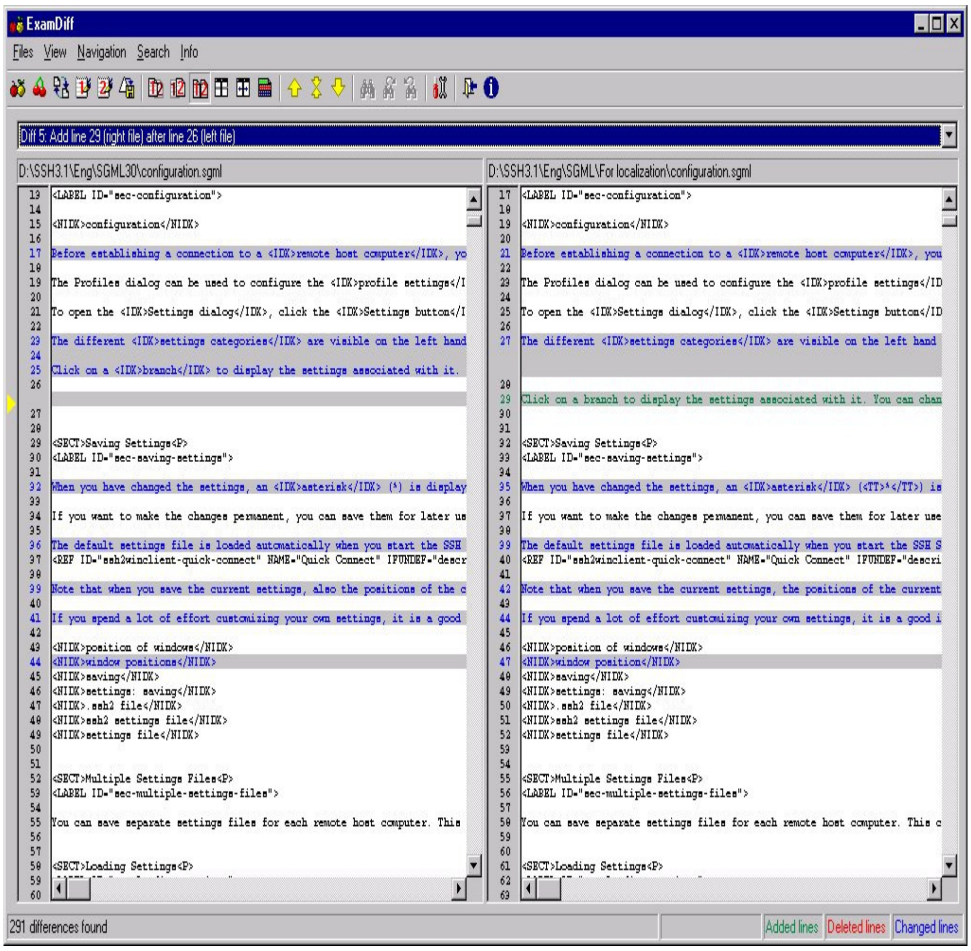
Most of the screen shots could be grabbed quite easily, but some of the screens used parts of a localized Windows operating system, which was because SSH Secure Shell used some Windows features for some user interface dialogs. For example, Print and Save As dialogs were taken directly from the operating system, and when the localized Secure Shell client was used in an English Windows, those appeared in English for the localized clients. There was a Japanese Windows operating system available, but no easy access to German or French Windows operating systems at the time, so those operating system specific screens needed to be faked. Grabbing the Japanese screens was not an easy task, but after starting with French and German screens and working with having the English version of the software open at the same time the task was not impossible. Some screens also would have required smart card readers and other accessories which were not easily available or obtainable, so it was necessary to fake parts of a few screen shots. In most cases, faking screen shots can be much easier and faster than setting up the whole configuration needed to produce a certain screen. There were some screen shots which could not be grabbed, since it was not possible to re-create the situation illustrated in the screens. This meant that the software files needed be searched for the strings and the text needed to be faked over the English text in the screen shots.

It was only after the process was over when I noticed that not all the screen shots were even being used in the Help systems or in the manual. Only 47 of the 55 screen shots were actually used, and the rest of the files were used in some earlier versions of the manuals and now had become obsolete. Many of those screen shots were the ones that were impossible to grab due to not being able to re-create the situation, which was not a surprise in hindsight. This meant that almost all the faking work was done for nothing.

## 4.11 Technical and linguistical QA, testing

After everything was in order what came to proofreading and correcting the software and the Help files, having the software built and the Help compiled, checking the built and compiled files and correcting a few layout errors, the software and Help were tested in-house by the QA team for technical aspects such as functionality and layout. In addition to this, the Japanese software was tested by SSH Japan office and also briefly by the Japanese expert in Helsinki for some functionality and language aspects. It was not surprising that the testing showed that some strings were not localized in the software. Part of the problem of non-localized strings was fixed when the developers noticed that some of the strings were coming from an third party library used by the software. The library fortunately had some localized language versions that could easily be used in the build process, thus replacing the English strings with localized ones. There was a small problem with some terminology, since the translator of the library did not use standard Microsoft terminology, but the difference was so minor that there was no use for trying to change the library, and most likely it would not have even been allowed. Since all three localized versions used the same source files and basic code, the same problems and omissions existed in all the languages, which made the problems faster to locate and fix.

The developers took the hard-coded strings from the bulk code and added them to the relevant parts of the .rc files, handing me a set of new files. Then I compared the old and new .rc files in a file comparison tool and located the new strings.



Diff 5: Add line 29 (right file) after line 26 (left file)

D:\SSH3.1\Eng\SGML30\configuration.sgml

```
13 <LABEL ID="sec-configuration">
14
15 <NIDX>configuration</NIDX>
16
17 Before establishing a connection to a <IDX>remote host computer</IDX>, yo
18
19 The Profiles dialog can be used to configure the <IDX>profile settings</I
20
21 To open the <IDX>Settings dialog</IDX>, click the <IDX>Settings button</I
22
23 The different <IDX>settings categories</IDX> are visible on the left hand
24
25 Click on a <IDX>branch</IDX> to display the settings associated with it.
26

27
28
29 <SECT>Saving Settings<P>
30 <LABEL ID="sec-saving-settings">
31
32 When you have changed the settings, an <IDX>asterisk</IDX> (*) is display
33
34 If you want to make the changes permanent, you can save them for later us
35
36 The default settings file is loaded automatically when you start the SSH
37 <REF ID="ssh2winclient-quick-connect" NAME="Quick Connect" IFUNDEF="descr
38
39 Note that when you save the current settings, also the positions of the c
40
41 If you spend a lot of effort customizing your own settings, it is a good
42
43 <NIDX>position of windows</NIDX>
44 <NIDX>window positions</NIDX>
45 <NIDX>saving</NIDX>
46 <NIDX>settings: saving</NIDX>
47 <NIDX>.ssh2 file</NIDX>
48 <NIDX>ssh2 settings file</NIDX>
49 <NIDX>settings file</NIDX>
50
51
52 <SECT>Multiple Settings Files<P>
53 <LABEL ID="sec-multiple-settings-files">
54
55 You can save separate settings files for each remote host computer. This
56
57
58 <SECT>Loading Settings<P>
59
60
```

D:\SSH3.1\Eng\SGML\For localization\configuration.sgml

```
17 <LABEL ID="sec-configuration">
18
19 <NIDX>configuration</NIDX>
20
21 Before establishing a connection to a <IDX>remote host computer</IDX>, you
22
23 The Profiles dialog can be used to configure the <IDX>profile settings</ID
24
25 To open the <IDX>Settings dialog</IDX>, click the <IDX>Settings button</ID
26
27 The different <IDX>settings categories</IDX> are visible on the left hand


28
29 Click on a branch to display the settings associated with it. You can chan
30
31
32 <SECT>Saving Settings<P>
33 <LABEL ID="sec-saving-settings">
34
35 When you have changed the settings, an <IDX>asterisk</IDX> (<TT>*</TT>) is
36
37 If you want to make the changes permanent, you can save them for later use
38
39 The default settings file is loaded automatically when you start the SSH S
40 <REF ID="ssh2winclient-quick-connect" NAME="Quick Connect" IFUNDEF="descri
41
42 Note that when you save the current settings, the positions of the current
43
44 If you spend a lot of effort customizing your own settings, it is a good i
45
46 <NIDX>position of windows</NIDX>
47 <NIDX>window position</NIDX>
48 <NIDX>saving</NIDX>
49 <NIDX>settings: saving</NIDX>
50 <NIDX>.ssh2 file</NIDX>
51 <NIDX>ssh2 settings file</NIDX>
52 <NIDX>settings file</NIDX>
53
54
55 <SECT>Multiple Settings Files<P>
56 <LABEL ID="sec-multiple-settings-files">
57
58 You can save separate settings files for each remote host computer. This c
59
60
61 <SECT>Loading Settings<P>
62
63
```

291 differences found

Image: Example of an output of a comparison tool

After I located the new strings, the next step was to create a new file consisting only of the strings which needed translation with some context when it seemed necessary and the file was sent out to the localization companies for translating. The reasons for me as the project manager doing the update this way were to keep the total effort as small and cost as low as possible. This is an example on the choices the project manager will face when preparing the material for translation, weighing the cost and effort, and selecting the solution which best fits the situation. In many cases the localization companies invoice 33% of the cost per word for going through the already translated strings found in the Translation Memory, which are considered to be 100% matches. For the cost for localization, this would mean that if a file set consists of, for example, 5,000 words and there would be a small update of 300 extra words, the costs would be as follows for the two main approaches:

Approach 1:
The new words are added into the original .rc files which were used as the source files of the localization. The English files with both the old material and the new material are sent to the localization company as they are.

Workflow:
The developers add the updated strings to the source (English) .rc files. The localization manager pre-processes the English .rc files for translation and re-translation, locking the strings and parts of strings which must not be touched. The localization company translates the whole file again including the old strings. The 100% matches are most likely auto-translated in the beginning of the translation process. All the strings are proofed and corrected by the localization company and by SSH. All the strings need proofing, since the translation might be context-sensitive at times, so a specific translation from the Translation Memory might not be accurate in all occurrences. Another major issue is that in addition to translating, the software needs to be completely re-sized and extensively tested for the layout as well, since all the possibly changed resizing and string location information of the .rc files is discarded with the already translated files, since that information is outside the localization units found in the Translation Memories.

