STC

intercom

NOVEMBER/DECEMBER 2014

THE MAGAZINE OF THE SOCIETY FOR TECHNICAL COMMUNICATION

SPONSORED BY Adobe

NEW ADVANCED WORKSHOP AVAILABLE

MADWORLD

The technical communication and content strategy event you don't want to miss.

SAN DIEGO | APRIL 12-14, 2015

FULL SCHEDULE NOW AVAILABLE

"This conference rocks! Can't wait to enhance my projects with all the new tips and trick I've learned."

*—Chad Krieger | **Policy and Procedure Analyst, Umpqua Bank***

FEATURED SPEAKERS

Matthew Ellison
UA, Europe

Derek Warren
Venafi, Inc.

Tony Self
HyperWrite

Angela Richer
Autotask Corporation

Laurie Pulido
eLearning Innovation LLC

Scott DeLoach
ClickStart

Allison Ellington
Travelport

Neil Perlin
Hyper/Word Services

OVER 40 SESSIONS AND MORE THAN 20 EXPERT SPEAKERS FROM AROUND THE GLOBE

Learn from Illumina, Blackbaud, Venafi, Autotask, Sabre, Travelport and more.

VIEW SCHEDULE

VIEW SPEAKERS

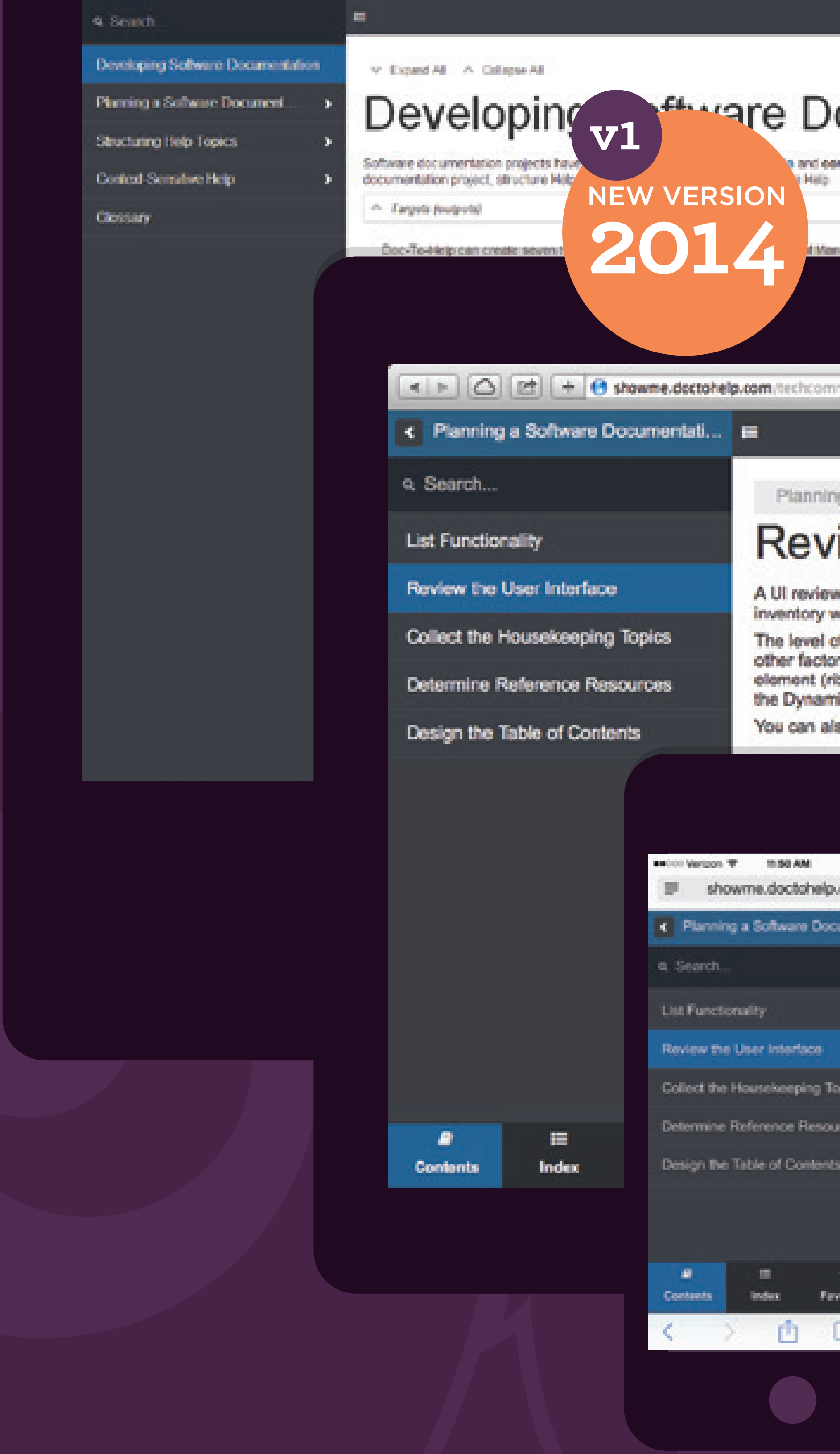
madcap software

One Output for Every Screen

Trusted by technical communicators all over the world to deliver professional-quality documentation, Doc-To-Help publishes technical instructions, policies, procedures, and training manuals for the web.

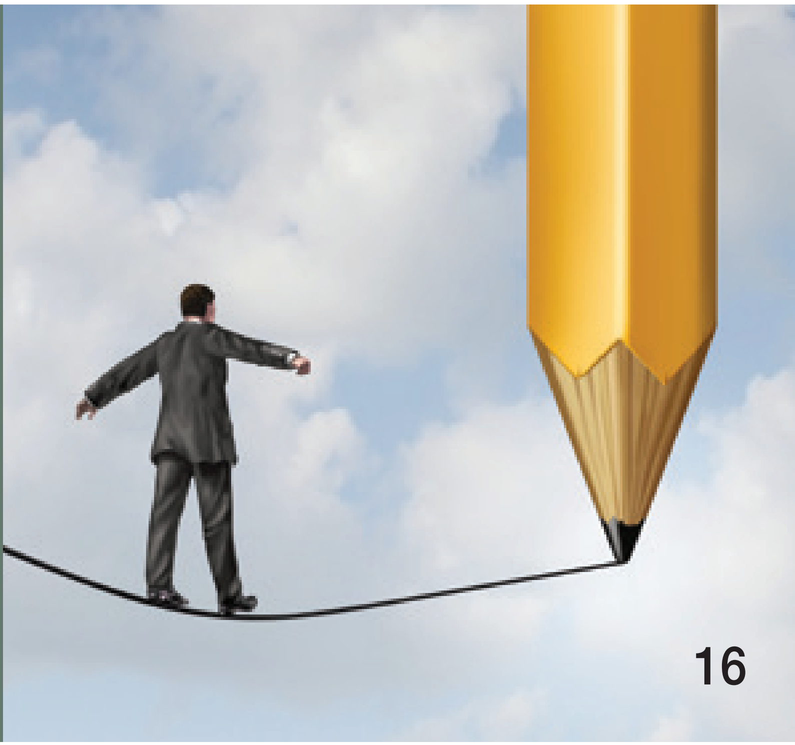
Gone are the days when web publishing only involved the traditional monitor. Now screen sizes range from three to 27+ inches and interactivity has moved from the traditional keyboard and mouse to touch.

Doc-To-Help has the solution for every situation: Responsive Help. This new output responds to screen size and adjusts itself accordingly. With Doc-To-Help's Responsive Help you need only produce one output for the web and it will work on any device.


Search...
Developing Software Documentation
Planning a Software Document...
Structuring Help Topics
Context-Sensitive Help
Glossary
Expand All
Collapse All
Developing
v1
NEW VERSION
2014
showme.doctohelp.com
Planning a Software Documentati...
Search...
List Functionality
Review the User Interface
Collect the Housekeeping Topics
Determine Reference Resources
Design the Table of Contents
Contents
Index

doc·to·help®

Download a Free Trial @
doctohelp.com



Nov/Dec 2014

Volume 61 | Issue 10

Adobe

More than 600,000 users across more than 20,000 companies worldwide rely on Adobe technical communication tools to deliver measurable gains!

MISYS

99%

reduction in output time for PDF and HTML files

ANA

80%

boost in staff efficiency in editing manuals

IBM

20%

reduction in development time for course content

medidata

60%

accelerated localization time

EUROPE AIRPOST

70%

reduction in printing and paper material cost

BIHLER

80%

reduction in turnaround time

Request private demo

Request more information

22

25

32

31

intercom

Intercom, the magazine of the Society for Technical Communication, is published to provide practical examples and applications of technical communication that will promote its readers' professional development.

