20 Years of Technical Writing at Altitude Software

Joaquim Baptista
Altitude Software
Alameda Fernão Lopes, 16 – 4 andar
1495-136 Algés
+351-21-412-9800
px@acm.org

ABSTRACT

Presentations by local technical writers have shown a surprising diversity of writing and content governance scenarios on mature software companies. Mature companies have unique business problems that require unique technical and human solutions.

On Altitude Software, writers struggle to control the complexity created by the organic growth of a mature software suite. As a response, feature-based documentation was rewritten in 2003 as task-based documentation, and is being rewritten again since 2010 into custom topic patterns.

Over time, Altitude Software learned to hire and train highly technical writers. Technical writers in Altitude Software became professional learners that must learn specific writing techniques, become experts in the product, and approach the background expertise of specific user audiences.

Categories and Subject Descriptors

K.6.1 [**Management of Computing and Information Systems**]: Project and People Management – *Life cycle, Management techniques.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Training, help, and documentation.* I.7.1 [**Document and Text Processing**]: Document and Text Editing – *Documentation management.*

General Terms

Documentation, Management, Theory.

Keywords

Writing, Writing patterns, Design patterns, Agile documentation, Agile management, uScrum, DITA.

1. INTRODUCTION

Reference technical writing books[16] describe guidelines for short-term consulting projects in great detail. Content strategy[17] provides guidelines for consultants that must maintain large sites. However, there is a scarcity of reports of the long-term processes required to maintain mature evolving products[12]. During 2013, a small geographical chapter of writers prompted several local companies to report their practices, including Altitude Software.

1.1 Altitude Software, the Company

Altitude Software was founded in 1993 as Easyphone. Twenty years later, the company has grown to 300 employees in 17 offices worldwide, with the headquarters and R&D based in Lisbon.

Altitude Software develops and sells a suite of client/server software called Altitude uCI (unified customer interaction). The software suite enables contact centers to handle a mix of outbound calls, inbound calls, inbound emails, inbound instant messaging, and workflow tasks. Besides selling the product, the company also sells related services, namely training, installation, customization, and maintenance.

Altitude Software has 1100 live installations in 80 countries, with 300.000 seats. Typical customer solutions range in size from a few agents to hundreds of agents, but the largest customers have thousands of agents.

1.2 Altitude uCI, the Software Suite

In early 2014, the software suite comprises 21 services, 26 graphical applications, 6 telephony gateways, 3 telephony switches, 6 application programming interfaces (APIs), and a proprietary scripting language. Some parts of the system require real-time performance, while other parts must handle large volumes of data efficiently.

Each customer uses a subset of these components, possibly integrated with different third-party telephony switches, database management systems, business applications, and legacy information systems. Customers often evolve their solution over time, adding software components for extra functionality, migrating between telephony switches, or replacing third-party systems. Customers often use the proprietary programming language to quickly build custom desktop applications for agents. As an alternative, customers can use one of the developer APIs to adapt their existing applications for contact center agents.

A small research and development team continuously adds and improves software components over time, carefully ensuring the continuous compatibility of all modules.

Documentation and training must follow the structure and evolution of the software suite. Writers must maintain a large body of related information published as documents and training courses, in a process similar to governing a large web site[17], and different from the traditional documentation projects[16].

1.3 Altitude uCI User Profiles

The documentation team identified the following typical user profiles, although job descriptions vary with the size of the customer, the culture, the business, and even the available people. Different users may use the same tool to perform different tasks.

Agents interact with other people through calls, emails, or instant messaging. Agents may also handle back-office work.

Supervisors train and assign agents to specific kinds of work, and then monitor agents, queues of work, operational indicators, and business indicators.

System administrators and system integrators install and configure hardware and software solutions, integrate third-party systems, and keep the solutions running. System administrators also manage the large amounts of data required and generated by a contact center.

Developers create customized graphical interfaces that guide agents through their work, present relevant information, and interact with third-party systems. Developers also create scripts that automate interactions of people with the contact center.