Costs:
The tangible costs are those shown in the localization company's invoice and the proofreader's invoice. The translation costs for 5,000 old words and 300 new words would be 5,000 * 0.33 + 300 * 1.0 words, which totals 1,950 words. This means that in order to translate just 300 new words, one technically has to pay for over 6 times that many words. The proofreading would be of 5,300 words instead of just 300 words.

Approach 2: The new words are extracted from the updated files and a new file containing only the new strings and their placement information is created. The new file is sent to the localization company with the updated untranslated .rc file and the old localized .rc file for context reference.

Workflow:

The developers add the updated strings to the source (English) .rc file(s). The localization manager makes a comparison file from the old and the new source .rc files and locates the added or changed strings from the software. Then the manager copies the new and altered strings to a new file with some context where necessary. Then the manager pre-processes the newly created file for translation, locking the strings and parts of strings which must not be touched. The localization company translates the file. All the new strings are proofed and corrected.

Costs:

The tangible costs are those in the localization company's invoice and the proofreader's invoice, which in this case would be only the 300 new words and maybe some context as well. A minimum invoicing policy might apply here, but all in all the costs are far less than in the Approach 1 above. Of course this workflow generates more work for the localization manager, but often this is by far the best, safest and fastest option. In addition to leaving the already translated and proofed strings intact for linguistic aspects, this workflow also maintains the possibly altered sizing and placement information of the software, thus saving the extra work of re-checking the whole software and doing the possible changes all over again.

The strings were translated, proofread and corrected, after which the software was re-sized, re-built and tested again. Japanese was the only language to have an in-house linguistic quality assurance. German was linguistically checked by a native German freelance translator and French by an experienced Finnish translator specializing in translations from and into French. The technical testing, including functional testing and layout testing, was carried out in-house by SSH Communications Security's QA team members.

The layout bugs were fixed by me and the developers fixed the more technical bugs. Then the developers re-built the software for regression testing. In the regression testing the testers went through the reported bugs using the latest build of software and made sure the issues no longer existed and that no new issues have been introduced during the bug fixing. The regression testing presented no new problems, so the software seemed to be ready to be released.

## 4.12 The last pieces of the puzzle

When the software was tested and the Help was fully functional and everything looked fine, the work appeared to be finished for the three language versions. Originally the files handed out to me to be localized were in just a few directories on the company file servers, and the software seemed to consist of those only. Only at this stage I found out that the files would need to be placed in a directory structure for the CD disc the software would be released as and also that the cover text of the package and the CD disc label needed to be translated. Fortunately it did not take long to find out just what needed to be done before the package could actually be finalized, but if all the information would have been presented at the earlier stages of the process, this would have been taken care of with rest of the translations and some time and money would have been saved. I, as the project manager, should have asked for more information on the total scope of the product to be released myself at the beginning of the project and included all the items in the localization workflow at an earlier stage. The folder structure and the files on the CD disc showed that that there was one large HTML file to be translated and a set of support files, including some small .gif files and some header files for the HTML page. Fortunately the Licence Agreements included on the CD disc were to be left in English because of the company's policy on licences.

When I sent out a small update, a RTF file of 2,000 words, to be translated using Microsoft Word, one of the localization companies tried to invoice us 3 hours of Translation Memory work in addition to the word prices, even though it certainly must have been just one person doing the single-file translation, and therefore no real Translation Memory work was required as such. I sent the localization company an E-mail about the figure being much too high, and the Translation Memory task ended up consisting of only one hour. This is yet another example on why it is more than useful for the project manager to know the tasks and their efforts.

The CD disc also contained English installation binary files for AIX, HP-UX, Linux and Solaris and some related information files, but that was an another set of files which was to be left untouched, just the titles of the links to the files were to be translated. In addition to the text on the CD disc, there was also the text on the product's box cover sheet to be translated. I sent out the files to be translated into German and French to the localization companies, and the Japanese texts were translated by our Japanese expert at the SSH's Helsinki office. The box cover sheet texts were Microsoft Word files which were then sent to a printing house for printing. Everything was fine with the German and French versions, but the Japanese text presented problems with the printing house. The printing house could not handle double-byte text because of software incompatibility issues. While this was quite surprising, we just had to accept it, since that house was to do the printing in any case. The workaround they and our Japanese expert at the Helsinki office came up with was to have us, or more specifically our Japanese expert, print the Japanese characters using an extremely large font on A4 sized sheets and have the printing house then scan the characters, after which they just printed the text on the cover sheets as re-sized scanned images instead of normal text.

The CD disc also contained the Help systems as a PDF format manual file. The PDF file was generated from the same source SGML files as both Help systems, but there were some more problems in the conversion process. Surprisingly the conversions did not work at first for any localized version, even though the source files worked just fine for the conversion into HTML files. This was not a big problem for German and French, since the issue was solved by me fixing a few tag problems which were caused by the tool not being able to handle some text formatting tags inside the paragraph tags for some reason. Japanese, however, was a major issue here as well. The *sshdoc2ps* tool proved to have no double-byte support whatsoever, and the only output the tool produced was a few hundred screenfuls of error messages.

This was a big surprise, and there was no easy way to solve the problem. The developers simply stated that the tool did not support double-byte characters and that it would be too much work at the moment to add the double-byte support to the tool, especially since the persons who had been developing the tool previously were unavailable for the task. The Project Manager of the product stated that it would be acceptable to leave out the Japanese

PDF manual, since the users did have the functional Help system at their disposal. The solution for me, however, was to take the responsibility for not noticing this problem earlier and fix the problem somehow. I, as the project manager, should have tested the conversion with some dummy Japanese files or asked the development team whether those characters would cause problems. The only sensible way was to create a new and empty Word file, paste the HTML Help pages into the file one by one, insert the screen shots in the places where they belong to, fix the fonts and the layout and then try and print the resulting file as a PS file, after which the PS file could be distilled into a PDF file with Adobe Acrobat Distiller. This was hard manual work, but it could be done nevertheless. First I used English Windows and English Word in pasting all the Japanese HTML material into a Word file. Then I adjusted the layout a little on the English system, but the file did not look completely correct. I found out that a Japanese Windows with Japanese Word were needed in order to be able to use all the correct fonts, fix the layout completely and then to print the PS file.

After doing this, the next step was to try and distill the PS file and create a PDF file, but for some reason it crashed Microsoft Word now and then, and after having successfully printed a PDF file, the fonts would not display on a non-Japanese system. I tried this several times, with and without embedding the fonts in the Adobe Acrobat Distiller, using several different print settings, but for some unidentified reason the result was not what was expected; the font information would not embed into the file even though all settings indicated it should. The only way to create a valid PDF file in a short amount of time was to outsource the creating of the PDF file to the Japanese localization company which was used for the translations. I sent the Word file to the localization company. The Japanese project manager was very supportive, perhaps due to our good communication relationship or just politeness, and because of the time difference and the small amount of work needed for the process, the PDF file was ready by the next morning. The resulting PDF file looked even better than the original English PDF file, since the *sshdoc2ps* tool did not allow any layout changes and produced quite a plain outlook.

This is a very good example on the new and unforeseen issues which can arise during any stage of the localization process and which need to be dealt in creative ways (instead of “by the book") in order to succeed. For example Esselink's (2000) approach does not in my opinion based on real life experience, prepare the readers well enough for the variety of problems possibly arising in the localization processes, but only gives examples on generic problem prevention, not enough damage control as such. As Berkun (2005) argues, the project manager needs to keep an open mind and be prepared to take on the challenges caused by new processes or untested workflows, and also be unafraid to think outside the box when encountering problems.

## 4.13 Customer feedback and double-byte support

After the localized SSH Secure Shell 2.4 clients were released, there was no feedback at first. Even though the bad news usually travel fast, it was only months later when the company received feedback from the Japanese users on the lack of double-byte support in the client. Even though the software was tested in SSH Japan office after it had been translated, proofed and built, this issue was not brought to public attention until a very long time had passed from the initial release. At that time the issue was documented and a solution had to be planned and implemented. As Hoft (1995: 113) argues, the customers in the target country can provide many clues as to what features in a product make sense to them, what features are missing, how effective the information products are and whether the translation and localization of an information product was effective.

The problem with the Japanese Secure Shell 2.4 client was in the way double-byte input and output was handled in the terminal window and the file transfer window. The product worked perfectly with the Western sites and operating systems, but when it was used with Japanese file systems and operating systems, the result was the same as when using the core version of the client – it just would not do what it was expected to do in that environment. This meant that even though the user interface and the Help systems were all in Japanese, the program itself – the code – did not support using Japanese systems, which was a major concern. When the user viewed a Japanese system in the terminal window, all that was shown was garbled characters, and same happened when Japanese characters were entered in the window. Also, when the user connected to a Japanese site for file transfer or tried to transfer files which had filenames that contained Japanese characters, the names appeared garbled and the files could not be transferred to or from the user's system.

After a lengthy discussion between the Japan office and the Secure Shell project management, they decided that a Japanese coder would come over to the SSH's Helsinki office and program the double-byte Japanese support for the Japanese client. When this would be finished and working correctly, it would be implemented in the core product as well, thus adding to its features. This was an excellent idea. However, the actual development process took months instead of weeks, and even after when the coder finished working and announced everything to be in order, further testing by a Japanese localization company showed that there was still a lot of work to do – even though most of the characters were recognized correctly, there still were a few which caused problems with the client. Eventually all the problems were solved with the invaluable help from some very competent Finnish coders and support from the localization company.

# 5. Lessons learned and defining the ideal process

All in all, localizing SSH Secure Shell into German and French proceeded smoothly from the start of the process to the end of it. This was mostly due to the lack of need for double-byte support and also to the use of third party translators and proofreaders with clearly defined roles and timelines throughout the whole process.

## 5.1 Mistakes and neglected issues

In addition to small general and normal glitches in the workflow – for example some unforeseen and unaccounted for updates – there were two major mistakes or neglected issues, and they both had to do with the Japanese version. The biggest problems were caused by me, the project manager, not understanding the needs of the Japanese users and also partly the Japanese culture and the ways of working.