Publisher Society for Technical Communication
Editor Elizabeth E. (Liz) Pohland
Assistant Editor Marisa Seitz

CONTRIBUTING EDITORS
Scott Abel, *Meet the Change Agents*
Andrea Ames and Alyson Riley, *The Strategic IA*
Thomas Barker, *The Academic Conversation*
Nicky Bleiel, *The Essentials*
Jack Molisani, *Advancing Your Career*
Neil E. Perlin, *Beyond the Bleeding Edge*
Linda Roberts, *All Access*
Derek Ross, *Ethics*
Kirk St.Amant, *Student Perspectives*
Brian Still, *Trends in Usability*

EDITORIAL ADVISORY PANEL
Beth Agnew, Barrie Byron, Liz Herman (Chair), Alan Houser, Paul Mueller

SOCIETY FOR TECHNICAL COMMUNICATION
9401 Lee Highway, Suite 300
Fairfax, VA 22031-1803
+1(703) 522-4114 | +1(703) 522-2075 fax
Send questions and correspondence to the editor at the above address. General information about STC: stc@stc.org *or* www.stc.org.

Address Changes: membership@stc.org
Intercom Editorial: intercom@stc.org
Reprint Permissions: www.copyright.com
Intercom (ISSN 0164-6206) is published 10 times a year (monthly, except for August and December) by the Society for Technical Communication, 9401 Lee Highway, Suite 300, Fairfax, VA 22031-1803, a nonprofit educational organization, as a service to its membership. Membership in STC includes a subscription to **Intercom**. Periodicals postage paid at Fairfax, VA, and additional offices.
Postmaster Send address changes to **Intercom**, Attn: Membership, c/o Society for Technical Communication, 9401 Lee Highway, Suite 300, Fairfax, VA 22031-1803.
Copyright © 2014 by the Society for Technical Communication. All rights reserved. Printed in the USA.

WINNER 2013 APEX® AWARDS FOR PUBLICATION EXCELLENCE

STC

A Note from the Editor

THE LAST ISSUE of *Intercom* for 2014 is a special issue devoted to Agile guest edited by Alyssa Fox, director of information development and program management at NetIQ Corporation in Houston, Texas. Alyssa is a member of Customer Experience Professionals Association (CXPA) and User Experience Professionals Association (UXPA). She is also a senior member of Society for Technical Communication (STC) and is currently serving as the STC Secretary. Alyssa speaks at numerous international conferences about various management, agile, and technical communication topics. You can find Alyssa on Twitter *@afox98*.

I want to thank Alyssa and all the authors who contributed articles to the print and online versions of *Intercom* for the sharing their expertise on Agile. STC has also launched a new Agile Special Interest Group (*www.agilesig.org/*) that may be of interest to those of you who want to keep a conversation going!

Liz Pohland

—Liz Pohland
liz.pohland@stc.org

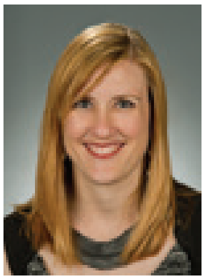
STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

INTERCOM (ISSN 0164-6206) is published ten times a year by the Society for Technical Communication, a nonprofit educational organization, as a service to the membership. Members may pay $60 for a print subscription to INTERCOM.

The mailing address of both the publication and the publisher is 9401 Lee Highway, Suite 300, Fairfax VA 22031-1803. The publisher is Elizabeth Pohland, and the editor is Elizabeth Pohland. The owner of the publication is the Society for Technical Communication.

There were 612 copies of INTERCOM published in September 2014. The average for the preceding 12 months was 498. The paid/requested outside-county mail subscriptions for the September/October 2014 issue were 572; the average for the preceding 12 months was 453. Total distribution for September 2014 was 572. The average for the preceding 12 months was 453. Twenty free copies of INTERCOM were distributed by mail in September 2014 and the average number of free copies distributed during the preceding 12 months was 25. Twenty copies of INTERCOM were not distributed in September 2014, and the average number of copies not distributed during the preceding 12 months was 20. The percent paid/requested circulation in September 2014 was 96 percent; for the preceding 12 months, percent paid/requested circulation was 95 percent.

A Note From the Guest Editor



AGILE IS TAKING the software world by storm. Its benefits—such as faster release times, closer team collaboration, and continuous user feedback—are hard to beat. Every time I talk with technical communicators at conferences, I have found that more and more of them work at organizations that are starting to adopt the Agile methodology. Yet questions abound. How does working in an Agile environment affect my work? How does working in an Agile environment help me? What will I need to know to get the most benefit from Agile?

I've proclaimed for years that information developers must be equal members on their project teams to really provide the most value to those teams and in their organizations. Historically, technical writers have been met with such challenges as lacking access to team members to get information to do our jobs, exclusion of our work efforts from the official project schedule, and demonstrating what all we really can do as content developers.

Agile is one of the best ways to enable writers to meet these challenges and excel. The emphasis on frequent and clear communication in an Agile environment showcases our skills as professional communicators. The planning meetings involved in Agile highlight our work as an important and equally relevant contribution to the project schedule. And the inclusion of writers' work in the definition of "done" for a user story helps solidify teamwork and increase awareness of what we are doing.

Additionally, working in an Agile environment increases the quality of the content we produce. Since content is produced each sprint alongside coding and testing work, the review and editing takes place within that same sprint. Team members are reviewing smaller parts of documentation during each sprint, so we get more thorough reviews, leading to more technically accurate and higher-quality content.

In my organization, including writers as equal members of Agile project teams has changed our working relationships with our fellow team members for the better. We are involved from the beginning of the project life cycle, we participate in and contribute to design discussions, we include estimates for our work in release and sprint planning meetings, and we help groom the backlog for what goes into the product in the future. Each of these avenues gives us a chance to show what we can do, how we can raise the bar for the team, and how we can make the product better.

This issue of *Intercom* contains articles that describe different technical communicators' perspectives of working in an Agile environment. From showing how to develop content in this environment to how our skills can help us take on different roles, the issue explains several possibilities that this new and exciting world gives us. Writers give us tips on learning how to plan in this arena, new ways to think of our deliverables in an Agile world, and ideas for how to build and solidify Agile teams.

Agile helps technical communicators contribute in new ways to products, and reinforces the importance of our current work throughout the release cycle. Showing our value at these various touch points consistently reminds people of who we are and what we can do. We don't just write documentation. We help teams run more smoothly. We serve as user advocates. We help design products. We solve business problems. And most importantly, Agile helps us make our customers happy.

—Alyssa Fox
alyssa.fox@netiq.com

Turning Obstacles into Opportunities:

shutterstock.com/Ellagrin

Agile for Technical Communication

By KAREN SMITH and PATTY GALE

YOUR ORGANIZATION'S DEVELOPMENT teams are adopting the Agile framework. Should your learning content team follow suit? Or continue to work using established processes to produce quality documentation?

If you are part of a learning content team thinking about moving to Agile, here are some pros and cons to consider. The pros describe the ways in which an Agile environment can streamline many day-to-day tasks that you face as a writer. The cons present obstacles and challenges that may seem daunting, along with proposed solutions that can help you work through these issues.

By working together in a supportive Agile environment, you and your team members will discover the strategies you need to be successful as Agile technical writers. You'll soon realize that, as 17th century English churchman Robert South declared, "An obstacle is often an unrecognized opportunity."

People Power

A typical Agile team is composed of five to nine people with cross-functional skills. The team works together to design, develop, test, and document products or features in small increments. When the Agile environment is adopted across an organization, many learning content teams fear that their staff won't stretch far enough to support all of the Agile teams. How can technical writers fit into this framework?

Pro: Increased Collaboration

Agile teams operate under the assumption that "we're all in this together." The team succeeds together or fails together. As a result, roles often blur: everyone on the team can help design, test, and document features.

For writers, this increased collaboration leads to greater insight into the development process and greater access to developers and subject matter experts. Ken Kidder, manager of information development for appliance solutions at Symantec Corporation, said, "I feel Agile requires more communication, and folks are beginning to understand that within the Scrum teams. We experience more communication between teams than ever before."

Con: Not Enough Writers for Agile Teams

Ideally, one writer is embedded in each Agile team. The writer performs as a full member of the team and attends daily stand-ups and team meetings. When you have more Agile teams than writers, perhaps each writer can support two teams. If writers need to support more than two teams, however, they may spend so much time performing Agile overhead tasks that they have little time for writing.

The Solution: If the writer-to-team ratio doesn't work, consider organizing the learning content team as its own Agile team that responds to requests from product teams for new feature documentation. For more information on this strategy, see our presentation on Slideshare, *Parkour: Lessons in Agility*.

Teamwork

Rather than a process, Agile is more accurately described as a framework or a way for teams to interact and get work done. Agile teams are self-organizing, value face-to-face conversations, and practice an all-hands-on-deck attitude. While the emphasis is on individuals and interactions, the tools and processes that are implemented as part of the Agile framework have value, too.

Pro: Greater Transparency Into Upcoming Work

In an Agile environment, you gain more insight into the work of the teams you support. Each team's development work is outlined and prioritized in their backlog. Tools such as JIRA (*www.atlassian.com*) or Rally (*www.rallydev.com*) help teams plan and track the work and provide visibility for others in the organization. Christine Brouillard, senior technical writer at CNC Software, Inc., explained, "We now get a lot of the information we need from backlogs and in face-to-face interactions during meetings. Agile has reduced the amount of time that we sit in developers' cubicles talking to them individually, which saves time for them *and* time for us."