The sales process involves partners, sales people, and pre-sales engineers. Partners or services people typically install the initial solution, develop initial agent interfaces, and train an initial set of supervisors and system administrators.

2. DOCUMENTATION AREAS

Although each customer uses a subset of the software suite configured in a specific way, writers must understand all the possible variations. Therefore, writers must become experts in specific components or in the needs of specific audiences.

In a typical assignment, writers spend up to half of their time learning, a quarter of their time structuring what they learned in a way that suits the audiences, and the rest of the time writing and revising. In a sense, writers are more professional learners than professional writers.

Over time, the documentation team identified nine different areas that require writers to have specific skills and mindsets. Writers proficient in one area are not automatically proficient in the next area, although knowledge of some areas helps understand other areas.

2.1 Operations Area

Agents and supervisors use a unified interface that hides most of the technical complexity of contact centers. However, Altitude uCI introduces its own set of unique operational concepts, especially to monitor queues and the work of agents.

Writers struggle to understand these unique concepts, how they might be used in typical contact centers, and the relationships to other parts of the software suite. For example, functionality may depend on the availability of optional components, or on the specific configuration of generic components.

Writers develop topics for concepts, report references, graphical application references, and interface tutorials.

2.2 Systems Area

System administrators must understand complex dependencies between functionality and components, including the graphical applications used to configure and monitor components.

In this area, writers benefit from a background in system administration, computer networks, and computer telephony.

Writers struggle to understand many small parameters. Although each parameter is simple to understand by itself, each possible installation scenario requires the correct configuration of many related parameters.

The problem is compounded by the open nature of the suite, which caters for the ready integration of several telephony switches, different database management systems, and many varieties of desktops for agents.

Although customers only care for the specific components and configurations required by their contact center, writers are forced to understand all the options, and often need to understand how servers work internally.

Writers develop topics for concepts, architectural descriptions, interaction cases, install references, command and file references, graphical application references, and interface tutorials.

2.3 Scripting Area

Programmers use a proprietary scripting language to develop graphical applications for agent desktops, to customize how interactions reach agents, to handle interactions automatically, or to orchestrate a combination of tasks in a workflow.

In this area, writers need a background in programming languages and compiler technology, including grammars.

Writers struggle to learn the unique constructs of the proprietary programming language. More advanced writers must also understand the specific requirements of the different use cases for scripting in a contact center.

Writers develop topics for concepts, programming language constructs, and explained examples.

2.4 Connectors Area

Connectors are pre-developed components to integrate the Altitude uCI suite with specific third-party products, which complement or replace components of the suite.

Writers need knowledge of systems, to understand how the third-party product fits into the suite. Sometimes, the third-party product impacts operations, or requires specific programming.

Writers often face obstacles in collecting information about third-party products, since the R&D department often has a limited knowledge of the technologies involved.

Writers must create documentation that explores the integration with the third-party product without delving into details that are outside of the control of Altitude Software.

Writers develop topics for architecture, install, usage and, sometimes, API reference.

2.5 Telephony Gateways Area

Agents, supervisors, and programmers can simultaneously use telephony switches from different vendors. Telephony gateways adapt a common telephony model to the peculiarities of each telephony switch.

Writers struggle to understand how the common model maps to third-party telephony switches, each one with a divergent set of concepts, features, and limitations.

Writers often struggle to collect and validate information about third-party products.

Writers develop topics for concepts, architecture, interaction variation, install, and configuration reference.

2.6 Developer Area

Programmers can use one of six APIs in several programming languages to match their environment and integration needs. Three more APIs are planned.

In this area, writers need a background in programming, including some experience in the most popular development environments.

Writers struggle to learn the peculiarities of each API and each programming environment.

On some APIs, writers edit comments directly in the source code of the APIs themselves, and must be careful not to disturb the code.

Writers develop topics for concepts, install, usage, programming language objects, and explained examples.

2.7 Training Area

Company consultants and selected partners deliver the standard training courses, or customize the courses to customer needs.