The Japanese proofreaders were all very polite towards me and each other, and at the same time they lacked strong opinions as such, most likely in order not to insult the other people taking part in the process. All the proofreaders just presented their suggestions for translating specific words in a specific way, not making too many definite corrections, even though they all seemed to have clear opinions on the terminology and style. I, as the localization project manager, should have taken a more active role in the process and just selected one proofreader for the whole project or to be in charge for the whole project, but instead I decided to give the SSH Japan office as much authority as possible on the linguistic aspects of the Japanese SSH Secure Shell version. I thought giving them the authority as early as at this stage would have made them work together and create a product which they could have adopted as more of their own product, since they were in charge of and lived in the market where the localized product was to be sold. However, this very issue only affected the release date of the localized product. It most likely did not affect the quality in a negative way, since all the proofreaders finally agreed on the result. There was no feedback of any kind on the linguistic quality as far as I know.

The worst issue by far was the lack of double-byte support in the Japanese SSH Secure Shell client 2.4. The lack of double-byte support was naturally not an issue for the German and French clients, since the client did support the character sets used in German and French, but for Japanese it proved to be disastrous. The SSH Japan office did mention the three different character sets used in Japanese, but since the client's user interface did support those in Japanese or Japanese-enabled systems, the issue was not noticed or thought about extensively enough. I, as the localization project manager, focused on the translation side, not the core code localization or internationalization, since I had no previous experience on double-byte systems besides just viewing them. I had previously managed software translation from English to Chinese and Japanese, but the customer had taken care of the double-byte coding issues without me or even the localization company I previously worked for taking any role in that process. The problem was in how the client handled double-byte characters sent and received in the terminal windows and in the file transfer window. This resulted in garbled characters and failed file transfers, both of which were unacceptable for the product behavior.

This double-byte support issue was fixed in SSH Secure Shell client version 3.0 with a lot of effort and testing, but until that the product was quite useless for the Japanese market as such. In addition to that, I as the project manager should have examined all the original product's material, using a real live product and its product CD instead of just separate batches of files on the company network or which were handed to me. Failing to communicate properly due to having insufficient experience on the product, the company, and also the software development field, I made unnecessary assumptions that my tasks somehow limited to handling the translation part of the project's components and that the development team or the Project Manager of the product would just hand me the material to be translated and tell me what needed to be done. Instead I should have taken a more active role in investigating further, figuring out the "hidden components" I now only encountered during the process, and also taking a more active part in the marketing and development issues as well.

## 5.2 Defining the ideal process

A) Needs and justification

The most important part of the process is to find out what the goal for localization actually is. It can be to localize the whole software product, user interface, help, manual, support files and possibly also some of the code or even to change the way the software looks and feels like or even how it works. However, in some cases it is necessary to localize only the Help and Manual instead of the whole product, either to save money or to make the turnaround time faster, since this approach will eliminate the need of testing the software after the core product has been tested. Ultimately the goal and the workflow of localization should be set by the needs of sales and marketing together with product management and development. Sales and product management together with marketing should create or announce the need for a localized product. There is no use localizing anything without an existing or a potential market for it, so market research needs to be carried out or there need to be requests from sales before making the decision to localize anything. The needs of the customers and the process requirements should help when deciding which parts of the product to localize. Naturally the cost of localization needs to be compared with the profits coming from the localized product. If the target market in question has special requirements, those need to be assessed at this stage. For example, double-byte support might be very expensive to implement and test, and it might require a lot of company's resources or even outsourcing the development for that part. The budget of localization depends very much on the needs and processes, in addition to number of languages the product is to be localized into. There is no set amount on how much localization for a product x into a language y will cost, but every situation and product needs to be separately analyzed.

There are services which can test and analyze whether products are prepared for entering a certain market and what additional work needs to be carried out, if any, before the product is good enough for a certain environment. (See for example BGS Japan 2002: 1-4.) Using such services can be of great use for a company with no previous experience of a market's requirements.

B) Commitment

Without support from all parts of the organization the localization cannot succeed. For example the development needs to commit themselves to the process as well. They need to give or allocate resources and support when needed, and the need is even greater if there is no separate localization management or organization in the company. A localization project manager can take care of the workflow and communication, but localization potentially generates quite a bit of work for the product development as well. The internal developers need to fix technical bugs and assist the localization in various other ways as well. Internationalization and double-byte enablement should either be totally outsourced and carried out by the development or performed completely by the internal development before localization is started. Many software publishers are moving towards total outsourcing models, where all globalization and localization-related activities are outsourced to large, multi-language localization vendors. (Esselink 2000: 8)

The different cultures inside the company need to learn to co-operate or at least co-exist peacefully as well. I noticed some attitude problems between different offices and different corporate functions, but nothing major. Based on personal discussions, the different country offices did not seem to quite understand each other's priorities and ways of operating, and the development did not seem value marketing and sales functions' ways (and the other way around). If the differences cause actual problems or bottlenecks in the processes, the localization project manager needs in addition to managing the project to act as an interpreter between the factions and make things work more smoothly.

## C) Workflow

When the goal is set and all parties have committed to the process, the next step is to work out the workflow. If double-byte support or some other non-Western systems or features are required, the requirements for that should be worked out at the very beginning. The best option is to consult a professional of the field or language area and have him/her work together with the developers or hire an agency to take care of assessing the coding requirements and other needs of the processes. If a company is using professionals in development and other functions, using a professional for double-byte enablement or coding is the best thing to do. All products are unique, so they have unique needs which are best assessed by a professional.

Before the translation work is started, the software and other components should have reached code freeze stage, which means that there are to be no changes unless something very critical comes up. If the localization needs to be started before the code is frozen, the project manager together with the development team need to set certain milestones and planned updates in the schedule, since the development work of the components will be going on while the files are being localized, and when the localization is finished, the source files have most likely changed quite a bit.

The first thing for the project manager to do after this is to make sure the software is prepared in a way which allows for easy localization. The project manager might have some internal engineers at his or her disposal for material preparation or inspection tasks. All the localizable strings should preferably be in as few files as possible, at least in the way that there are just a few files instead of possibly hundreds of small files. This makes translation, validation and project management easier. In addition, the linguistic (and technical) quality of the source text should be checked and improved if needed, since translating bad quality source text often results in unnecessary guesswork and incorrect translations. When designing and writing documentation it is very important to keep in mind that the material is to be localized. Of course the documentation should always take into account the expertise levels of the intended users, but when localization is added in the workflow, the technical writers should be even more careful on how the documents are structured and formed, because the localization tools allow only a little creativity in ordinary translation. Screen shots and other pictures naturally help understanding the message and the procedures, but they need to work together with the text. It is especially important to use the same or similar terminology in text, pictures and picture captions, since the translation can make the meaning even more unclear than in the original text, especially in technical texts. (Schriver 1997: 407-408)

D) Software – composite strings, date and time

One of the biggest challenges in software internationalization are composite messages. Composite messages occur where two or more strings are combined to create one message. When the translator works with composite strings, there is no way of knowing what the gender of the non-visible or unknown noun is in the message. (Lingo Systems 2000: 39)

The strings of the software should be divided between units as little as possible. Sometimes the software strings are divided in a way which only takes into account the English grammar and not the problems connected with translating the string into an another language with different grammar. Also, too many composite string structures make translation very hard at times. Here are some examples of composite messages:

```
File '%s' not found!
Sending %s...
Failed (%lx) processing: %s.
%s's desktop%s
%s of %s
ERR: Setup Failed Error Code: (hr) = %1x, installing: %s to %s destination code(%1x)
%s for %s failed in %s because of %s.
```

In the above examples *%s*, *%d* and *%1x* are placeholders which are replaced with a pre-defined string or value when the software is run. In most cases the strings are easy to translate, but in others, such as the last example, it is very hard to translate the string so that all grammatical rules are adhered to. More than that, the placeholders can also be replaced by words such as file or process names or file types, for example. In the string "*%s*'s desktop*%s*", the second placeholder is most likely for a plural construction. If the user whose name is inserted in the place of the first placeholder has more than one desktop, the plural s is inserted. This would form, for example, a string "Jack's desktop" or "Jack's desktops". Of course even this would be grammatically incorrect, if the user's name would end with an s, for example, “Bess's desktop”. This construction would not work for quite a few languages other than English, and not all English plural constructions use simply a plural s either.

The last string could actually be shown to the user, for example, as “*Deletion for folder failed in C:\Folder because of File Protection.*” or “*Copying for File.ext failed in Network Drive because of Network Error.*” This will be hard to translate correctly into a number of languages, even if the nature of each placeholder was explained in the translation instructions or file comments, since not only is the string non-descriptive but it also does not allow the translator to change the order of the placeholders. One way to solve some potential problems with the problems arising from the format used by the last example is to use different placeholders for different items, for example “*%1s* for *%2s* failed in *%3s* because of *%4s*.” This would allow the translator to change the order of the items in the string.

In general, if composite strings are used, they should be used very sparingly. If it is possible, the types of errors or locations and other items should not be inserted in the software from a table but rather be included in the string itself, even if it would create multiple strings which are almost similar to each other. The project manager should work with the development team and make sure this requirement is understood. (See also Microsoft Office XP Developer 2001.)

Nowadays most development is for 32-bit and higher operating systems, which means that the string lengths are no longer restrictive, but for 16-bit operating systems the string resources are limited to 255 bytes. If the English string is close to this limit, the translated string is likely to exceed the limit. Also, the dialog box buttons and other UI text should be developed so that the target text characteristics are kept in mind and enough space is left for the translation. (Lingo Systems 2000: 36)

When the software is prepared in the way that all localizable strings are in a couple of – preferably not in just one, and not in dozens of – localizable files and the strings of the software are prepared for easy localization with proper instructions and a minimum possible number of hard-coded strings, the project manager needs to convert or save the software files as for example RTF format to allow for easy localization. Then the files need to be prepared in a way which locks out the actual code which should not be altered. After the preparation, the files should be briefly scrolled through to make sure all the strings are divided in the right places. The macros I have encountered have problems with the hash character (#), since it is used in the comment string divider. The strings get often cut off after that character, and the style of the software string needs to be changed to allow full localization.

Date and time formats can vary from country to country. Some countries, like the US, use the 12-hour clock (for example 9:35 PM) while many European countries use a 24-hour clock (21:35). The formatting of dates is MM/DD/YY according to the American standard, while the European standard is DD/MM/YY. Similarly, the first day of week can vary as well. In America it is Sunday, and in Europe generally Monday. Abbreviating the days of the week should be avoided, since some languages have the same first letters for all days of the week. Most native operating systems have formatting routines to help with dates and times. (Lingo Systems: 38) In general, it might be a good idea to make the software use the operating system formats for date and time. This would reflect the user's preferences automatically and hopefully also reduce the coding efforts. Some restrictions may of course apply, but the development should be aware of them.