With increased visibility into upcoming work, you can evaluate the pace at which features are being developed, and plan your work accordingly. The accidental discovery of last-minute features that require documentation occurs less frequently than before.

Brouillard added, "In the past, we would stumble across software changes by chance. The communication mechanism to get the information to us just wasn't reliable. With Agile, even if you don't have all the details about what the changes will be, at least you know what areas are changing, so you can plan for what's coming."

Con: More Meetings

Agile meetings are work sessions where team members plan and prioritize the work that needs to get done (sprint planning), provide status (daily stand-ups), and demonstrate work completed (sprint demos). They provide the framework or structure around which the development work is completed.

When you first transition to Agile, the meeting load can feel burdensome. During the course of a two-week sprint, an Agile team can spend eight hours or more in meetings. You may wonder when you'll have time to develop learning content or perform other work.

A common mistake is to add the new Agile meetings to the existing meeting schedule. Agile meetings are designed to be sufficient for the team to complete their work. Everyone in the organization has well-defined touch points to interact with the Agile team. Ad-hoc meetings can address immediate needs within a sprint.

The Solution: All Agile meetings are time-boxed and have a recommended duration. The Agile team must learn to adhere to the recommended frequency and duration of the meetings. This adjustment can take practice and experimentation. However, under the guidance of the Scrum master (the team's servant leader), the team can improve their Agile practices and interactions, including how often and when the team meets. Casie Oxford, senior content manager at Zix Corporation, said, "When we first started Agile, the meetings were too long. But now that we know what we're doing, we're much more efficient."

Workloads

In a waterfall environment, the writing workload is often cyclical. Early in the release cycle, you may have a light workload. As development work for new features is completed, however, you often encounter "crunch time." You suddenly have more work than you can manage, and you work long hours to get it all done in time for the product release. How does this compare with workloads in an Agile environment?

Pro: A More Consistent, Sustainable Level of Work

One of Agile's biggest benefits to writers is the even distribution of work throughout the year. When collaborating with Agile teams, you document features as they are developed. The product and its documentation are tested along the way, leading to fewer errors or last-minute surprises before a release. And because the documentation reflects the current state of the product, the learning content team can support more frequent releases.

For Casie Oxford, release-driven crunch times are a thing of the past. "We have been doing Agile for three or four years, and there are not as many highs and lows for us. Our workloads are pretty level over time, which means we don't feel nearly as crunched at the end of a release as we used to."

Con: Be Prepared to Reverse Your Work

When you document features as they are being developed, there is a chance that stakeholders will decide a feature is not yet ready for release. Not only must the feature be removed from the product, but its documentation must also be removed.

The Solution*:* When you adopt Agile, consider how you will address such issues. At the start of a release cycle, make a release management plan for the documentation. Use a content management system with version control. Develop a way to track the documentation that is added and updated for each feature. If a feature is omitted from a release, you can ensure that its documentation is removed.

Iterative Development

In a waterfall development environment, you may wait until the development teams are done designing, coding, and testing the product before you begin developing the documentation. Because you're documenting the finished product, the content is not likely to change.

In contrast, when you work in an iterative development environment like Agile, smaller working pieces of the final product are delivered at the end of each sprint. How does this difference change your work?

Pro: Work in Smaller Chunks and Iterate

In an Agile environment, you're drafting documentation during the sprint in which the feature is developed, or perhaps a sprint behind. As a technical writer, it may feel uncomfortable to not have all the information to document the product. This ambiguity can be unsettling.

On the other hand, knowing that the product is likely to evolve in future iterations means you'll have an opportunity to further refine and improve the quality of the documentation before it's delivered to users.

Con: Projects Are Too Large to Fit Into Sprints

If you're not Agile yet, you're probably used to working through a large, complex project until it's done. You may spend weeks or months on the same project. Keeping up the momentum (and the motivation) can be challenging.

When you transition to Agile, you'll quickly learn that the work required to complete a large project doesn't fit into a sprint. In addition, external dependencies require interaction with people outside your team, and those interactions may extend beyond sprint boundaries. These realities can make it challenging to complete a project within a sprint.

The Solution: Learn how to chunk a large complex project into smaller pieces. Work with team members to break the project into chunks that fit into a sprint, and share the load with others on your team to get it done. Identify dependencies and plan them in your sprint scheduling. Flexibility and adaptability are the keys to making this work.

User Experience

Organizations now recognize that technical communication is an important part of the user experience (UX). Agile teams are learning to incorporate UX design with lightweight prototyping and rapid iteration to evolve designs within a sprint. Some teams invite a panel of customers to sprint demos, or invite their feedback on a prototype during a sprint to make sure the team is providing a positive user experience. How does this emphasis on UX affect documentation in an Agile environment?

Pro: Higher Quality, More Usable Documentation

Being involved with the team during active feature development means you have a deeper understanding of the feature and its use cases. UX designers can provide real-world examples for conceptual topics. Quality assurance (QA) engineers can offer sample data sets for screenshots. Team members review and test the new or updated documentation during the sprint. Having just designed, coded, and tested the product, technical reviewers have immediate knowledge of the features, so their feedback is more accurate and detailed. Ken Kidder explains, "The writers on my team like the fact that reviews are occurring within each sprint, which leads to a technically accurate and successful demo at the end of each sprint."

By participating in UX activities (taking notes during user interviews, observing usability tests, and responding to user questions during a beta release), you can gain insights into users' pain points and be sure to address them in the documentation. When you show documentation to customer panels during sprint demos, you receive early feedback and can iterate the content to better meet user needs. The end result? Documentation that is more accurate and more usable.

Con: Writers Have Little/No Experience with UX

If you have never performed tasks such as interviewing users and conducting usability tests, you may not feel confident participating in these UX activities.

The Solution: Ask your user experience team to hold workshops that introduce writers and other Agile team members to UX methodologies. When UX activities are performed during a sprint, ask if you can participate in some way, such as taking notes or observing sessions. Look at your expansion into the UX realm as another tool in your technical writing toolbox, contributing to your professional development and making you a more valuable member of your Agile team.

Obstacle or Opportunity?

When it comes right down to it, your learning content team may not have much choice: if the organization is adopting the Agile framework, you may be required to follow suit or at least adapt to support it.

If that's the case, look at Agile as an opportunity, not an obstacle. For example, you may find that working in an Agile environment can increase awareness and credibility for the learning content team.

Sprint demos provide an opportunity to share your work, making it more visible to the rest of the organization. For example, before our organization adopted the Agile framework, most of our co-workers thought we only documented new features. Now they understand that we also enhance the quality of existing learning content, improve searchability, and reach out to users by replying to their questions and comments. Our sprint demos have made the organization more aware of these efforts.

In an Agile environment, quality is a team sport. When you participate in the Agile framework with developers, designers, QA, and product management, you work together to mitigate risks, overcome obstacles, and celebrate successes in delivering a better product to your customers. There's nothing like being a team player to build credibility within the organization. Your visibility increases, and you build a greater awareness of the business value your team brings to customers. i

PATTY GALE *is a principal learning content developer for a technical writing team at Autodesk. She also serves as the team's information architect and content strategist. A technical communicator for over 25 years, Patty has worked at businesses all sizes, from small start-ups to large corporations. She has received multiple awards from her employers and STC competitions, including an International Award of Distinguished Technical Communication. She presented about Agile at the STC Summit 2014. Patty holds Bachelor's degrees in Computer Science and Business Management.*

KAREN SMITH *is a senior learning content developer at Autodesk, where she has worked with Agile teams since 2011. Currently, Karen serves as the Scrum master for a horizontal service-based Scrum team that develops learning content for over 20 Scrum teams around the globe. She presented about Agile at the STC Summit 2014. Before joining Autodesk, Karen was an instructional designer, e-learning developer, and online instructor for IBM. During her career, she has worked as a curriculum developer and technical trainer for a variety of for-profit and nonprofit organizations. Karen has a degree in public communication.*

REFERENCES

Brouillard, Christine. Senior Technical Writer. CNC Software, Inc. Personal interview on 26 August 2014.

Gale, Patty, and Karen Smith. 2014. *Parkour: Lessons in Agility* [slides]. Available from *http://slidesha.re/1pmHJvI*.

Kidder, Ken. Manager of Information Development for Appliance Solutions, Symantec Corporation. Personal interview on 26 August 2014.

Oxford, Casie. Senior Content Manager, Zix Corporation. Personal interview on 21 August 2014.

South, Robert. *http://en.wikisource.org/wiki/South,_Robert_(DNB00).*

Agile technical documents are frequently updated and enhanced. They are no longer monolithic piles of information, but rather evolving, ever-improving, integral parts of your product.

Tech Docs as Agile Deliverables

By JOHN COLLINS

"THE BIG SOFTWARE release is only six weeks away, and we still have 12 weeks of work before we can ship the online help system," your colleague says. "We've got to talk to the project manager."

This scenario may be all too familiar, but does it need to be? Do you need to talk to the project manager? The answer to both might be no.

If you think about it, the software is probably bigger and more complex than your help system. So why would you need six more weeks to finish the help system than the developers need to finish the software? "That's easy," you say. "The team has far more developers than technical writers, and some of the features were just finished in the software—or still aren't built yet."