After reaching a suitable understanding of the underlying area, writers struggle to create a sequence of concepts and tasks supported by a corresponding sequence of hands-on exercises.

As the software suite evolves, writers maintain the training courses by adding and changing lessons, and retesting all the exercises.

Writers develop topics for lesson plans, slides, exercises, solutions, and delivery guides. Training materials support 19 days of training covering the areas of operations, systems, scripting, and telephony gateways.

Writers occasionally deliver training to test new courses, to test significant changes, or to train new trainers and writers.

2.8 Illustration Area

Documents and training benefit from professional illustrations that reduce complexity by capturing variation in simple visual terms. For example, two families of images in the systems area capture architectural variations, expressed as 97 images supporting 192 topics with 47676 words.

Both the illustrator and the writers struggle to create visual metaphors for the unique concepts of the software suite.

2.9 Publishing Tools Area

The set of documentation and training materials is based on DITA topics[22][23] and published through the DITA Open Toolkit[14], with minimal customization[6]. Source materials are stored in plain directories as text and image files under version control[11].

The documentation consists of 1360k words in 6800 topics organized in 170 maps, plus 3800 images and other support files.

Publishing generates online and printable documentation in HTML, CHM, and PDF formats. Some CHM and HTML outputs serve as help for graphical applications.

Publishing also generates online and printable training materials in HTML and PDF formats. Some HTML topics are post-processed to generate HTML slides[25].

Since the documentation set is based on open formats, writers are not limited to any one editor or frontend tool. Therefore, some repetitive tasks are best tacked by applying regular expressions or classical Unix tools such as vi, sed, awk, perl, python, and make.

A custom Python script extracts comments and method signatures from C# source files to generate 500k words in 3700 reference topics for four API documents.

Another custom Python script extracts structure and labels from a large graphical interface to update 700 reference topics with 280k words. The same script locates and publishes tooltips for the graphical interface.

3. HIRING AND TRAINING WRITERS

During the last 17 years, the documentation team hired 19 writers, two editors, and one illustrator, represented as lines in Figure 1.

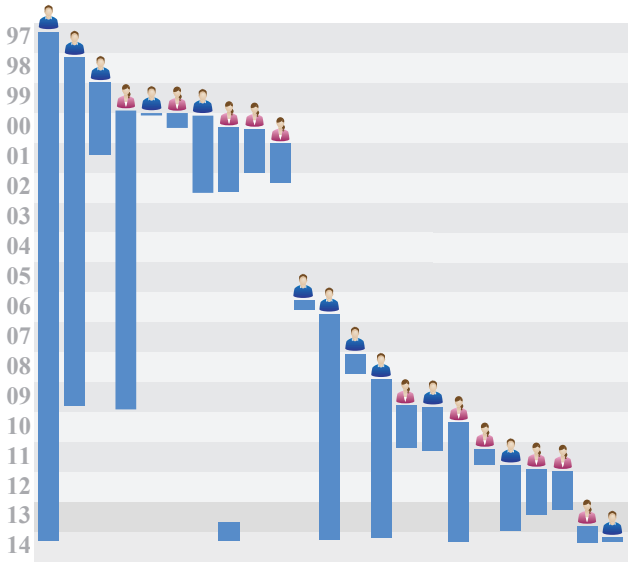

Figure 1. Team members as vertical lines over time.

Hiring has required significant effort, because Portuguese universities have no courses on technical writing, and also because Altitude Software has very specific needs. The result has been a very diverse team over time, where most team members migrated from other areas, countries, or cultures.

3.1 Sourcing and Initial Screening

The first step in hiring is to identify prospective candidates. The team has used four different strategies over time, with moderate success.

Newspaper advertisements bring all kinds of people, impose a huge filtering overhead, but may produce unexpected candidates.

Recruiters search their databases for candidates with the obvious keywords, but have a hard time in identifying candidates for the specific needs of Altitude Software, especially if they never worked as technical writers. In one case, the candidate convinced the recruiter himself.