### E) Help and other documentation

When the software is in order, it is time to handle the Help system and other localizable material. Usually any type of Windows Help system is easy to localize, since the source files are most likely in HTML, SGML or RTF format, and those formats are generally easy to localize using the most common translation tools. The compilation of the original untranslated files should be tested to make sure that all the necessary files are available and everything works the way it should. The possible compilation problems are easier to solve before the Help files are localized, since then the changes need to be done only once instead of doing it once per language version. It is also very useful to have all parts a product component consist of translated and proofread at the same time by the same people. If there are other localizable components, such as a manual or various support files, the project manager should inspect the material to make sure all necessary files are included and that there is no other material included on the product installation CD or in the package that needs localization. It is also useful to find out whether there are any product brochures or press releases on the localized product versions to be published on the product and either have the same party translate them or at least make sure that the same terminology is used in both the brochures and press releases and the product itself.

When all the material which makes up the product and related to the localized version(s) of the product is determined by the project manager, it is a good idea to make a checklist on the various components and other materials to be localized or translated and double-check with the product team members that the list is accurate. The simplified list with just the main item names can for example look like this:

Software
Help and support files
Manual
Installation package and support files
Package & CD texts

After it is clear what material needs to be localized and translated and the files have been converted into formats supported by the translation tools, the project manager needs to analyze the material. This is easiest and safest to do against a Translation Memory. If there are no previous Translation Memories available, creating an empty Translation Memory and analyzing the files against it works well. The files can be analyzed as a large batch of files or one by one. If the files are analyzed as large batches, it is possible to see how much of the strings are repeated throughout the product text content. Also, when the whole number of words to translate is known, the project manager is able to prepare a schedule and translation order, taking into account the right translation order of components and the requirements of proofreading and correcting stages. This analysis information can be saved to a file and sent with the translatable material to the localization company doing the translation work for quotations and schedule estimates, since it will give quite clear a picture on what the project will consist of. The proofreading parties should also be notified well in advance about the workload and the schedule.

F) Translation tools & external parties to be used

The safest way to carry out a localization project is probably to use the Trados Translation Tools, since most localization companies and freelance translators are using them. The Trados tools are advanced and work well in most situations. A list of other Translation and Translation Memory tools can be found for example at http://www.translatum.gr/dics/translation-memory.htm.

When the tools to be used and the scope of localization are defined, it is time for the project manager to select a localization company to be used. The best way is to send out translation samples of the actual product's software and documentation to several localization companies and have the samples evaluated by an independent and qualified proofreader or a linguistic expert of the company. Then the proofreading results are reviewed together with the quotations from the companies and the most suitable localization partner is chosen. The chosen company should also commit to the schedule set by the localization project manager of the company.

G) Communication and File Transfer

Especially if the project in question is large, setting up or using a non-anonymous FTP server for file transfer is a good idea. Most E-mail servers have file or mailbox size limits, so in order to make sure that all the files get transferred regardless of their size or number, FTP server is the solution. If the files contain trade secrets or other sensitive material, the communication should be protected. Password-protected Zip files is one option, PGP key protection works too, and using a FTP server set up by either company with username and password protects the file transfer as well. The password and username are best sent via some other channel than E-mail, for example reading them on the phone should work. The localization companies do not usually get much of the source code for any product, so moderate protection measures should be sufficient.

If the project is large or the communication between the teams needs to be made direct, a communication map should be created. This map should have the names and contact information for the people responsible for each component or area, such as project managers, testers, QA people, technical writers and so on. Usually a localization project manager of the software company communicates with the people of the software company, possible outside assistants and the project manager of the localization company, who in turn communicates with the translators, QA people and other people of or hired by the localization company. In most cases this is sufficient, but if there are for example double-byte issues to be solved or double-byte coding to perform, it definitely is best to let the people who understand the issue communicate directly with each other; for example the developers of the company and the experts of an outside party do speak common language of code, and to avoid delays, misunderstandings or frustrations, they should be communicating with each other unless inter-person problems arise.

Before the translation begins, it is useful to prepare a style guide for the translators if the linguistic style of the company’s documentation has been established. The guide should be prepared by a proofreader or a linguistic expert from the company. Also, the glossaries to be used need to be selected. For Windows products the obvious choice would be Microsoft glossaries, and there are also Microsoft's style guides available if needed. Most localization companies are used to using Microsoft’s glossaries and style guides, so this should not present a problem when working with professional translators. If there is any useful reference material, it should be sent to the parties involved in carrying out the localization tasks. Too much or irrelevant reference material can be a burden, but a good package of background information, working software builds or other reference material will improve the quality and answer some questions before they are even asked.

### H) Translating, order of components

The components should, if possible, be translated so that software is translated in the very beginning of the process. The other components most likely refer to the software in some way, either by having screen shots of the software – most brochures do – or by using the software strings when showing the user how things are done or what the software is capable of doing. If other components are translated before the software, there is a good chance that the translator has to translate a software item he or she does not understand, and that will lead to guesswork. The software is the basis of the product, and there cannot be too many compromises in the translations. If for example a dialog box item is translated in a non-satisfactory way in for example the brochures or the Help, the item needs to be translated in a different way in the software, and this will lead to either the other components not corresponding to the software completely or changes needed to be carried out in the other components. It is good to keep in mind that it is useful to have for example the built un-translated software and its Help at hand when translating, since some context is often needed in order to produce accurate translations. When the software is translated and proofread first before working on the other components and the corrected strings are in the Translation Memory, the translators who translate the other components can use the Translation Memory for easy reference to find out how the software strings are translated. The order of other components is usually not very important for the process-related aspects, but the software component needs to be translated first. The company's internal requirements might dictate the translation order; for example a press release or some other publication might need to be translated early in the process in order to get it published on time. If one has access to some project management software, such as Microsoft Project, it might pay off to input the components and workloads in the tool.

When the order of components is figured out and the localization company and the proofreaders have confirmed their availability for the tasks and agreed to the set schedule, the translation work can be started. The localization company should already have the files by this time, since they have carried out their own analysis with the files. The best way to handle the components is to have them handled separately, but collected into one invoice, if that is possible, since then it is harder to inflate the project management or Translation Memory handling hours without it appearing over-invoicing. The localization should be started with the software, and after the software has been translated and proofed, other components can be translated while the software is already being re-sized, built and tested.

I) Processing the software after translation

As mentioned earlier, the project manager should send the prepared software files to the localization company before the other components, either with the necessary files needed to open the software project in a development application or just the translatable files themselves. A built un-localized version should be sent for reference, if possible. When the software has been translated and proofread by the localization company, the files can be proofread internally by the company's linguistic experts or an outside party assigned by the company. After the latter proofreading round, the comments or changes made by the proofreader should be sent to the localization company, who then make the changes and incorporate them into the Translation Memory. After this, the other files and components can be localized in any order according to the company's needs and preferences. The proofreading for other components should be carried out the same way as with the software, but the handback could be a staggered one, meaning that if there are for example 10 files with a total of 40,000 words to be translated, it might be useful to ask half the files back as soon as they are finished, since it will shorten the time needed for the company's proofing after the translation is finished and will also give room for useful feedback which can be implemented before the same potential errors are made throughout the component.

When the software files are ready linguistically, they should be opened in the development application for control code integrity verification, layout adjustment and hotkey checking. The development application usually reports the integrity errors in the code which was not to be touched, after which they are quite easy to correct. If the errors are not obvious, the code can be compared with the similar section of the original file which was sent for localization and then just copy the original code on top of the damaged code of the localized file. This comparison can be carried out manually or using a difference reporting tool, which compares two (or more) files and then shows the parts which have been changed, added or deleted. When the files can be correctly opened in the development application, it is time to check out the layout of the dialog boxes and duplicated hotkeys. 30 minutes of fixing and adjusting work might save a couple of hours of checking and reporting from the testers, so it pays off to go through all the dialog boxes of the software and check the overlapping or truncated items carefully before handing out the software to be built.

J) Testing

When the software is tested, the testers need to know what they are to do. This might sound obvious, but the project manager need to give them clear test case lists with the necessary steps for testing, instructions about what they should be looking for and on how and using which channel to report the bugs found. If it is possible to use native testers for the technical testing as well, it would be the best option, since they would carry out at least some linguistical testing at the same time with the functional testing. The actual linguistical testing or comprehensive double-byte testing of the product should be outsourced to professional testers, since they will concentrate on the right issues and do the work in the shortest possible time with the best possible results. Some issues can and should of course be tested in-house. If the trivial bugs are fixed before the builds are sent out to be tested professionally, the professionals can concentrate on the field of expertise not found within the company, linguistics or double-byte, for example, and it will save both time and money.

After the first round of testing is finished, the bugs need to be fixed. The layout bugs and related fixes can be fixed either by the localization project manager or someone assisting him or her. The developers do not necessarily have the right point of view on those details and their time usually is very limited, so it is best not to bother them with non-technical issues.

When the bugs are fixed and marked as fixed, the developers will generate a new build of the software. The list of fixed bugs is then used to check that the reported bugs appear as fixed in the new build, which is usually called regression testing. If a bug is fixed, it can be closed, and if the bug is not fixed or has caused an another bug, it is returned for fixing or a new bug is reported. Also, if there are significant updates to the core or localized software code, a new testing round or at least some additional testing is needed to make sure the product still operates the way it is supposed to.

K) Help compilation and checking

As mentioned earlier, the translated Help files can be sent back to the commissioning company in batches before all the files are translated to save time. In large projects this staggered handback will be extremely useful, but in smaller ones it can cause too much hassle to be of any real use. The proofreading process is the same as with software.

When the Help body text and support files are proofread and corrected, it is time for the project manager to try and compile the Help system. As with the software, the damaged control code is usually easy to spot with the compilation tools' internal reporting functions, and if the errors are not found easily, then manual checking or using a difference reporting tool will help solve the problems. When the Help is compiled, the layout needs to be checked to make sure the changed text length and other items do not cause problems and that no tags are added or moved in a way which changes the outlook of the Help pages. Depending on the Help structure it might also be necessary to go through the links of the Help or use a checking or validation tool, such as HelpQA (see Internet references) to check the links automatically. Special attention needs to be paid for the language setting, which dictates the sorting order of letters, among other things, and also the titles and other information in the support file set. If the sorting order is not correct with just changing the language setting, the Help might need to be compiled in a native environment, which could be Japanese Windows, for example. It is useful to test the compilation of the Help using the original non-localized screen shots, fix the possible problems and then re-compile the Help again when the localized screen shots are ready. After performing the compiling with the localized screen shots the layout needs to be checked again, but this is usually only a sanity check which is rather fast to carry out.