Those things may be true, but you can probably deliver *something* in six weeks, to match the software release. Simply put, you can apply the same Agile principles to your documentation as your software development team applies to its work.

Now is a good time for disclaimers.

- You may be in a regulated industry with specific requirements on technical documentation. You may not have as much leeway.
- This article is written with digital deliverables in mind. Physical deliverables become more complicated.
- As with just about any article or blog post related to technical communication, your mileage might vary.

What Is Agile?

Agile software development can take many forms, from the common Scrum to Extreme Programming to Kanban and a number of other methods. At its core, Agile is iterative and flexible, changing when needed and geared toward action.

Technical writers may be most familiar with Scrum, so let's focus on that. In Scrum methodology, there are three roles on the team:

1. Product Owner: Drives the vision and requirements and makes the final decision when big questions arise.
2. Scrum Master: Facilitates, clears blockers, protects team members' time, and helps to implement the team's agreed-upon process.
3. Team Member: Works in some capacity on the project. Includes developers, quality assurance analysts, technical writers, and so on.

Often there's a backlog of features to be built and bugs to be fixed. Everyone expects additional features and refinements to come in future releases.

The goal is to finish each sprint with working software. To some teams, that means releasing code to customers. To other teams, it means that a new feature is finished and none of the code is broken at the end of the sprint, even if a complete release isn't quite ready to ship.

The beauty of Agile is that fully functioning software is released regularly, yet additional features are added over time as scope allows and user feedback comes in. When done properly, users get more usable software from an Agile process than something that's defined, developed, and released all in one isolated chunk of time.

Oh, and that pesky line in the Agile Manifesto about valuing working software over comprehensive documentation? Don't get defensive about "comprehensive documentation." Instead, understand that "comprehensive documentation" in the context of working software has more to do with design documentation like specifications and requirements than the user documentation you write. But also realize that this frees you up to deliver useful documentation instead of comprehensive documentation.

What Would Agile Tech Docs Look Like?

Like Agile software, Agile technical documents are frequently updated and enhanced. They are no longer monolithic piles of information, but rather evolving, ever-improving, integral parts of your product.

In fact, try thinking of your documentation deliverables as if they are software themselves and approach your work as if you were a software developer. If you're working in single-sourced XML or HTML-based content, it basically *is* software.

Define the absolute minimum that your deliverables need, both in terms of content and form (as online help or as PDF). Then prioritize for the backlog what improvements should follow. The backlogged priorities can then work their way into future sprints.

Let's look at some examples of Agile documentation in two sprints.

Sprint 0	Sprint 1
Write instructions for quick-start guide and release a plain, unbranded PDF	Add branding to output and release a completely branded PDF quick-start guide
Outline the structure for new document	Complete the first draft of the new document
Release fully-finished quick-start guide	Release a basic user guide
Release the English help system	Release the localized help system in top two user languages
Add Feature X topic to user guide and help system	Add Feature Y topic to user guide and help system
Migrate Word documents to single-sourcing tool	Refine single-sourced content and output types for initial draft outputs
Release help system	Release PDF user guide
Add indexing and keywords to help system	Add relationship tables to help system

Why Should I Do This?

My background is in book publishing and print journalism. While we had regular production cycles at the newspaper where I worked, it was still more of a linear "waterfall" process. The move to Agile when I changed careers involved quite a learning curve.

If you struggle with thinking in a waterfall way, you may find yourself at odds with everything else that's going on in your organization. Do you struggle with paralyzing perfectionism? I did. Moving your technical documentation approach to be more like the Agile methodology will bring you in line with your project management and developer colleagues who are all Agile all the time. You'll find yourself able to embrace sprints as a chance to fix mistakes and constantly improve instead of waiting until your work is perfect and ready for the masses.

But more importantly, working with Agile technical documentation lets you manage scope, know what you're trying to accomplish, and focus on priorities. You gain the mechanism to concentrate on what your users really need. For example, instead of shipping no document or shipping a document that's a catalog of every interface element, you can ship a basic task-based document. Perfect form can come later.

Feeling unappreciated as a technical writer? Maybe stuck in a rut describing every screen in your product and producing thousands of words that you doubt anyone reads? Move your documentation to an Agile approach to focus on higher priorities and produce high-value deliverables. As a result, you'll gain relevance and prove your value to your organization.

How Do I Do This?

Here are some tips about how to make your tech docs more Agile.

Find a Product Owner

To adopt an Agile documentation approach, you'll need to be a champion for change. But you'll also need someone who can help you pull it off. Find someone to be the product owner for the technical documents. Hopefully that's the actual product owner of the software team you're on. If that's not possible, find someone else like a project manager or the technical writing manager who can drive the requirements for the Agile docs and build and manage a backlog of documentation features and bugs.

If the technical documents product owner is not also the software product owner, the people in those two roles need to make sure that they agree on the technical documentation approach.

Get Control of Your Releases

As long as you're dependent on others to distribute your digital deliverables, you're going to lose the beauty of iterative improvements to your work. If another team uploads your PDFs to the knowledge base or if product rollouts are required to update PDFs or help systems, then you're at the mercy of those teams.

If needed, work with your developers—and system administrators—to get the infrastructure and access you need to put your technical document deliverables directly on the Internet.

If you control your releases, you'll want to have some process set up around release management. Again, model what you do after what software developers do.

What might be part of your release management process?

- Rollout Plan: What steps need to happen to put your deliverables online?
- Version or Release Numbering: I like my deliverables to show a version number. It's an easy way to know what you're looking at. Otherwise, you may have to search for that one sentence you rewrote or the screenshot that you updated to know if you're looking at the current version. Maybe you use the versioning number of the software, but maybe you want something more specific to your deliverable.
- List of Current Releases: If you manage multiple deliverables, keep a list that shows each deliverable, the release number, the date released, and some notes about the deliverable, if desired.
- List of Past Releases: Keep a running page with information about all releases. Include links to copies or source control tags of the deliverable; links to epics, stories, or tickets related to the release; and a summary of the release and changes since the last release.

As an Agile participant, you want to be transparent, so you want to publish these items somewhere accessible within your company, such as a Wiki or Google Docs.

Be Agile

You're not just adjusting your deliverables. You're changing how you think and how you act.

So be Agile. Know what you're shipping and what you're not (yet). Iterate. Seek and use feedback for your iterations. Say no in a positive way. ("If we do that, then we'll have to make concessions in XYZ.")

Think like a user and produce what they need. Think like a product owner and focus on requirements while planning for nice-to-haves. Think like a developer and plan your work to be shippable on a sprint cadence.

Are there things that you can automate in your tech doc process? Could you use scripts to check parts of the content? Can you set up a system that automatically builds updated outputs for internal stakeholders at some interval (each commit, nightly, weekly, etc.)?

What Should I Watch Out For?

Let's be clear: no methodology is perfect. Of course there are drawbacks.

As mentioned earlier, Agile will be harder to implement if you're producing physical deliverables, such as printed documents. You won't be able to iterate as often, but you could maybe do smaller print runs to give yourself a chance at more iterations. (That may not be realistic, but hopefully you get the idea.) You can still probably chunk out the production work into sprint-sized tasks.

If you're in regulated industries, you may face a lot of limitations on the ideas presented here for Agile documentation. Obviously, you've got a clear list of requirements that a product owner can rely on, and that gives you a starting point (that you've always been working toward).

Can you break down those requirements into stories that can fit into individual sprints? Maybe this will help you manage the workload or maybe it will be a formality that hinders you from getting stuff done.

In Agile, a backlog is a useful tool to help you track and prioritize, but it can also become an unattainable wish list that weighs on the conscience of a thorough technical writer. This can happen when you have a plan of improvements to make to your docs and organizational priorities make it so you never get to the planned improvements. It can also happen when your backlog grows faster than you can knock out items on it.

A practical word about the backlog: If you're an Agile technical documentation team, then it's fairly easy to set up your own backlog. If you're embedded with design, development, and QA, maybe you can have your backlog combined with the entire team's backlog. If not, then things might get a bit more complicated, but think about using tags or labels in your team's ticketing system as a way to build out your backlog.

The good news is that a well-documented backlog may make it easier for you or your manager to make the business case for a new hire. And if the backlog is in really good shape, that new hire will know exactly what needs to be done when they start.

Maybe next time, you'll hear your Agile colleague say, "The big software release is only six weeks away, and we'll deliver technical documents then, too. No problem." i

JOHN COLLINS *was a senior technical writer for five years at Rosetta Stone in Harrisonburg, VA. Hired as the second full-time technical writer in the company, he was team leader and Scrum master of the eight-person technical communication and localization team. Now senior UX content strategist at Rosetta Stone, John's role sits at the intersection of user experience, content strategy, technical communication, and localization. He blogs at* www.intersectUX.com/blog *and you can find him on Twitter* @jrc_collins.

FURTHER READING

Ariel, Mikey. "My Personal Tech-Writing Agile Manifesto." *https://voicerepublic.com/venues/225/talks/866* (audio) and *www.slideshare.net/Develcz/mikey-arieldevel-cz14mikeytalknoclick* (slides).