Candidates applying for development may not want to work at anything else. Personal recommendations have produced strong candidates, although many recommendations lacked critical skills.

When analyzing curricula vitae for prospective candidates, the team first looks for a demonstrated ability to write in English, either through certified training, or through a writing hobby. Then, the team looks for the ability to handle technical detail. The best candidates have a Computer Science background, but Electronics or Physics may be good enough. The team occasionally considers bright candidates with less technical backgrounds.

3.2 Testing Candidates

Initially, the second step was to call promising candidates, explain technical writing, and call the candidates for an individual interview complemented with two English tests. However, the ability to write clearly in English has proved to be the major reason to reject candidates, so the team now starts by testing the candidates, often in bulk. The tests started as arbitrary exercises in 2000, but they succeeded in broadly classifying the relative abilities of the candidates. After seeing enough answers, the team learned to recognize both the common mistakes and the exceptional answers.

The first test asks candidates to write clear text for an illustrated procedure. The team examines the tests for precision, clarity, and structure.

The second test asks candidates to understand and rewrite a page of confusing text. The team looks for the ability to reason the confusion away, let go of the original words, and write a clear and precise explanation.

An optional test probes candidates that claim to be developers for the ability to modify a simple program after consulting an API reference.

3.3 Interviewing Candidates

The team calls the best candidates for an individual interview with the whole team, or with the senior writers of the team. During the interview, the team clarifies ambiguities in the curriculum, explains what is technical writing at Altitude Software, answers questions, and looks for motivations and for a fit with the current team. Depending on the background of the candidate, the team may also explore writing samples, or ask the candidates to take additional tests.

If the candidates have not worked as technical writers, it is important to stress that the bulk of the work is figuring out what to write, either during the first call or at the beginning of the interview.

3.4 Training and Coaching Technical Writers

The company hired the first writer for having participated in the students' newsletter while studying Computer Science. Initial training consisted in reading a few books[15][16][18][20]. Around 1997 it was not difficult to meet people with similar stories at technical writing conferences.

Writers without previous technical writing experience start by reading one or two traditional textbooks[15][20], replaced by DITA-aware textbooks[8][13][24] in 2011. New writers require in-house training[19] and coaching to develop their writing skills, as well as their ability to structure information in topics[30].

Writers depend on the coaching and guidance of senior writers until they have learned a significant part of the software suite, which typically takes a few years. When possible, having a small team of writers working in the same task accelerates learning.

Developing training courses also requires senior writers to master instructional design techniques[10].

The product training materials developed by the team had a huge impact in the learning effort required. In 2000, new writers took about one year to become minimally independent, and four years to become senior writers. In 2012, new writers started contributing after a few months, and approached seniority after two years in one of two documentation areas. Recently, design patterns have guided new writers into quickly writing topics that seamlessly meld with the existing documentation. Since 2007, the team has measured an average training and coaching effort of one month for each writer-year (see Figure 3).

At different times, the company hired seven writers with different levels of technical writing experience. Experienced writers struggled to maintain existing documentation, as opposed to writing new materials from scratch, which limited their ability to immediately reuse their writing skills. Experienced writers also struggled to learn the software suite.

The team has accumulated insights from books and conferences that are passed to new members over time, often accompanied by actual company stories. These insights become important guidelines to senior writers.

Three important books are known by the colors of their covers. The red book[1] shows how to identify the important strategic decisions, instead of just the urgent daily ones. The blue book[3] unifies software development with technical writing by recasting software development as a learning activity. The yellow book[9] dismisses the myth that "no one reads the documentation" which often haunts new writers.

To guide the search for new illustrations, the napkin book[26] suggests how to draw systematically, while the STOP report[30] suggests how to find parts of documents worthy of a drawing. Mental models[31] are useful to capture and structure tasks.

For team and project management, Hackos[16] is the unavoidable reference, even when the team follows methods conceptualized by content strategy[17] or when tasks cannot be estimated[3][29] at all. Schwaber[27] is the standard reference for Agile thinking. Peters[21] provides examples of excellence achieved by organizing work in less traditional ways.