The localized Help systems usually maintain the binding information from the original version, which means that the context sensitive Help system maps the topics to the right parts of the software by using for example format similar to this: *ID_VIEW_CUSTOMIZE=View_Menu.html.* If the mappings are checked before compiling the Help, the Help system will function exactly the same way the non-localized Help functions and just some sanity checking will be needed. Usually the compilation tool reports if some mapping is not found, which makes the checking easier.

The new Internet browsers and operating systems are capable of detecting and viewing pages which are using special character sets. However, to make sure the Help pages are shown the way they are intended, the language and character set tags should be added to the Web page source code. (See W3C 2006 for more information on using the tags and a list of language tags.) The following is an example of language and character set tags of a Japanese Web page:

```
<html lang="ja">
<META http-equiv="Content-Type" content="text/html; charset=EUC-JP">
```

L) Screen Shots

Manuals and Help systems usually have screen shots embedded or inserted in them. Screen shots can be faked at any point of the process. However, the easiest way to take care of them is to grab them when the software is fixed and built. Most screen shots will be taken from the software, since they are used in the Help, but sometimes there are screen shots from the Help system as well, demonstrating its use. In that case the Help needs to be compiled before it can be actually finalized and re-compiled again after inserting the screen shots. The best way to get a screen shot is to use Windows' internal Print Screen function and paste the screen shots to an image manipulation application, after which the image files can be saved using the same names the original Help system uses. If some screen shots cannot be grabbed from the software because the software is not yet built and ready or for some other reason, the screen shots need to be faked. In practice this means opening the original screen shot in an image manipulation application and modifying all the on-screen text of the image to correspond the localized version. The texts can be found at least in the Translation Memory. If the image size or the number of colors in the image need to be changed, it is easy to do by checking the original image's properties and then changing the properties of the new image in the image manipulation application.

The localized screen shots should be copied to replace the non-localized screen shots in the Help compilation directory, after which they can be used in compiling the Help. They can also be used when creating the manual or they can be inserted in the manual file or files individually, depending on the requirements of the process.

M) Updates

Much more often than not localization projects have updates. This is because the development might still be going on when the localization is started, the code might be fixed or changed due to some late bug found before the release, or there might be hard-coded or otherwise unlocalized strings found. A good way for the project manager to handle the updates is to arrange an update round to take place quite late in the process, but if there are updates in the software, the update round should take place before the comprehensive testing of the software in order not to have to do duplicate testing efforts. If there are many testing rounds, not only does it create more management work, but it also costs much more than doing just one round.

N) Final checking

When everything has been localized, proofread, corrected and tested, all components have been built or compiled, the install package – if there is one – has been created and all the support files are placed in the same places where they are in the original non-localized version, the whole installation needs to be checked out in different configurations in order to make sure everything installs and works correctly both in systems where the product has not been installed before and where it has been previously installed, in addition to for example different Windows platforms where the product is supported on. The normal product testing will of course be carried out in those systems when the product is checked functionally and linguistically, but it is important to check the whole package again before it can be released.

## 5.3 Changes or improvements needed in the future processes

I have discussed localization at SSH with various people from various backgrounds, and it seems to be that most people do not quite grasp the concept of localization, or if they do, they do not quite understand the requirements set by the localization process or the benefits of localization. This is of course natural, since I cannot claim to have a thorough understanding of for example software development myself. What was questionable, though, was that the people who were responsible for making the decisions on whether a product would be localized or not and into which languages it would be localized, did not find it relevant to carry out specific market research, despite it being suggested to them and requested from them numerous of times. They did agree that it might be useful to conduct some market research, but the research never took place and the decisions were made based on other things, such has the sales figures of an another product of an another company in a different language area.

SSH Communications Security had offices both in Japan and German in addition to the Helsinki and Kuopio offices, and SSH Secure Shell was localized into Japanese and German, but as far as I know, it was marketed properly only in Japan. The Japanese office translated the brochures into Japanese by themselves, but the German office did not see it necessary to market the German product in German language, and the sales were very low. Due to the low sales, the German product was not included in all the localization rounds or updates, which in turn did of course nothing to boost the sales. Also when SSH took part in a trade conference in German, they did not see why the brochures should also be localized into German – or rather, when it was suggested to the people in charge, said it was a good idea but still did not do it – and used only English brochures instead. That situation could have been handled much better.

At the time of the writing, in 2006, SSH Communications Security now has clearly defined processes for all departments and functions of the company, so localization most likely will be considered in more detail in future. However, since there is no localization manager or coordinator in the company any more, nor does the task directly belong to anyone else in the company, localization will most likely be totally outsourced, which makes the number of internal tasks and efforts as small as possible.

Before resigning from SSH, I created a Localization Project Plan together with SSH's Documentation Manager to outline the localization process as a whole for future localization. The main improvement was to define localization as a sub-process for the software development process and thus clearly bind the localization tasks to the core project development tasks. This made more of the company personnel involved in the localization sub-project so that the Software Engineers, Quality Engineers, Project Managers, Product Managers and the local sales offices will each have specific tasks and responsibilities for the localization, while most of the responsibility was placed on the localization manager in earlier processes. This will result in more commitment and consideration in localization tasks throughout the company. The localization project manager still has the responsibility for main communication and workflow management, but the localization tasks and certain responsibilities are assigned more clearly than before. The Localization Project Plan contains company confidential information on SSH's internal processes and workflows, so I cannot attach it to this thesis nor quote it, but the main points and contents for the localization aspect are described in chapter 5.2.

# 6. Conclusion

When a software company is considering localizing an application, they need to have a definite need and a clear goal for localization. Depending on the size of the company, either the whole company or the relevant parts of it need to understand the requirements and effects of localization, agree with them, and support the localization processes from the very beginning to the end. Without adequate resources and commitment localization cannot succeed.

A project plan with definite dates and deadlines and task/responsibility assignments will help to eliminate problems, but every time localization is launched in a new company or environment, there will be new issues which cannot be predicted beforehand. I dare say that all localization processes contain surprises even to the experienced project managers due to the code and environments being different, technology and tools being constantly developed, companies having their own policies and processes, and last but certainly not least, to the fact that the work of the company is carried out by different individuals from different cultural backgrounds. This in part contributes to a constant learning process and helps to keep localization interesting; a “contact sport”, if you will.

Good localization project managers have enough technical skills to know each stage of the localization projects and processes, but more important than that is that they are able to gap the cultural differences between different groups of people, smoothen the edges, make all team members understand their position and effect in the project, and help them work in the best possible way towards a common goal. It is extremely useful for a localization project manager to understand what the localization project's processes consist of and the ways how a software company, the client alias the customer, and a localization agency see the project and handle different tasks.

All too often, localization has very tight schedules. A reference product is very seldom available for finding out all the necessary context in order to make justified decisions on translation, and because localizers often have to work with very strict guidelines and limits, the outcome is often more quantity-oriented than quality-oriented. Even though using various tools, such as Translation Memories, saves time, money, and effort, the tools have limits which, when the material is improperly prepared, can have adverse effects on the outcome. Tight deadlines and limited budgets also help prevent a “perfect result” from taking place, and localization often needs to settle for a result which is a compromise of a sort.

More than just a manager or a leader, the project manager can also be seen as an interpreter between different groups of people, different geographical cultures, and different company cultures and departments/functions in the company. In addition to being able to plan, communicate, and carry out the localization tasks, a localization project manager needs to have an open mind and excellent problem-solving skills in both technical and interpersonal levels in order to lead a successful project.

# Works Cited

## Printed sources

Berkun, Scott. 2005. The Art of Project Management. O'Reilly Media, Inc.

Blue Sky Software Corporation. 1999. Help Basics, RoboHelp Classic Version 2000 Getting Started.

Bowne Global Solutions Japan (BGS Japan). 2002. Bowne Global Solutions' Software Asianization Laboratory. Version 1.3.

Esselink, Bert. 2000. A Practical Guide To Localization. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Freivalds, John. 2002. Northern Enterprise, 5-6 2002: The Baltic Boom in Localisation, Lesser-known languages are starting to become a mainstay of localisation.

Goleman, Daniel. 1998. Working with Emotional Intelligence. Bantam Books.

Handbook of Terminology Management. 2001. Vol. 2. Application-Oriented Terminology Management. Compiled by Sue Ellen Wright and Gerhard Budin. Licia Corbolante and Ulrike Irmler: Software Terminology and Localization. John Benjamins Publishing Company.

Hoft, Nancy L. 1995. International Technical Communication. How to export information about high technology. John Wiley & Sons, Inc.

Hutchins, John. 2001. Machine Translation and Human Translation, International Journal of Translation, Vol. 13, No. 1-2, Jan-Dec 2001. Bahri Publications.

Lingo Systems. 2000. The Guide To Translation and Localization. Preparing Products for the Global Market. www.lingosys.com

McCrum, Robert, MacNeil, Robert and Cran, William. 1992. The Story of English, New and Revised Edition. Faber and Faber Limited and BBC Books.

Schriver, Karen A. 1997. Dynamics in Document Design. Wiley Computer Publishing, John Wiley & Sons, Inc.