Beck, Kent, et al. "Manifesto for Agile Software Development." Accessed 31 August 2014. *http://agilemanifesto.org/.*

Berry, Tana, and Ann Gentle. "Writing End-User Documentation in an Agile Development Environment." *Just Write Click* (blog). Accessed 31 August 2014. *http://justwriteclick.com/2007/07/02/writing-end-user-documentation-in-an-agile-development-environment/.*

Maddox, Sarah. "The Agile Technical Writer." *FFeathers* (blog). Accessed 31 August 2014. *http://ffeathers.wordpress.com/2008/01/20/the-agile-technical-writer/.*

Mazet, Jean-Luc. "Agile Technical Documentation." *WritersUA.* Accessed 31 August 2014. *http://writersua.com/articles/Agile_doc/.*

Agile, an Awesome Alternative

shutterstock.com/grmarc

By GAVIN AUSTIN

WHEN WAS THE LAST TIME you heard a bunch of technical writers say, “Agile is the best thing in the world”? Probably never. Since I began speaking about Agile in 2008, I’ve rarely heard writers include the words “Agile” and “awesome” in the same sentence. However, I believe that they should. No matter how painful and incorrect an Agile implementation is at a company, an Agile environment offers some awesome alternatives to the typical writer role.

Writers can use Agile to expand their role in ways that were unavailable to them in the old-fashioned waterfall approach to software development. You remember the waterfall model, right? You wait until something is developed in a test environment, and then you hunker down and write documentation according to whatever is testable. In that model, the writer’s role is literally summed up in one sentence, and who wants a one-sentence role? Not only is such a role limiting, routine, and dull, but unless you experiment, take risks, and find alternative ways to add value, you won’t grow personally or professionally. You can use Agile’s emphasis on iterative development and continuous improvement to expand your career.

Unleashed by Agile

Agile’s focus on daily face-to-face communication and close collaboration with development team members should have writers jumping up and down with joy. Those two Agile principles alone lay the foundation for writers to pursue new opportunities and develop skills that were previously out of reach. Granted, not every company extends these principles to include writers, but that’s because not every company is practicing Agile as it was intended. Since the writers I work with are considered equal members of every development team, they’ve had the chance to speak up and volunteer to work on tasks far beyond writing documentation. Here are some of the opportunities our writers have chosen to pursue that you can also try out:

- **Serve as Scrum master**. If your company is practicing the Scrum methodology of Agile, you can take Scrum master certification courses to lead critical meetings and remove blockers from teams to facilitate smoother product releases. Using a bit of project management and a bit of people management, you can show that you can help manage development teams.
- **Position products**. You can work closely with product managers and marketing executives to ensure that videos, online help, user interface text, and social media messaging is consistent, whether created by marketing or the documentation and user assistance team. This type of cross-department collaboration can lead to movement to other areas in the company, such as developer marketing.
- **Create campaigns**. You can work with marketing to design HTML-formatted email campaigns that encourage targeted customers to try out new applications. Our marketing teams were so impressed with our writers’ content and involvement that they helped the writers create similar campaigns to distribute “Getting Started” guides to customers and potential customers.
- **Design API names**. While understanding and documenting programmatic features, you can influence developers to reevaluate API names or code samples. Our user advocacy skills help us demonstrate more intuitive ways to name APIs based on product behavior and customers’ expectations, and this enhances code and API business cases as well.
- **Change the user experience**. When speaking with designers or subject matter experts, voice concerns

about clunky or confusing user interfaces. Offering a new perspective on how a customer might experience an application, or volunteering to test a UI as a de facto customer, can give you the ability to change how customers interact with a product used by millions of people.

- **Test for quality**. To help development teams meet tight deadlines, volunteer to test applications when quality assurance engineers are swamped. Not only can this create camaraderie with teams, but you can also find major unidentified quality issues based on your unique perspective as user advocate, rather than from what can be a lofty and limiting point of view of an engineer.
- **Bridge communication gaps**. Writers on multiple teams often hear conflicting or inconsistent communications on how applications will function or be rolled out to customers. By bringing these inconsistencies to the team's attention, you can use your professional communication skills to facilitate discussions and agreements between teams who were unknowingly building duplicate or contradictory features.
- **Evangelize teams and products**. With deep knowledge of a particular product or process, you can showcase your team and company by writing articles or blog posts for recruiting purposes. You can also speak at conferences outside of the technical communication realm, such as at Dreamforce, Agile 2013, Girl Geek Dinners, and the Grace Hopper Celebration of Women in Computing, representing your company and its approach to Agile implementation.
- **Celebrate good times**. After a sprint or product release cycle has finished, consider taking on the role of coach and morale booster by organizing and hosting team parties, lunches, or excursions to cooking classes or bowling alleys. What might seem like a tiny gesture can have a big impact, showing you as a leader and contributing to company culture.

Focus on the Positive, Forget the Negative

Still not convinced that Agile can offer you some awesome alternatives to your current role? Believe me, I understand. Based on my experience with my team's transition to Agile in 2007, and from what I've heard in the industry, a lot of writers feel too overwhelmed by meetings, improper planning, or incoherent user stories to want to believe that Agile is awesome. But over time, Agile can get better. It takes executive support and voicing your concerns at retrospective meetings—a key component of any successful Agile implementation. As Winston Churchill once said, "If you're going through hell, keep going." The *Salesforce.com* Documentation and User Assistance team kept going, and our lives got infinitely better, especially after we came to the realization that we didn't have to be just writers. As customer advocates, Agile lets us pursue interests outside of documentation. Agile allowed us to do more. It helped us grow. By focusing on this positive outcome, the negative side effects of a few more meetings and occasionally unclear user stories didn't seem so bad.

Unleash Yourself

This article isn't meant to sound like a self-help book written by a motivational speaker, but the key to our team's accomplishments is that the writers chose to speak up. You can choose to speak up, too. Use the collaboration principle of Agile to your advantage. By voicing your opinions and volunteering to take on tasks beyond your traditional role, you can gain new career experiences and build up your resume and portfolio in ways that aren't available to writers outside of an Agile environment.

When you attend daily standup meetings, try to do the following:

- **Listen**. What do your teams and customers need help with? Can you use your unique skills as a writer, communicator, and knowledge expert to help?
- **Ask**. If you don't know what your team needs help with, ask. Often there are tasks not talked about because they're not top of mind, but that doesn't mean they don't need to get done. You might be the perfect person to do them. Doing them might make you a hero on the team.
- **Volunteer**. Your team might think of you as just a writer, but once you start volunteering for tasks other than writing, you'll open a door to all kinds of new possibilities. People could start recommending you for projects or positions you never thought would come your way.
- **Take risks**. Maybe you won't be able to complete some tasks perfectly, but are you willing to take risks and find out? Getting out of your comfort zone leads to growth and new opportunities.

Finding Alternatives = Awesome

The bottom line is that if you're looking to grow personally and professionally and explore tasks, opportunities, and skill sets outside of a traditional writer role, then Agile offers you plenty of opportunities to do that. The kicker is that you have to find those alternatives. Nobody's going to hand them to you. By listening to what your team and company needs help with, and pitching ideas or volunteering to take on new projects or responsibilities, you can create the role you want instead of working the role someone else assigns to you. If you use Agile to your benefit, you'll find some awesome alternatives to the routine workday and career. As one colleague told me while we were chatting about Agile, "Working on so many different types of stuff helps keep my job interesting." i

GAVIN AUSTIN *is a lead technical writer at* Salesforce.com, *where he writes everything from UI text to API developer guides. He has delivered presentations on Agile at Agile2013, WritersUA, STC Summits, STC Web Seminars, Blue Shield of California, and San Francisco State University. Additionally, he coauthored* A Writer's Guide to Surviving Agile Software Development, *which was featured on the Scrum Alliance website.*

Adapting to Change:

By LEE TURNER

shutterstock.com/Lightspring

Why Make Plans in an Agile World?

WHY PLAN IN AN AGILE ENVIRONMENT? Agile development is an ever-changing, turbulent environment that may cause you to wonder if you even have time to plan. But planning is the key element of Agile—its heart and soul, if you will. And what cannot be missed but is often overlooked is that the plan or planning phase is not the goal. Results or outcomes are the goal. When the plan becomes the destination, Agile ceases to be, well, agile, and the software environment merely replicates older models of development.