3.5 Why do Writers Leave?

A quarter of the writers stayed less than one year in the team (see Figure 1). Half of the writers stayed less than two years in the team. One third stayed more than three years, becoming backbones of the team with senior knowledge of several documentation areas.

Team members have left the team for one of six reasons, in roughly the same amount: downsizing, fired for misconduct, or to pursue other careers abroad, technical writing abroad, technical writing in Portugal, or other opportunities within the company

4. DOCUMENTATION TEAM HISTORY

The twenty years of technical writing at Altitude Software can be broadly divided into six phases.

4.1 Marketing and Developers, 1993–1996

The first attempt at software documentation was to have the developers write help files for four applications, in Portuguese, because the first customer was in Brazil. The developers followed the Microsoft guidelines to create rich text format (RTF) files that were compiled into native Windows 3 help files.

Marketing wrote a 45-page overview document that was technical enough to provide a good starting point for all kinds of users.

When the software suite was sold in other markets, the company asked an English marketing assistant to translate the help files into English. However, the end result was to have weak Portuguese translated into worse English. The company concluded that it would be better to write the help files directly in English.

The company asked for a quote from IBM UK, which proposed to start by writing two manuals, estimated at 150 pages each, using either one writer for 12 months, or two writers for 6 months.

After realizing that the manuals would become obsolete during the timeframe required to write them, the company decided to start a writing department.

4.2 Clear Thoughts in Clear Words, 1997–2001

The company hired the first writer in 1997. By 1999, the team had grown to three writers, had rewritten most of the developer documents as small feature-based documents. The motto of the team was "clear thoughts in clear words."

Localizing the product and documentation into Japanese exposed the high costs associated with desktop publishing of translated documents in Microsoft Word. The expectation of localization to five more languages justified the adoption of XML. The company hired a consultant and created a custom XML environment, loosely based on the Linuxdoc SGML DTD[28]. Later localization efforts for French, Spanish and Mandarin already benefited from the significant cost-savings associated with XML.

In 2000, the company (Easyphone) merged with Intervento to become Altitude Software. A strong growth strategy saw the number of employees double every year, creating growth-related stress in every department. The documentation team grew to nine writers, with plans to hire five more writers in the next year.

The global economic downturn of 2001 forced the company to downsize, but not before the team moved the documentation to XML. Also, an internal survey on the needs and usage patterns of documentation users provided insights for further work.

4.3 Stable Agility, 2002–2004

In 2003, a scaled down team of three senior writers maintained the product documentation for an evolving product. The stable environment and the experience of the writers fostered significant innovation.

Following-up on an internal survey of internal users in 2002, the three senior writers rewrote the documentation around tasks aimed at the different audiences.

The constant evolution of the software suite demanded constant leaning from even experienced employees. After recognizing that technical writers became *temporary experts* on new features as they were documenting them, writers offered just-in-time training workshops[4] to share that fresh knowledge with sales, support, and services people.

The team was using a wiki since mid-2002 to keep extra detail, usually about third-party product. At the beginning of 2004, the wiki became the technical repository for company engineers.

After analyzing how the customers and the employees consumed the documents created by the team, by marketing, and by the training department, the team proposed a specific structure of wiki areas[7] that enabled other departments to adopt the wiki during 2005. The structure has mostly survived until today.

To efficiently juggle the efforts of writers against the business value of components and features, writers started to measure their time in 2003. In 2004, a yearly retrospective led the team to start managing the maintenance work in monthly sprints. A year later, the team would recognize its practices in the Agile principles and adopt the Scrum[27] terminology.

During 2004, the team developed custom software to generate API reference documentation, reusing comments in the code to create XML documents.

By the end of 2004, the organic growth of the documentation had stretched the custom XML environment beyond its design limits. The time to generate the HTML outputs reached 44 hours, and the company searched for a way to abandon the custom XML environment.