## Internet sources and references

Adobe Systems Incorporated: Adobe PDF IFilter v5.0 Downloads (2006), "http://www.adobe.com/support/downloads/detail.jsp?ftpID=1276"

Cho, Raymond: Localize and Truly Go Global (2002), "http://www.ita.doc.gov/exportamerica/NewsFromCommerce/nc_LISA.html" (6.2.2007)

Finlex: Kuluttajansuojalaki, "http://www.finlex.fi/fi/laki/ajantasa/1978/19780038" (6.2.2007)

Julie Layden: Localization: Essential for Competing in the Global Marketplace (1997), "http://www.internationaltrade.org/aotm/0698.html" (6.2.2007)

Microsoft Corporation: Microsoft Office XP Developer, Localization Guideline for Your Code (2001), "http://msdn2.microsoft.com/en-us/library/Aa163852" (6.2.2007)

Microsoft Corporation: Microsoft Terminology Translations, "http://www.microsoft.com/globaldev/tools/MILSGlossary.mspx" (6.2.2007)

Peters, Tom: Pursuing the Perfect Project Manager, 19.4.1991

"http://www.tompeters.com/col_entries.php?note=005297&year=1991" (30.4.2007)

PGP Corporation, "http://www.pgp.com" (6.2.2007)

SDL International: HelpQA Overview, "http://www.sdl.com/products/helpqa.htm" (6.2.2007)

SSH Communications Security: About SSH, "http://www.ssh.com/about/company/" (3/2004)

SSH Communications Security: Main Page, "http://www.ssh.com" (6.2.2007)

The Localization Industry Standards Association: Frequently Asked Questions about LISA and the Localization Industry (2007), "http://www.lisa.org/info/faqs.html" (6.2.2007)

The Localization Industry Standards Association: The 2003 LISA Asian Globalization Resources Survey (2003) "http://www.lisa.org/products/surveys/asiasurvey.html" (1.5.2007)

Trados: Translation Memory (2004),
"http://www.trados.com/solutions.asp?page=1267" (3/2004)

Translatum: Translation Memory Tools, Computer Aided Translation, Other Tools,
"http://www.translatum.gr/dics/translation-memory.htm" (6.2.2007)

Unitech Systems' Unikaihatsu Software: Double-Byte,
"http://www.usindia.com/localization.htm" (6.2.2007)

W3C: Tutorial: Using language information in XHTML, HTML and CSS (2006),
"http://www.w3.org/International/tutorials/tutorial-lang/" (6.2.2007)

Webopedia Computer Dictionary: Regression testing (2004),
"http://www.webopedia.com/TERM/R/regression_testing.html" (6.2.2007)

WinZip International LLC,
"http://www.winzip.com" (6.2.2007)

Wikipedia: File Transfer Protocol (FTP),
"http://en.wikipedia.org/wiki/FTP" (6.2.2007)

Wikipedia: Secure Shell (SSH),
"http://en.wikipedia.org/wiki/Secure_Shell" (6.2.2007)

Wikipedia: Software versioning,
"http://en.wikipedia.org/wiki/Version_numbering" (1.5.2007)

Wikipedia: Telnet,
"http://en.wikipedia.org/wiki/Telnet" (6.2.2007)

**Project Manager's role in localization projects – Localizing SSH Secure Shell**

Suomenkielinen lyhennelmä

## 1. Johdanto

Projektipäällikön rooli lokalisointiprojekteissa ei ole helppo, ja siihen liittyvät tehtävät saattavat vaihdella suuresti projekti- ja organisaatiokohtaisesti. Erityisen kiinnostavan tästä roolista tekee se, että siinä voi myös yhdistyä eri rooleja. Globalisointi- ja lokalisointiasiantuntija Bert Esselink on kirjoittanut suositun lokalisointioppaan, jossa hänen lähestymistapansa lokalisointiprosessin projekteihin ja hallintaan on erittäin tehtäväpohjainen. Projektinhallinnan ja tuotesuunnittelun konsultti sekä luovan ajattelun opettaja Scott Berkun sen sijaan korostaa projektinhallintaa käsittelevässä kirjassaan ihmisten roolia ja merkitystä sekä projektityöryhmien ihmisten välisiä suhteita.

Tässä vahvasti käytäntöön perustuvassa tutkimuksessa tarkastelen oman työkokemukseni pohjalta sitä, miten nämä projektinhallinnan tehtävien ja projektipäällikön roolin näkemykset vastaavat käytäntöä todellisessa ohjelmiston lokalisointiprojektissa.

## 2. Projektinhallinta ja projektipäällikön rooli

Projektipäällikkö on tärkeä osa jokaista lokalisointiprosessia projektin koosta riippumatta. Hän toimii yhteyshenkilönä tai -pisteenä projektiin osallistuville tahoille, ja yleensä hänen vastuullaan on luoda projektin osien aikataulut ja tarkkailla projektin etenemistä. Yhdellä projektipäälliköllä voi olla samaan aikaan hallittavanaan useita projekteja tai vain yksi projekti, jos projekti on suuri. Useimmilla suurilla lokalisointiyrityksillä eli käännöstoimistoilla ja myös suurilla ohjelmistokehitysyrityksillä, joiden tuotteita lokalisoidaan jatkuvasti eri kielille ja alueille, on erillisiä ainoastaan projektinhallintaan keskittyviä projektipäälliköitä.

Projektipäälliköillä on yleensä useita tehtäviä ja vastuita, joihin kuuluvat tarjousten luominen uusille projekteille, projektien valmistelun koordinointi tai itse valmistelu, projektien suunnittelu, projektien taloustietojen seuranta, resurssien ja laadunvarmistuksen hallinta sekä muutoksenhallinta. Pienissä lokalisointiyrityksissä kokeneet kääntäjät toimivat usein käännöstyön ohella projektipäällikköinä ja lokalisointi-insinööreinä, joiden tehtävänä on tiedostojen ja osien tekninen käsittely. Esselinkin (2000) mukaan projektipäälliköillä odotetaan olevan seuraavat taidot: yleiset projektinhallintataidot, kokemusta lokalisoinnista tai monikielisistä käännösprosesseista, erinomaiset viestintä-, raportointi- ja neuvottelutaidot, hyvä tuotantotehtävien organisointikyky, kokemusta suunnittelusta, budjetoinnista, resurssienhallinnasta, projektinseurannasta, riskienhallinnasta ja laadunvarmistuksesta, tietoja lokalisointiprosessien sekä käännöstekniikan eduista ja rajoituksista, joustavuus ja mukautuvuus, vieraan kielen osaaminen (suositeltu) ja teknisiä taitoja (suositeltu). Hänen mukaansa onnistunut lokalisointiprojekti saadaan valmiiksi aikataulussa sekä noudattaen budjettia ja aiemmin sovittuja laatuvaatimuksia.

Useimmat lokalisointiprojektit alkavat tarjouspyynnöillä, joiden avulla toimeksiannon antava yritys valitsee edullisimman ja tarkoitukseen parhaiten sopivan tahon tekemään työn. Suurilla lokalisointiyrityksillä on erityiset henkilöt tarjousten laatimista varten, mutta pienissä yrityksissä tai jo luoduissa asiakassuhteissa tarjousten laatimisesta ja laatimiseen liittyvästä työstä, kuten materiaalin analysoimisesta sekä projektiin liittyvien työvaiheiden ja niiden työmäärien selvittämisestä, vastaa usein projektipäällikkö.

Kun lokalisointiyritys saa toimeksiannon, eli lokalisointiprojekti alkaa käytännössä, projektipäällikön tehtävänä on valmistella projekti, luoda projektiaikataulu, resurssisuunnitelma ja budjetti, järjestää lokalisoitavien osien valmistelu tai valmistella ne itse ja luoda myös viestintäsuunnitelma eli -kartta. Projektipäällikkö varmistaa, että projektin osat käsitellään oikeassa järjestyksessä ja että kaikki projektiin osallistuvat saavat kaikki heidän tarvitsemansa tiedot ja tiedostot. Kun projektin lokalisointityö on käynnistynyt, projektipäällikön on tarkkailtava prosessin etenemistä esimerkiksi tilaraporttien avulla ja varmistettava projektin pysyminen aikataulussa tarvittavien korjaustoimien, kuten resurssien lisäämisen avulla.

Lokalisointiprojektin onnistunut hallinta edellyttää kommunikoinnin eli viestinnän onnistumista. Projektipäälliköt viestivät asiakkaiden, kääntäjien, teknisen henkilöstön, lingvistien, tarkistajien ja kaikkien muiden projektiin osallistuvien kanssa. Hänen on pidettävä yrityksen sisäinen ja ulkoinen viestintä erillisinä ja varmistettava, että tietoja viestitään projektitoiminnan kannalta oikea määrä oikeaan aikaan, sekä kannustettava tiedonkulkua projektityöryhmän sisällä.

Berkunin (2005) mukaan projektinhallinta voi olla ammatti, työ, rooli tai tehtävä. Yrityksen sisällä voi olla vahvoja ryhmittymiä kehitys- ja liiketoimintaroolien ympärillä, mikä hidastaa toimintaa ja aiheuttaa turhautumista. Yrityksen sisällä olevat eri "kulttuurit" voivat aiheuttaa kitkaa ja väärinkäsityksiä, mikä aiheuttaa usein projektituloksen heikkenemisen. Projektipäällikkö ymmärtää projektista sekä liiketoimintanäkökulman että teknisen näkökulman ja auttaa tarvittaessa myös työryhmää ymmärtämään molemmat näkökulmat. Projektipäällikön on myös osattava toimia ja johtaa projektia muuttuvien tilanteiden mukaisesti.

Hyvän projektipäällikön on osattava johtaa sekä asioita että ihmisiä, eikä hän saa keskittyä liiaksi kaavioihin, taulukoihin, tarkistusluetteloihin tai raportteihin, vaan itse työryhmään ja työhön. Projektipäällikön tulee luoda hyvä sosiaalinen verkosto ja pitää tiedonkulku aktiivisena sekä toimivana. Lisäksi hänen pitää tehostaa aikataulujen noudattamista ja ymmärtämistä saamalla kukin projektin osallistuja näkemään, miten tämän työ vaikuttaa projektiin.

Sekä Esselink (2000) että Berkun (2005) ovat samaa mieltä siitä, että mikäli viestintä ei toimi ja onnistu, projekti ei ole menestyksekäs. Projektipäällikkö viettää runsaasti aikaa viestiessään yksilöiden ja ryhmien kanssa, joten hänellä on muita työryhmän jäseniä suurempi vastuu tehokkaasta viestinnästä. Projektipäällikön on tunnettava projektin tekninen puoli ja liiketoimintapuoli, mutta lisäksi hänellä on oltava taito viestiä projektiin liittyvien ihmisten kanssa niin, että eri kulttuurien väliset erot tasoittuvat ja kaikki tekevät työtä saman yhteisen päämäärän saavuttamiseksi.