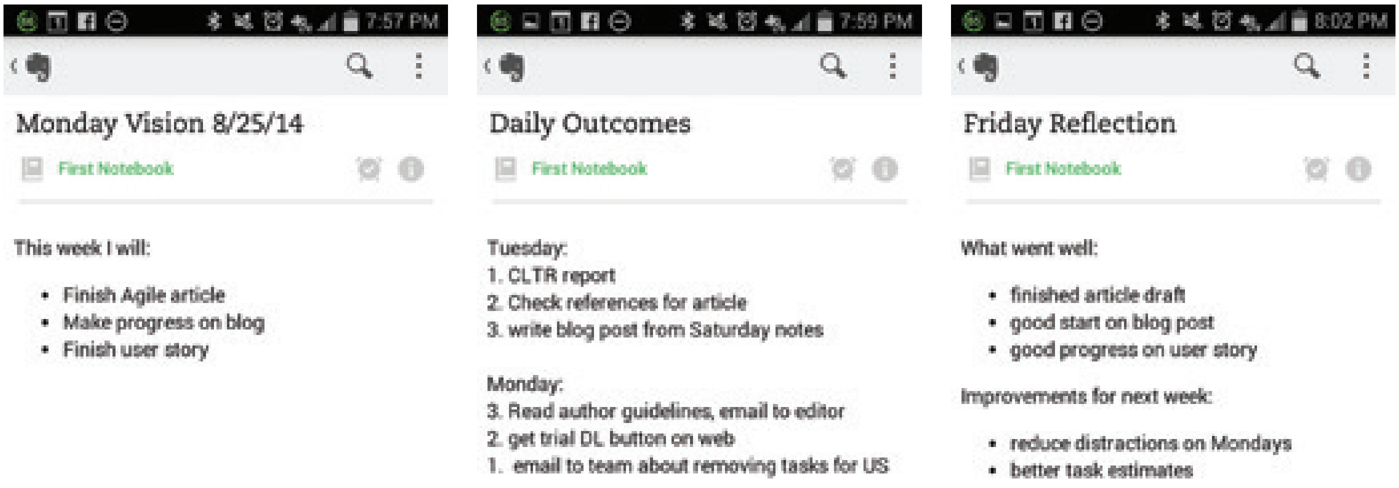
Agile plans are primarily foundations. As with a home, while the foundation is important, no one wants to live in a house with only a foundation. Proper Agile planning takes this philosophy into consideration, and such planning within an Agile development environment remains key exactly because it gives the organization that foundation with which to be Agile. So you may need to change the way you have thought about and approached planning in the past to create effective plans in an Agile environment.

Weekly Planning

The most effective methodology for Agile planning that I have used comes from J. D. Meier's system of Monday Vision, Daily Outcomes, and Friday Reflection (Figure 1). To use this method, start by selecting three results you want to achieve for the week: *the Monday Vision*. Then sift through the tasks associated with accomplishing these results, and select three smaller results to achieve each day: *your Daily Outcomes*. The Daily Outcomes should all support your three big goals for the week. As a key ingredient to this planning phase, make every effort not to merely list all your tasks. These lists can be unending and often do not work cohesively together to accomplish larger goals. At the end of your week, review what you have accomplished toward your goals: *Friday Reflection*. By focusing on goals or results, you will create a real metric that will allow you to recognize and measure how your daily outcomes have helped you achieve larger results. This iterative prioritizing of tasks and results blends well with Agile environments.

As you go through the week, whenever you are ready to start on a new task, review your daily and weekly results goals to make sure you are working on a task that supports your goals for the week. Adjust tasks or daily outcomes only when necessary to achieve the goal or if some higher priority goal surfaces after you have decided on your goals for the week. Start each day by deciding the three outcomes you'd like to accomplish that day. If you accomplish all three by lunch time, start on your next best result to accomplish for the day. By having three daily and weekly outcomes defined, you can keep your focus on accomplishing results.

When the end of the week hits, spend some time reviewing the goals you set and how well you accomplished them. Try to list three things you did well during the week and three things you would like to improve on. As the weeks go by, you should be able to improve your goal setting and your ability to reach your goals. Even when unexpected items find their way into your week, you should be able to quickly react and get back on track. By reviewing what went well and what you can improve each week, you



Monday Vision 8/25/14

This week I will:

- Finish Agile article
- Make progress on blog
- Finish user story

Daily Outcomes

Tuesday:
1. CLTR report
2. Check references for article
3. write blog post from Saturday notes

Monday:
3. Read author guidelines, email to editor
2. get trial DL button on web
1. email to team about removing tasks for US

Friday Reflection

What went well:

- finished article draft
- good start on blog post
- good progress on user story

Improvements for next week:

- reduce distractions on Mondays
- better task estimates

Figure 1.

should have a few items in each category that bubble to the top of the list that you can review with your Agile team during sprint retrospectives.

User Story Development

User story development is one early step in the Agile development process that gets you started with planning. Planning through user stories helps define use-cases and gives you an opportunity to define task-based content related to that user story. A well-written user story should also help you visualize the best medium for conveying your content. As you identify tasks for your user story, you gain a good reference point for identifying your daily outcomes.

Early Writing

Start writing content early in the development cycle to help with planning. As Simon St. Laurent points out, while some folks assume you must be a master of something to write about it, your best content may come from your own early experience with what you are writing about. Who better to speak to your new customer than someone new to the product or feature? Even existing customers will be "new" to the features you add to your product after the initial release. Keep your new customer in mind as you write early in the software development cycle so you don't forget the things you learned as you discovered how a feature works. As you learn more, iterate and improve your content while you keep important details that might trip up the newest users and ensure they remain in your content.

Another method for planning and getting started with writing is to generate content that is "good enough" for now and version your results. This approach is the perfect application to writing in an Agile environment. How many times have you "finalized" your documentation plan months before a product release? Even as you execute the exact plan you made, you realize by the end of the release cycle that you would have made a different plan if only you had known then what you know now. For example, I have documented a feature during the sprint where it was implemented by development, only to realize later that the feature really did not need documentation above the content included in the software. After putting in so much effort, it is extremely difficult to change direction when you have invested time and energy into what you already have. Wouldn't it be better to provide less content that you can easily tweak in several different directions once you know what questions your audience actually has?

Social Media

Social media can help you find out what questions your audience has, allowing you to connect with actual and prospective customers to base your content on what the customer needs to know. If your company maintains a user forum, this is the perfect place to pose questions about how your customers use your product. If your product is released early to select customers for a beta release, you should be included on the team to contact the customer for feedback on the content you provide as well as to understand what business problem the customer is solving when using your product. Make sure that a part of your planning includes time in the social media environment to keep you fresh and up to date.

Agility and Progress

By keeping your planning phases short and iterative, which matches the Agile development environment's hallmark attraction as a development philosophy, you spend less time making plans than traditional planning. This, in turn, keeps you nimble and able to respond to changes without negating the planning you've done. Shorter planning cycles also let you avoid analysis paralysis and maintain a bias for action. When you set goals for a week, then identify your daily outcomes to support those goals, you jump on doing something toward accomplishing your goals rather than over-planning. Using the Monday Vision, Daily Outcomes, and Friday Reflection cycle to set a week's goals by daily three- to four-outcome chunks, it's easy to see progress, as well as where you stray off track. This cycle also gives you time to change direction, if needed, when the product or feature changes direction. Your planning for results also gives you tangible results to show your team how you are progressing toward the overall goal of the larger project.

Conclusion

Planning in an Agile environment is critical to success. As you adjust to a shorter, iterative approach to product development, allow your planning to be shorter and get you on the road to achieving results more quickly. Remember, your plan is in sand and your goal is in stone. Adjust your plan as needed to achieve your goal.

Essential Steps in the Agile Planning Process for Personal Productivity

1. Adjust your mindset and your approach to planning.
2. Make plans each week using Monday Vision, Daily Outcomes, and Friday Reflection.
3. Focus on results and review your goals as you begin work on new tasks each day.
4. Change your goals for the week or day only when a truly higher priority goal is discovered during the week.
5. Measure your outcomes and use them in your next set of plans.
6. Be nimble, be agile, be productive. i