4.4 Training First, 2005–2008

Writers at Altitude Software typically spend about half their time leaning new components or new features. After the team argued that this learning effort could be reused, the team gained the responsibility to develop modern training materials[10] in early 2005.

The year 2005 saw the first public release of the DITA Open Toolkit[14], and training provided an excuse to replace the custom XML environment. The team adopted the toolkit with minimal customization[6], but adapted the HTML output to produce slides[25], and adopted Subversion[11] for version control.

The effort to convert the existing documentation exceeded the initial expectations, as the team realized that DITA was both a tool and a writing methodology.

In 2006, the team started to develop training materials, and slowly grew to five writers. Training fostered a deeper understanding of the software suite that fed insights back into the documentation, which gained better examples and more advanced descriptions.

The SIGDOC conference in Lisbon provided an excellent opportunity to reflect and report the Agile practices of the team[5], as well as the experience in using DITA[6] without any specialization.

4.5 Against the Complexity Wall, 2009–2011

After years of incremental evolution, R&D delivered a new and much larger release of the software suite that changed established ways of working, both for employees and for end users.

The signs of looming complexity had been accumulating in the previous years. Writers struggled to understand some parts of the software suite. The final proof had a senior writer struggle to create a unified install document in 2009 that was both hard to follow and difficult to maintain.

Writers had hit a complexity wall, as the task-based approach that had served the team since 2002 reached its limits. Two senior writers left the team, and the year ended with a team of mostly junior writers that required training from scratch.

2010 started the search for a new approach to writing. A top-down approach created a tentative structure that would be very difficult to execute by junior writers alone, but that clearly enumerated the issues that made the documentation complex.

A later bottom-up approach, however, succeeded in creating systematic reference topics for suitable parts of the software suite. These references successfully captured lots of small detail, by embracing complexity instead of creating dubious procedures full of minute detail.

The team rewrote most of the documentation for the previous release as references, which proved easier to maintain. Later, the team succeeded in quickly applying the same principles to large parts of the new version of the software suite.

At the same time, the new team recovered the principles of curriculum development and started to develop training materials for the new version, as a way to systematically collect and consolidate knowledge.

The whole approach proved its worth when turnover hit the team again in 2011: new writers learned very quickly to create relevant references from scratch. The team grew to three senior writers, three junior writers, and an illustrator.

4.6 Visual Writing Patterns, 2012–2013

Recognizing the power of design patterns[2] as a conceptual tool to structure documentation, the team started a systematic search for adequate patterns. The major insight was that writing patterns were not universal; the team would first need to understand its own unique problems. Understanding the forces in each problem was a major step in designing new writing patterns, and also in understanding the limitations of those patterns.

The illustrator became an important part of the search for design patterns. The illustrator had the objective of creating images that, besides being decorative, provided intuitive insights into the documentation. The search for interesting images melded with the search for design patterns, and advances in each area influenced the other area.

The result was the creation of visual models, that is, families of illustrations that capture important variations of one or more concepts. Writing patterns explained and expanded the visual models, using the images for insight and the text for detail. For example, two visual models originated 97 images that supported 192 topics.

At the end of 2013, with an ever-growing software suite, the team had developed reasonably complete documentation and new product training materials. Most of the documentation followed clearly identified writing patterns.

Some writers struggle to learn writing patterns, which are actually patterns of thinking that streamline the solution of a family of problems. Those writers face the complexity wall again with bare conceptual hands, and usually create inferior results.

4.7 Summary of Volume, 1999–2014

Over the years, the documentation grew from hundreds to thousands of pages (see Figure 2). The years in the figure correspond to the last update of major versions of the product, because major versions usually gain functionality over time.

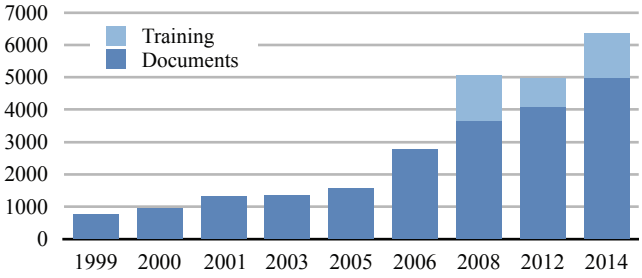

Figure 2. Number of pages of major product versions.