# 3. Lokalisointiyritysten kanssa työskenteleminen

Lokalisointiyrityksiä on sekä yhteen kieleen keskittyviä että useiden kielten palveluita tarjoavia. Mikäli useita kieliversioita käännätetään saman yrityksen kautta, asiakkaan projektinhallintaan kuluva aika lyhenee ja vaiva pienenee, ja lisäksi kyseinen malli saattaa tuoda mukanaan myös alennuksia. Siitä saattaa kuitenkin seurata ongelmia tai laadun heikkenemistä, sillä suurten monikansallisten yritysten monikielisten käännösprojektien projektirakenne tuo suurimmat voitot työn asiakkaalta saaneelle toimistolle ja muut toimistot saattavat joutua tekemään työtä tavallista pienemmin hinnoin, mikä puolestaan kannustaa näkemään työssä mahdollisimman vähän vaivaa. Eri toimistojen käyttäminen eri kieliversioille antaa asiakkaalle ja asiakkaan projektipäällikölle paremman näköalan projektin joka osan tapahtumiin ja mahdollisiin ongelmiin sekä parantavat viestinnän laatua ja nopeutta, vaikkakin osa viestinnästä saatetaan joutua käymään useita kertoja, mikäli samat ongelmat ilmenevät jokaisen kieliversion yhteydessä.

Lokalisoinnin ulkoistamisesta on sekä etuja että haittoja. Jos yrityksellä, jonka tuote on lokalisoitava, ei ole palkkalistoillaan lokalisointiin pystyviä tai ehtiviä työntekijöitä tai jos sellaisten palkkaaminen yritykseen ei ole järkevää tai mahdollista liiketoiminnan kannalta, ulkoistaminen on ainoa vaihtoehto. Jos yrityksellä on käytettävissä omia lokalisoijia, he pystyvät hyödyntämään yrityksen muita resursseja ja tietolähteitä ulkoistettuja lokalisoijia paremmin sekä nopeammin, ja lisäksi heillä on todennäköisesti runsaasti taustatietoa yrityksestä ja sen tuotteista, mikä parantaa tuloksen laatua ja helpottaa tiedonhakua. Ulkoistaminen puolestaan kuluttaa vähemmän yrityksen omia resursseja ja saattaa nopeuttaa prosessia, sillä tällöin lokalisointityön tekevällä yrityksellä on yleensä käytettävissään huomattavasti toimeksiantanutta yritystä suurempi joukko työhön pystyviä henkilöitä.

Toimeksiantajan eli ohjelmistoyrityksen näkökulmasta lokalisoinnin tavoitteena on laajentaa yrityksen markkina-aluetta ja kasvattaa tulosta. Yrityksen on tuotteensa lokalisointipäätöstä tehdessä otettava huomioon lait, joiden mukaan tuotteen lokalisointi saattaa olla vaatimus sen myymiselle, tuotteen soveltuvuus lokalisoitavaksi, tuotteen aiotun käyttäjäkunnan kielitaito, lokalisoinnin kustannukset, yrityksen valmius tukea lokalisointia ja lokalisoitua versiota sekä lokalisoinnin tukemiseen vaaditut resurssit sekä niiden saatavuus.

Lokalisointiyritystä käytettäessä käännösten kielellinen laatu on yleensä hyvä, koska kaikkiin toimeksiantoihin kuuluu yleensä käännöksen lisäksi kielitarkistus, mutta tekninen laatu sen sijaan saattaa aiheuttaa ongelmia, sillä kääntäjät eivät ole kaikkien alojen teknisiä asiantuntijoita. Yleensä lokalisointiyritykset pyrkivät pitämään saman asiakkaan töissä saman projektityöryhmän kääntäjineen, sillä tuolloin kääntäjien oppiessa tuotteesta käännösprosessin aikana laatu paranee ja työ nopeutuu, eikä uusien kääntäjien perehdyttämiseen kulu aikaa. Mitä enemmän kääntäjiä yrityksellä on käytettävissä, sen joustavampi voi aikataulu olla, mutta mitä useampi kääntäjä työhön osallistuu, sitä suurempi riski epäyhtenäisestä tuloksesta on. Useimmilla lokalisointiyrityksillä on eri tehtäviin erikoistuneita työntekijöitä, kuten projektipäälliköitä, kääntäjiä, lokalisointi-insinöörejä, DTP-asiantuntijoita ja tarkistajia. Koska ohjelmistoyritysten aikataulut muuttuvat usein, myös lokalisointiyrityksen aikataulut muuttuvat. Valitettavasti usein lokalisointityön aloituspäivämäärän siirtyminen eteenpäin ei siirrä määräpäivää vastaavasti, sillä ohjelmistoyrityksillä on kokemusteni mukaan taipumus olla tiukempi ulkoistettujen toimintojen aikataulujen kanssa kuin sisäisten. Tämä saattaa luoda kiireen, joka puolestaan voi heikentää laatua.

Kääntäjät voivat työskennellä yrityksen sisällä tai freelance-kääntäjinä, ja näillä molemmilla ryhmillä on omat etunsa. Yrityksen sisäisillä kääntäjillä on yleensä melko kiinteät työajat ja heillä ilmenee freelance-kääntäjiä vähemmän ongelmia kommunikoinnin ja tiedostonsiirron suhteen. Freelance-kääntäjät voivat olla aikataulujen ja työaikojen suhteen erittäin joustavia, ja koska heidän ei tarvitse osallistua yrityksen toimintoihin, kuten kokouksiin, he voivat olla käännösmäärässä mitattuna huomattavasti sisäisiä kääntäjiä tuottavampia. He ovat myös lokalisointiyrityksille sisäisiä kääntäjiä edullisempia käyttää, sillä piilotettuja kustannuksia, kuten laitteisto-, työtila- ja terveydenhoitokustannuksia ei ole. Koska freelance-kääntäjille maksetaan yleensä käännetyn sanamäärän mukaan, heillä on sisäisiä kääntäjiä suuremmat paineet ja kannusteet pysyä tuottavina. Mahdollisten epäsäännöllisinten työaikojen vuoksi freelance-kääntäjien käyttämisessä on suurempi riski, sillä ongelmien ilmetessä ei apua ole välttämättä nopeasti saatavana esimerkiksi tiedostojen vioittuessa tai ohjeiden puuttuessa.

# 4. SSH Secure Shell -asiakasohjelman lokalisointi

SSH Communications Security -tietoturvayrityksen, jolla oli toimistot Helsingissä, Kuopiossa ja Tokiossa (Japani), SSH Secure Shell -asiakasohjelma päätettiin lokalisoida maalis-huhtikuussa 2000, ja minut palkattiin yritykseen huolehtimaan kyseisen tuotteen Windows-version lokalisointiprosessien luomisesta sekä tuotteen lokalisoimisesta saksaksi, ranskaksi sekä japaniksi. Lokalisoinnin projektipäällikkönä valmistelin projektin ulkoistetuille osille budjettiarvion. Analysoin itse tuotetta ja selvitin, mistä osista tuote koostuu (itse sovellus, online-ohje HTML-muodossa ja erillinen ohjejärjestelmä, joka oli online-ohje hieman toisessa muodossa), minkä jälkeen pyysin käännettävät lähdetiedostot Secure Shell -kehitysryhmältä. Ohjeistin kehitysryhmän projektipäällikköä kertomaan kehittäjille, että itse sovelluksen kaikki käännettävä teksti tulee olla .rc-resurssitiedostoissa, koska koko kehitystiedostopakettia ei voi lähettää lokalisoijille tietoturvasyistä. Koska lokalisoinnin työprosessi oli kehittäjille uutta ja he näkivät sen ensin pelkkänä ylimääräisenä tehtävänä, jouduin viestimään heidän projektipäällikkönsä välityksellä. Kuten Berkun (2005) toteaa, yrityksen eri funktioiden välillä voi olla kitkaa, mutta pystyin tehostamaan yhteistyötä ja sujuvoittamaan prosesseja vierailemalla kehittäjien huoneissa, keskustelemalla heidän kanssaan ja kysymällä heidän mielipiteitään asioista pelkkien kasvottomien sähköpostiviestien lähettämisen sijaan.

Sain kehittäjiltä kattavan paketin sovelluksen Microsoft Visual C++ -tiedostoja ja tuotteen tekniseltä kirjoittajalta dokumentaation SGML-tiedostot sekä joitakin tukitiedostoja. SGML-tiedostot (18 kpl) muunnettiin tukitiedostojen ja sisäisten työkalujen avulla HTML-tiedostoiksi (231 kpl), joiden kokoaminen CHM-ohjejärjestelmäksi edellytti lisäksi kuvatiedostot, hakemisto- ja sisällysluettelotiedostot sekä joitakin tukitiedostoja. Analysoin tiedostot tyhjää käännösmuistia vasten ja selvitin sanamäärät. Tämä vaihe toimi hyvin pitkälle Esselinkin kuvaamalla tavalla. Päätin käyttää kääntämisessä Tradosin käännösmuistityökaluja ja muunsin myös sovelluksen .rc-tiedostot Tradosin tukemaan RTF-muotoon Word-makron avulla. Muunnos aiheutti pieniä ongelmia, jotka pystyin korjaamaan manuaalisesti muuttamalla asiakirjan tyylejä. Tässä vaiheessa en voinut olla varma, että kaikki käännettävä sovellusmateriaali olisi .rc-tiedostoissa. On varsin yleistä, että vasta testauksen aikana käy ilmi, ettei kaikki materiaali olekaan ollut käännettävissä tiedostoissa. SGML-muotoisia ohjetiedostoja ei tarvinnut muuntaa, sillä Tradosin työkalut tukivat niitä sellaisinaan .dtd-tukitiedoston muokkaamisen myötä.

Kun olin valinnut käytettävät tiedostomuodot ja valmistellut tiedostot kääntämistä varten, pystyin analysoimaan tiedostot tarkasti ja lähettämään sanamäärät lokalisointiyrityksille tarjouksia varten. Valitsin aiempien kokemusteni perusteella muutaman yrityksen kullekin kielelle ja pyysin yksikköhinnat eri tehtäville. Sen jälkeen valitsin edullisimmat yritykset yhdessä tuotteen projektipäällikön kanssa. Lisäksi valitsin tarkistajat saksan ja ranskan kieliversioille ja sain tietää, että SSH:n Japanin toimisto oli halukas huolehtimaan japanin oikoluvusta, mihin tietysti suostuin. Lyhyesti sanottuna, neuvottelin hinnat ja lähetin tiedostot käännettäviksi ohjein varustettuina. Kun sain tiedostot takaisin, tarkistin ne ensin itse teknisten aspektien suhteen ja lähetin ne oikoluettaviksi.