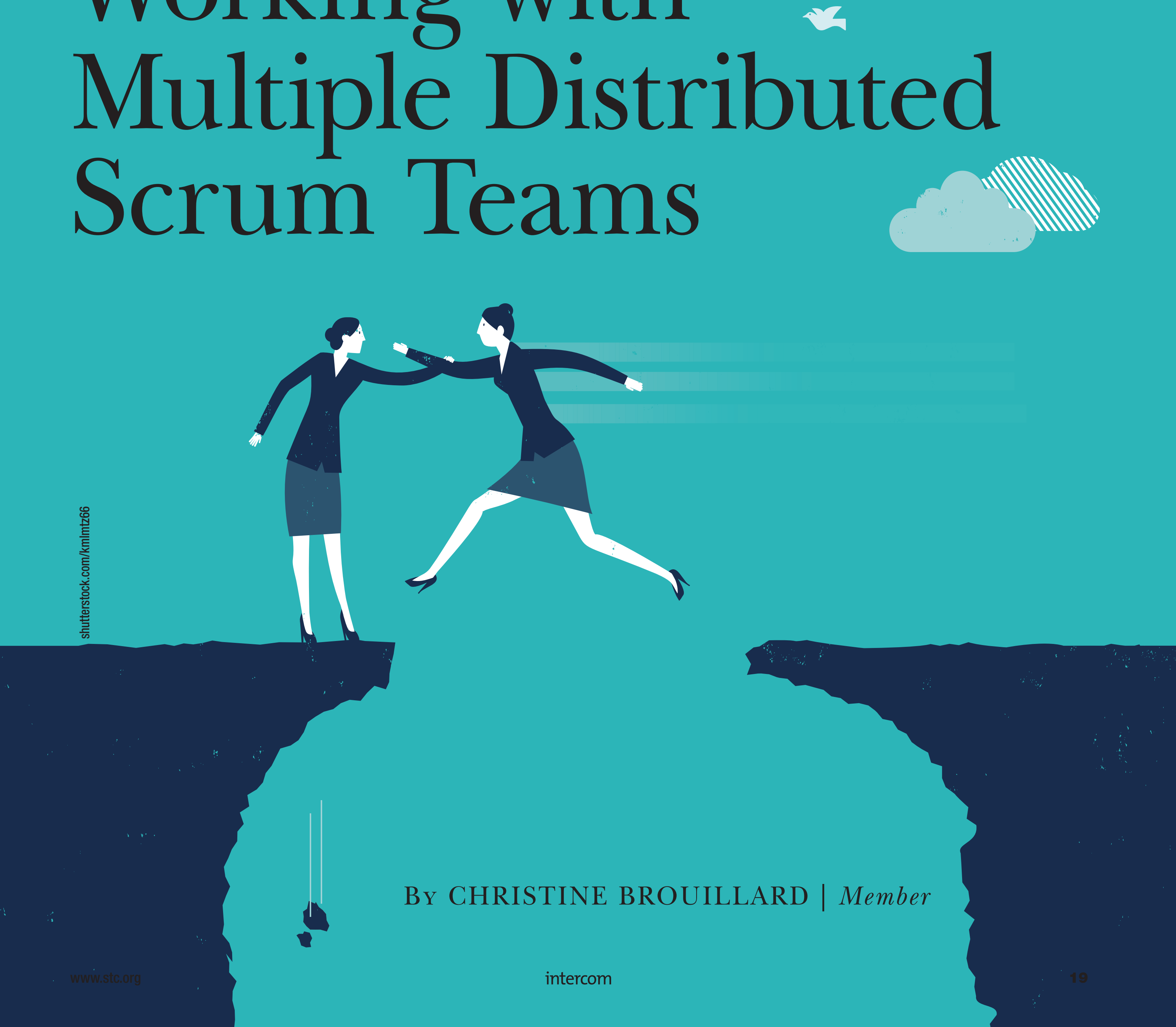
LEE TURNER *believes you are the product. She has been working in Agile development environments since 2002 and leading information development projects for NetIQ Corporation since 2009.*

SUGGESTED READING

Meier, J. D. 2010. *Getting Results the Agile Way.* Bellevue: Innovation Playhouse.

St. Laurent, Simon. "Writing Without Knowing," *O'Reilly Radar blog. http://radar.oreilly.com/2014/02/writing-without-knowing.html.*

Overcoming Us Vs. Them: Working with Multiple Distributed Scrum Teams



shutterstock.com/kmlmtz66

By CHRISTINE BROUILLARD | *Member*

NEITHER THE SCRUM GUIDE nor the Agile Manifesto—two of the foundational documents for Agile development methodologies—mention anything about distributed teams. But working with developers in remote locations is becoming more and more common. There are books, white papers, and blog entries devoted to the topic. Many of these writings focus on the increased need for communication in these situations. Technical communicators are expertly equipped to help bridge the great divide among distributed Scrum team members.

Adjusting to the New Normal

When my company adopted Scrum as our new Agile development methodology, we created twelve development teams with software engineers, product owners, and Scrum masters. Since we had only six members in the technical documentation department, some of us had to cover multiple teams. Some of the teams had programmers who lived in other U.S. states, but I was lucky enough to be assigned to two of the teams who had developers in Europe. On each of these teams, at least half of the resources are located in a distant time zone with English as a second language. Team 2 has three developers in Romania, Team 11 has four programmers in Switzerland, and the rest of each team is located in Connecticut. So how do you optimize this situation? How do you get the information you need for quality documentation?

Use Communication Tools

Communication can be a challenge even for Scrum teams who are co-located and regularly benefit from the nuances of face-to-face communication. Adding time zone and language differences just compounds the issues. Since sending all of the team to one location or the other on a regular basis typically isn't a viable option, the next best alternative is to use technology to bridge the gap.

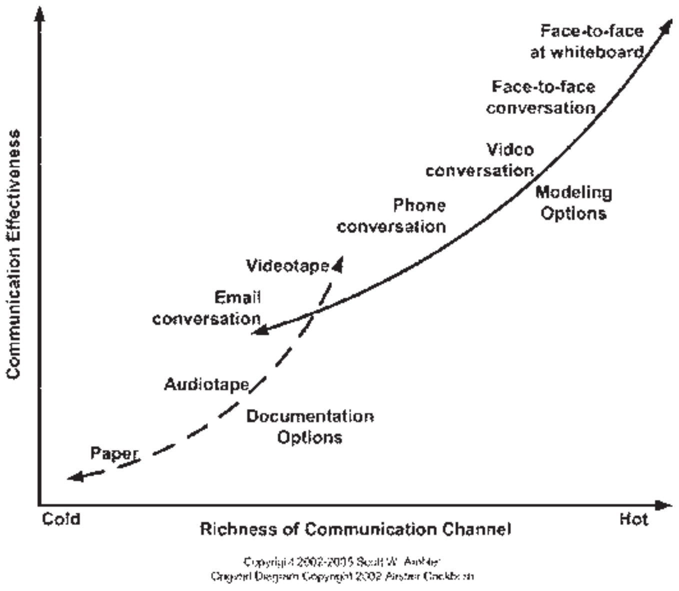


Figure 1. Modes of Communication

At CNC Software, we use video teleconferencing and GoToMeeting to create virtual meeting spaces. Our conference rooms in Connecticut are equipped with large plasma screens, wide-angle webcams, and high-quality conference phones. We have had some issues on the other side of the ocean with connectivity and equipment, but the developers in Romania and Switzerland now have their own webcams and headsets for attending meetings. The developers often use Skype for one-on-one conversations during each sprint.

Figure 2. An Agile team teleconference

For Shannon Wallner, senior technical writer at Four Js Development Tools in Texas, the idea was similar but the implementation was different. "The important thing I learned is to not introduce new tools, but to figure out how you can work with the tools your developers already use. For collaboration, I like GoToMeeting, but the developers use Skype, so we screen-share on Skype and I use a recording software called Evaer to record the meeting if needed. I get the information I need without putting an obstacle in their way."

A common data storage platform for the team is also helpful. Team 11 uses a combination of a Wiki and Google Docs to collaborate on projects and do team planning. All the teams store their backlogs and sprint data in VersionOne, which is often displayed during standups to help answer questions. Wallner's team uses a similar online tool. "I would like a more robust content management system, but we have instead learned to use JIRA. By working in the same platform, we feel more connected to the overall product plans and more in sync with the developers. Since moving to JIRA, we have improved our topic review processes because we are able to set up the reviews as JIRA tasks and assign them to developers no matter where they are in the world. By looking at how we can work with the tools familiar to them, we've made good strides in improving communication, collaboration, and the deliverables we publish."

Be the Translator

Even the best teleconferencing equipment won't always overcome language differences. Technical communicators can often put their skills to good use by trying to paraphrase from one side to the other. When you get to a sticking point, reiterate what the team member said and see if you got it right. Another method I use often is to start typing a document that the whole team can instantly view and revise. This information is often part of the research you need for your deliverables, so the effort helps both you and the team.

Videos are a great way to share information, especially on potential software defects. We often use the video tool in SnagIt or Jing to make quick videos to share. It is typically much faster than typing out all the steps to show the problem. Or if you happen to have the right people in the room, open the software and work through the problem in real time. We've found that these visual techniques really cut down on frustration levels.

Respect Differences

Just as you would research a new product you're writing about or a new company you're working for, take some time and research the other countries and cultures of the people you're working with. As Monica Yap noted in her white paper "Successful Distributed Agile Team Working Patterns," cultural barriers can be challenging to overcome. "When team members come from different cultural backgrounds, these differences can easily create misunderstanding and generate mistrust among team members."

One example of this happened during a sprint review with the team from Switzerland. In an attempt to give some constructive criticism on an area in the code, a developer in Connecticut used the word "silly" and the entire Swiss team was horrified. The team was uncomfortable around each other for weeks, and it took a formal apology to set things right.

Your research may include learning about their holidays, understanding their definition of work/life balance, or learning a few key phrases in their language. Being able to say "good morning" and "thank you" in Romanian went a long way with Team 2. But be careful and recognize that jokes, slang, and colloquial phrases in English may not translate well. There were many times that the team in Connecticut made a reference to a movie or song that the Romanians did not know. We often took this as an opportunity to educate them in American culture and ended up watching a short YouTube video or playing a song in iTunes. It made for team bonding moments.

Time differences are also important to respect. John Collins, senior UX strategist at Rosetta Stone, worked with a team that included members in Korea, France, Virginia, and Arizona. "Flexibility is obviously a key for all parties. Be willing to stay late or come in early occasionally. My localization manager worked much of this summer from Switzerland and compromised in his work hours, partway between the Swiss time zone and my time zone."

Find Common Ground

Finding places to meet doesn't just apply to time zones, but to team members as well. Establishing trust is a crucial component of a high-functioning Scrum team. As Monica Yap explains, "When a team doesn't possess a minimum level of trust, it's more difficult to deal with the challenges as they appear—it is often easy to blame and criticize the 'other' groups and the teams break down into competing tribes."