The values for 2006, 2008, and 2012 refer to significantly different versions of the documentation for version 7, while the value for 2014 refers to version 8. In 2012, the refactoring of documentation topics precluded their reuse in training materials.

4.8 Summary of Effort, 2003–2013

The documentation team has measured time consistently since 2003 (see Figure 3). The graph starts when the team started to document version 7 of the product, and shows the long transition to the current version 8.

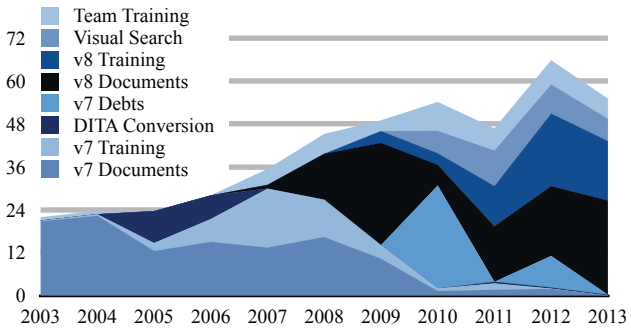

Figure 3. Documentation effort in persons-month.

After documenting version 7, the team converted the documentation to DITA in 2005, developed training materials and, finally, rewrote significant parts of the documentation in 2010 to accommodate illustrations and reflect accumulated knowledge.

Version 8 started by sharing many topics with version 7 in 2009, then benefited from training materials, the search for illustrations, and the maturity of the previous version.

5. CONCLUSIONS

Technical writers in Altitude Software are first and foremost professional learners that must learn specific writing techniques, become experts in the product, and approach the background expertise of specific user audiences. The initial learning required precludes the use of consultants.

Writing tests and interviews with the whole team enabled the company to hire highly technical writers in a country where Universities do not offer technical writing courses.

Over the years, advances in product training enabled writers to learn the product faster, while conceptual innovations enabled writers to learn effective writing techniques faster.

6. ACKNOWLEDGMENTS

The author would like to thank all team members for their support, and the presenters at “Technical Writers @ Lisbon” for their insights and inspiring presentations.

7. REFERENCES

[1] Ackoff, R. L. 1994. *The Democratic Corporation: A Radical Prescription for Recreating Corporate America and Rediscovering Success.* Oxford University Press, New York.

[2] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. 1977. *A Pattern Language*. Oxford University Press, New York.

[3] Armour, P. G. 2004. *The Laws of Software Process: A New Model for the Production and Management of Software.* Auerbach Publications, Florida.

[4] Baptista, J. 2004. Just-in-time training workshops. *STC Management SIG News* 8, 2 (October 2004), 14 - 16.

[5] Baptista, J. 2008. Agile documentation with uScrum. In *Proceedings of the 26th annual ACM international conference on Design of communication* (Lisbon, Portugal, September 22 - 24, 2008). ACM, New York, NY, USA, 275-276. DOI= http://doi.acm.org/10.1145/1456536.1456596.

[6] Baptista, J. 2008. Pragmatic DITA on a budget. *In Proceedings of the 26th annual ACM international conference on Design of communication* (Lisbon, Portugal, September 22 - 24, 2008). ACM, New York, NY, USA, 193-198. DOI= http://doi.acm.org/10.1145/1456536.1456577.

[7] Baptista, J. 2010. The birth of a company-wide wiki. In *Proceedings of the Workshop on Open Source and Design of Communication* (Lisbon, Portugal, November 8, 2010). ACM, New York, NY, USA, 7-10. DOI= http://doi.acm.org/10.1145/1936755.1936758.