Saksan ja ranskan suhteen kaikki sujui varsin hyvin, mutta japanin oikoluku ei tapahtunut odotetulla tavalla. Japanin oikolukuun osallistui kolme eri tahoa, joista kukaan ei toiminut varsinaisena auktoriteettina, joten keskustelua ja muutosehdotuksia oli runsaasti. Annoin oikolukijoille kuitenkin vapaat kädet, olihan oikoluku oman toimistomme järjestämä ja japani tärkein lokalisointikohteista, enkä tiennyt japanin kielestä tai kulttuurista juuri mitään. Tämä oli kuitenkin virhe. Oikoluku kesti yli kolme kertaa niin kauan kuin muille kielille. Tämä oli hyvä esimerkki kulttuurieroista, joita projektipäällikkö saattaa projekteissa kohdata. En osannut lukea sähköpostikeskustelun rivien välistä ja ottaa itse ohjaksia käsiini prosessin suhteen, vaan luotin heidän pystyvän siihen itse. Japanin oikolukukeskustelun aikana yksi Japanin toimiston työntekijä kertoi meille, että japanilaisen ohjelmiston on pystyttävä käyttämään japanilaisia merkistöjä, mutta ymmärsimme–me on tässä kehitysryhmä, tuotteen projektipäällikkö ja minä–tämän tarkoittavan vain oikeiden kirjasinten käyttämistä, sillä kenelläkään meistä ei ollut kokemusta japanilaisista tietokonejärjestelmistä sekä kaksitavuisten merkkien aiheuttamista ongelmista ja vaatimuksista. Tein suuren virheen, kun en ottanut etukäteen selvää japaniksi lokalisoimisen kaikista edellytyksistä. Lokalisoinnin projektipäällikön on selvitettävä lokalisoinnin kulloinkin asettamat erityisvaatimukset ja viestittävä ne kehitysryhmälle tai eskaloitava asia edelleen.

Kun kaikki oli oikoluettu ja korjattu, minun tuli käsitellä materiaali edelleen ja muuntaa tiedostot edelleen ohjejärjestelmiksi sisäisillä työkaluilla. Tässä prosessissa jouduin korjaamaan useita ongelmia, joita en pystynyt ennakoiman. Lyhyesti sanottuna jouduin korjaamaan tunnistevirheitä, koska käännösprosessi oli rikkonut tiedostojen rakennetta, pyytämään lokalisointiyrityksiä toimittamaan joitakin puuttuvia käännöksiä ja korjaamaan "ylikäännettyä" materiaalia takaisin englanninkielisiksi ohjaustunnisteiksi. Sisäisiä työkaluja jouduttiin muuttamaan prosessin aikana, sillä ne eivät tukeneet muita kuin englanninkielisiä merkkejä, ja jouduin tekemään luovaa korjaamista varsin runsaasti. Myös testaus osoitti joitakin virheitä ja ongelmia, joista kosmeettiset virheet ja ohjejärjestelmän virheet korjasin itse, puuttuvat eli kääntämättömät osat käännätin päivityksenä. Sovelluksen ongelmat korjasi kehitysryhmä. Japaninkielisen sovelluksen testasi toimintojen ja kielen suhteen nopeasti Japanin toimisto. Kaikki vaikutti olevan kunnossa, mutta sitten kävi ilmi, että en ollutkaan saanut kaikkia tarvittavia tiedostoja projektin alussa, vaan tarvittiin vielä julkaisupakettiin liittyviä käännöksiä ja muunnoksia. Minun olisi projektipäällikkönä pitänyt alussa kysyä tarkemmin projektin laajuudesta ja tutkittava itse valmista alkuperäistä julkaisupakettia sen sijaan, että tutkin vain palvelimillamme olevia tiedostoja. Tämä lisätyö oli onneksi varsin nopea tehdä, mutta jälleen japanin merkistöt aiheuttivat ongelmia, sillä sisäiset työkalut eivät tukeneet sitä. Sain kuitenkin korjattua ongelman tekemällä runsaasti manuaalista työtä tiedostojen suhteen. Myöhemmin kävi lisäksi ilmi, ettei japaninkielinen julkaistu tuote toiminut japanilaisten merkkien kanssa oikein, ja sen korjaamiseksi jouduttiin tekemään runsaasti ohjelmointityötä. Tuote oli hyödytön Japanin markkinoille, kunnes japaninkielisten merkkien tuki saatiin toteutettua.

Esselink (2000) ei valmistele lukijaa mielestäni tarpeeksi hyvin niitä kaikkia teknisiä ongelmia ja esimerkiksi eri kulttuureista aiheutuvia kommunikointiongelmia varten, joita kokemusteni mukaan lähes jokaisessa lokalisointiprojektissa projektipäällikkö ja lokalisointi-insinöörit joutuvat kohtaamaan. Berkun (2005) sen sijaan korostaa, että ongelmiin tulee varautua ja että projektipäällikön on oltava avoin sekä valmis näkemään vaivaa. Projektipäällikön tulee varautua ongelmiin ja varata niiden ratkaisemista varten aikaa myös projektisuunnitelmaan.

# 5. Opitut asiat ja ideaaliprosessin määritys

Pienten yleisten ja tavallisten ongelmien ja virheiden lisäksi prosessissa oli kaksi suurta virhettä tai huomiotta jäänyttä asiaa, jotka molemmat liittyivät japaninkieliseen versioon. Suurimmat ongelmat aiheutuivat siitä, etten projektipäällikkönä ymmärtänyt japanilaisten käyttäjien tarpeita ja osittain myös Japanin kulttuuria ja työtapoja. Minun olisi pitänyt ottaa aktiivisempi rooli japanin oikoluvussa ja nimetä yksi oikolukija prosessin vastuuhenkilöksi sen sijaan, että annoin prosessin edetä heidän omaan tahtiinsa. Tämä ongelma kuitenkin vaikutti vain julkaisupäivämäärän siirtymiseen hieman. Selkeästi pahin ongelma oli japaninkielisen sovelluksen kaksitavuisten merkkien tuen puuttuminen. Projektipäällikkönä keskityin kääntämiseen sen sijaan, että olisin kiinnittänyt tarpeeksi huomiota ydinkoodin lokalisoimiseen tai valmistelemiseen, sillä minulla ei ollut aiheesta aiempaa kokemusta käytännössä lainkaan.

Sen lisäksi minun olisi pitänyt tarkastella alkuperäisen tuotteen koko materiaalia käyttäen julkaistua alkuperäistä tuotetta sen CD-levyltä sen sijaan, että tarkastelin yrityksen verkossa olleita tai minulle lähetettyjä erillisiä tiedostoeriä. Koska minulla ei ollut tarpeeksi kokemusta tuotteesta, yrityksestä ja ohjelmistokehitysalalta, en osannut viestiä oikein ja tein virheellisiä olettamuksia siitä, että tehtäväni rajoittuivat saamieni projektin osien käännättämiseen ja että minulle kerrottaisiin selkeästi, mitä pitää tehdä. Tämän sijaan minun olisi pitänyt ottaa aktiivisempi rooli ja selvittää "piilotetut osat", jotka tässä projektissa kohtasin vasta prosessin kuluessa. Lisäksi minun olisi pitänyt ottaa aktiivisemmin osaa markkinointiin ja tuotekehitykseen liittyviin asioihin.

Ideaaliprosessissa lokalisoinnin päämäärä on selvillä, markkinatutkimus tehty ja tuote valmisteltu helposti lokalisoitavaksi. Tarvittavat organisaation osat ovat sitoutuneet lokalisointiin ja tukevat sitä yhdessä. Prosessin työnkulku valmistellaan huolella niin, että käännösmuistia käytetään kustannusten minimoimista varten ja tuotteen osat käännetään sekä käsitellään oikeassa järjestyksessä. Kääntämisen jälkeen osat oikoluetaan ja korjataan sekä testataan huolellisesti.

# 6. Päätelmät

Kun ohjelmistoyritys harkitsee sovelluksen lokalisoimista, lokalisoinnille on oltava selkeä tarve ja päämäärä. Yrityksen on ymmärrettävä lokalisoinnin edellytykset ja vaikutukset, hyväksyttävä ne ja tuettava lokalisointiprosessia alusta loppuun. Ilman tarvittavia resursseja tai sitoutumista lokalisointi ei onnistu.

Kattava projektisuunnitelma auttaa välttämään ongelmia, mutta aina, kun lokalisointi käynnistetään uudessa yrityksessä tai ympäristössä, ilmenee uusia ongelmia, joita ei pysty ennakoimaan tai ennustamaan. Kaikki lokalisointiprosessit tuovat yllätyksiä jopa kokeneille projektipäälliköille, koska koodi ja ympäristöt ovat erilaiset, tekniikkaa ja työkaluja kehitetään jatkuvasti, yrityksillä on omat käytäntönsä ja prosessinsa sekä erityisesti siksi, että työn yrityksessä tekevät eri yksilöt, joilla on erilaiset kulttuuriset taustat. Tämä osaltaan tekee oppimisprosessista jatkuvan ja auttaa pitämään lokalisoinnin mielenkiintoisena, eräänlaisena "kontaktilajina".

Hyvillä lokalisoinnin projektipäälliköillä on tarvittavat tekniset taidot, jotta he tuntevat lokalisointiprojektien ja -prosessien kaikki vaiheet, mutta sitäkin tärkeämpää on se, että he pystyvät tasoittamaan kulttuuriset erot eri ihmisryhmien välillä, vähentämään kitkaa, saamaan kaikki ryhmän jäsenet ymmärtämään vaikutuksensa projektiin sekä auttamaan heitä tekemään mahdollisimman hyvin töitä yhteisen päämäärän saavuttamiseksi. Lokalisoinnin projektipäällikön on erittäin tärkeää ymmärtää, mitä lokalisointiprojektin prosessit sisältävät ja käsittävät sekä miten ohjelmistoyritys eli asiakas ja lokalisointiyritykset näkevät projektin ja tekevät eri tehtävät.

Prosessien ja ihmisten johtajana toimimisen lisäksi projektipäällikkö voidaan nähdä myös tulkkina eri ihmisryhmien, maantieteellisten kulttuurien, yrityskulttuurien ja yrityksen osastojen/toimintojen välillä. Sen lisäksi, että lokalisoinnin projektipäällikkö pystyy suunnittelemaan, viestimään ja toteuttamaan lokalisoinnin tehtävät, hänen on oltava avoin ja omattava erinomaiset ongelmanratkaisutaidot niin teknisellä tasolla kuin ihmistenvälisellä tasollakin, jotta hän pystyy johtamisellaan tekemään projektista onnistuneen.