[8] Bellamy, L., Carey, M., and Schlotfeldt, J. 2011. *DITA Best Practices: A Roadmap for Writing, Editing, and Architecting in DITA* (1st ed.). IBM Press.

[9] Brown, J. S. and Duguid, P. 2002. *The Social Life of Information.* Harvard Business School Press.

[10] Clark, R. C. 1989. *Developing Technical Training.* Addison-Wesley, Reading, MA.

[11] Collins-Sussman, B., Fitzpatrick, B., and Pilato, C. 2004. *Version Control with Subversion* (1st ed.). O'Reilly Media, Inc.

[12] Colwell, R. P. 2005. *The Pentium Chronicles: The People, Passion, and Politics Behind Intel's Landmark Chips*. Wiley-IEEE Computer Society Pr.

[13] DeRespinis, F., Hayward, P., Jenkins, J., Laird, A., McDonald, L., and Radzinski, E. 2011. *The IBM Style Guide: Conventions for Writers and Editors* (1st ed.). IBM Press.

[14] DITA Open Toolkit Project. DITA Open Toolkit. http://dita-ot.sourceforge.net/

[15] Gerson, S. and Gerson, S. 1992. *Technical Writing: Process and Product.* Prentice-Hall, Inc.

[16] Hackos, J. 1994. *Managing Your Documentation Projects*. John Wiley & Sons, Inc., New York, NY, USA.

[17] Halvorson, K. 2009. *Content Strategy for the Web* (1st ed.). New Riders Publishing, Thousand Oaks, CA, USA.

[18] Hitt, W. D. 1988. *The Leader-Manager: Guidelines for Action.* Battelle Press.

[19] Horn, R. E., Nicol, E., Kleinman, J., and Grace, M. 1969. *Information Mapping for Learning and Reference.* Cambridge, MA: I.R.I (Air Force Systems Command Report ESD-TR-71-165).

[20] Nagle, J. 1996. *Handbook for Preparing Engineering Documents: From Concept to Completion.* IEEE Press.

[21] Peters, T. 1994. *The Tom Peters Seminar: Crazy Times Call for Crazy Organizations.* Vintage.

[22] Priestley, M, and Hackos, J. (eds.). 2005. *Darwin Information Typing Architecture (DITA) Architectural Specification v1.0.* OASIS Open.

[23] Priestley, M. 2001. DITA XML: A reuse by reference architecture for technical documentation. In *Proceedings of the 19th annual international conference on Computer documentation* (Santa Fe, New Mexico, October 21 - 24, 2001), SIGDOC'01, ACM, New York, NY, 152-156. DOI= http://dx.doi.org/10.1145/501516.501547.

[24] Pringle, A. and O'Keefe, S. 2009. *Technical Writing 101: A Real-World Guide to Planning and Writing Technical Content* (3rd ed.). Scriptorium Publishing Services, Inc.

[25] Raggett, D. 2006. Slidy: A web based alternative to Microsoft PowerPoint. In *Proceedings of XTech 2006* (Amsterdam, Netherlands, May 19th, 2006).

[26] Roam, D. 2008. *The Back of the Napkin: Solving Problems and Selling Ideas with Pictures.* Portfolio.

[27] Schwaber, K. 2004. *Agile Project Management with Scrum.* Microsoft Press, Redmond, Washington.

[28] Schwarz, D. 1995. Flexible formatting with linuxdoc-sgml. *Linux Journal* 1, 18 (Oct. 1995).

[29] Stacey, R. 1992. *Managing Chaos.* Kogan Page, London.

[30] Tracey, J. R., Rugh, D. E, and Starkey, W. S. 1999. Sequential thematic organization of publications: how to achieve coherence in proposals and reports. *SIGDOC Asterisk J. Comput. Doc.* 23, 3 (August 1999), 4-68. Reprint of a January 1965 article. DOI= http://doi.acm.org/10.1145/330595.330597.

[31] Young, I. 2008. *Mental Models: Aligning Design Strategy with Human Behavior* (1st ed.). Rosenfeld Media, Brooklyn, New York.