No matter where you are located, we all have celebrations and challenges in our lives. Use these life events to create connections between team members. On Team 11, we've sent gifts for new babies, sent flowers for funerals, and celebrated team birthdays. We joke about the numerous public holidays in Switzerland and make a big deal when we finally have a national holiday in the US. During the sprint retrospectives on both teams, I started the practice of adding a bonus question. It's always a more personal question to bring out information to connect the team members. Questions have included, "What was your favorite part of the summer holidays?" and "What's your favorite type of music?" When we asked the second question in Team 11, we learned that one of the Swiss developers plays in a brass band in his hometown and has recorded albums!

Wherever your Scrum team members are located, participate as much as possible and get to know them. Making connections helps communication during the Scrum process. i

CHRISTINE BROUILLARD *is a senior technical writer and has been at CNC Software, Inc. in Tolland, CT, since 1997. She has experience in management, project management, localization, and Agile practices. She became a Certified Scrum master in May 2013. She was named one of the 400 Most Influential in #Techcomm and #ContentStrategy on Twitter in 2011 (*@cebrouillard*). She was a speaker at the STC Summit in 2014 and the WritersUA East Conference in 2013. She has also been a judge for the International Society for Technical Communication competition.*

SUGGESTED READING

Cohn, Mike. 2010. *Succeeding with Agile: Software Development Using Scrum*. Ann Arbor, MI: Addison-Wesley Professional).

Eckstein, Jutta. 2010. Agile Software Development with Distributed Teams, Perfect Paperback.

Rubin, Kenneth S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Ann Arbor, MI: Addison-Wesley Professional, 2013.

Scrum.org, *https://www.scrum.org/*.

Scrum Alliance, *www.scrumalliance.org/*.

Starr, David. 2012. "Distributed Scrum," *http://msdn.microsoft.com/en-us/library/jj620910.aspx*.

Woodward, Elizabeth. 2010. *A Practical Guide to Distributed Scrum*. IBM Press.

Yap, Monica. 2010. "Successful Distributed Agile Team Working Patterns," *www.solutionsiq.com/docs/successful-distributed-team-working-patterns.pdf.*

shutterstock.com/Monkey Business Images

The Power of Technical Communication Skills in Scrum

By SARAH BACA | *Senior Member*

WHEN YOU LOOK AT the basic descriptions of a Scrum master and a technical communicator or writer, you might not see much in common between the two. On the surface, it seems as if technical communicators simply write down what they are told by a subject matter expert (SME) and Scrum masters ensure the team is following the Scrum framework and processes.

However, most of us who have worked as writers know that the technical communicator role is much more than writing down what we are told and making some pretty tables. In the same way, being a Scrum master goes much deeper than making sure a team is following a process. In my role as Scrum master, I have been able to use skills that I gained as a technical communicator, increasing my value to the Scrum teams.

Communication

I used to be a writer on a team with members in development, quality assurance, user experience, and product management. While we were creating software, team members had their own perspectives of the team's goals. We defined as much as we could from the beginning, but there was always more to discover. I worked with everyone on the team to try to determine what our use cases were and how the software should look when it was done. The funny thing was, I don't think my real contribution as a technical writer was the words that I wrote. My main value to the team was what happened when I asked a question and someone would say "Oh wow. I didn't think of it that way." Then off she would go to write a new test or modify the code she had just written. By constantly communicating with everyone on the team about the impact of our decisions on our users, I shifted the team to focus on a bigger picture—a mindset that benefited our users the most.

By constantly communicating with everyone on the team about the impact of our decisions on our users, I shifted the team to focus on a bigger picture—a mindset that benefited our users the most.

As Scrum master, I use the same skill set but I get to do it on a broader scale. I'm involved more in the everyday processes, so I can refocus the whole team on our users and the bigger picture more often. Because I'm no longer responsible for writing user documentation, I have more time to help facilitate team communication. I do still use my documentation skills—I just document processes instead of writing documentation for users. I still find that when I get an idea in black-and-white and in front of people's eyeballs, they still have the "Wow, I didn't think of that" moment. But now it's about a process that affects the whole team or even the whole company. Frequently, everyone thinks they are doing things the same way. After I challenge them, they discover they all understood it in a slightly different way. Also, by documenting processes, our team helps those who come after us know how we do things. Clarifying the processes for everyone means we don't have to waste time figuring out how to get things done; we can just get to work.

Connection

As human beings, we all crave connection. One of the traits that makes good technical communicators great is their ability to connect with SMEs on a level that goes deeper than asking the obvious questions. Great technical communicators know they have to connect on a human level so they can learn what questions to ask and move beyond superficial information. When I was a technical communicator, I found I couldn't get the information I really wanted until I had formed a meaningful connection with my colleague.

Once I was on a team documenting a product that was completely new to me. I was having a hard time learning why our user would use the product. We ended up locking ourselves in a room (this is called "team swarm" in the Scrum terminology). While I was talking to the developers about what we were building, we started using the white board and writing out what we were trying to do. Becoming immersed in the problem with the developers, I was able to connect with them as an equal member of the team. They were able to write software with a higher quality and I was able to write documentation that really helped the users of our new product.

Now, as a Scrum master, I facilitate the same connection between all the members of the team. I ask them questions to clarify what they are saying and help them make sure everyone understands. I ask managers to take a step back and allow the team to solve their own problems. I facilitate retrospectives that encourage the team to reflect on what they did during the last sprint and work together to improve. Retrospectives especially are highly valuable. They force the team to pause, assess what is working and what is not working, and improve. Constant improvement creates better software, happier customers, and more profit for the company.

Just as I did when I was a writer, I help teams identify patterns—what common themes are we finding in our problem solving? How can we think outside the box? How can we apply what we know about X to Y? This analysis is also part of what I did when I was a writer.

When we use the Agile Scrum framework, we are operating in a completely new way compared to traditional forms of business. We're shifting from a command-and-control environment to an environment where we have a team of people communicating with each other to find solutions, failing quickly, and getting stronger. This environment is perfect for a technical communicator who is constantly looking for patterns and trying to improve the clarity of communication. By improving team dynamics, I'm empowering my teams to have better communication and more transparency. I'm not only increasing productivity and making people happier in their jobs, I'm also helping create a higher quality product.

Work Management

Another important skill I developed as a technical communicator was to manage multiple projects at a time. Working on multiple teams meant I frequently had content deliverables due at different times, with varying complexity and audiences. I also had to manage the content itself from a large conceptual idea and then narrow it into granular instruction.

The Agile Scrum framework breaks down work in a similar manner. As a Scrum master, I help the team break down large units of work—such as a large feature—into units small enough to execute. Dividing work into smaller tasks is similar to writing good documentation—you make sure you have a standard format, make sure everyone is on the same page, and write it down. You facilitate communication so everyone knows what their teammates are saying and nothing is assumed. Writing down information and using a standard format makes it easier for the team to understand in the same way that it makes it easier for users to understand. The same principles apply for the team. By creating a template and making it clear to everyone what we are building, we build a better product. By breaking the work into right-sized chunks, everyone can pick up a piece, instead of spinning their wheels trying to figure out what to do next. Without breaking the work into manageable chunks, the team is paralyzed.

Authenticity

One of the most personal lessons I learned as a technical communicator was to be authentic. If I wasn't genuinely expressing my frustration, confusion, or excitement, it cheated everyone around me. If I needed help, they couldn't help me if I didn't tell them what was going on and I couldn't do the best job possible without their help. When I first started as a technical communicator, I would pretend to understand what people were saying around me even if I didn't understand. I quickly learned that this practice kept the documentation from being as clear as it could be. Unfortunately, it was really hard for me to admit when I didn't understand what someone was talking about.

One day I came up with an idea to help me feel more comfortable admitting that I didn't know something. My husband is a brilliant engineer, who also happens to be patient and kind. I went into the office and pretended that whichever genius I was interviewing from my team was just like my husband, and I took down the protective layer so I could "keep it real." I didn't pretend to understand if I didn't. Sometimes I even said to them, "Pretend I'm a fifth grader and explain it again." I quickly learned that they were usually happy to explain things to me in a way I could understand. Frequently, it helped them understand the topic better as well.

One of the most personal lessons I learned as a technical communicator was to be authentic.

Brene Brown writes in *The Gifts of Imperfection*, "Owning our story and loving ourselves through that process is one of the bravest things we'll ever do." For me, this is true—being honest about yourself is painful. Brown writes that the brain perceives rejection in the same area as it registers physical pain. To our brain, being rejected feels like getting stuck with a hot poker. The tricky thing is that if we don't risk rejection, we never grow and we never form the connections that are vital for our survival. I learned that I can't pretend to understand what is going on if I really don't. It doesn't matter if there's a C-level executive in the room or how stupid I look. What matters is that I am true to myself and being honest with those around me. I've known some very talented technical communicators who were so afraid to look stupid that they never dropped their shields and it kept them from succeeding. It's sad because they were never able to reach their full potential, and it was all because of fear.

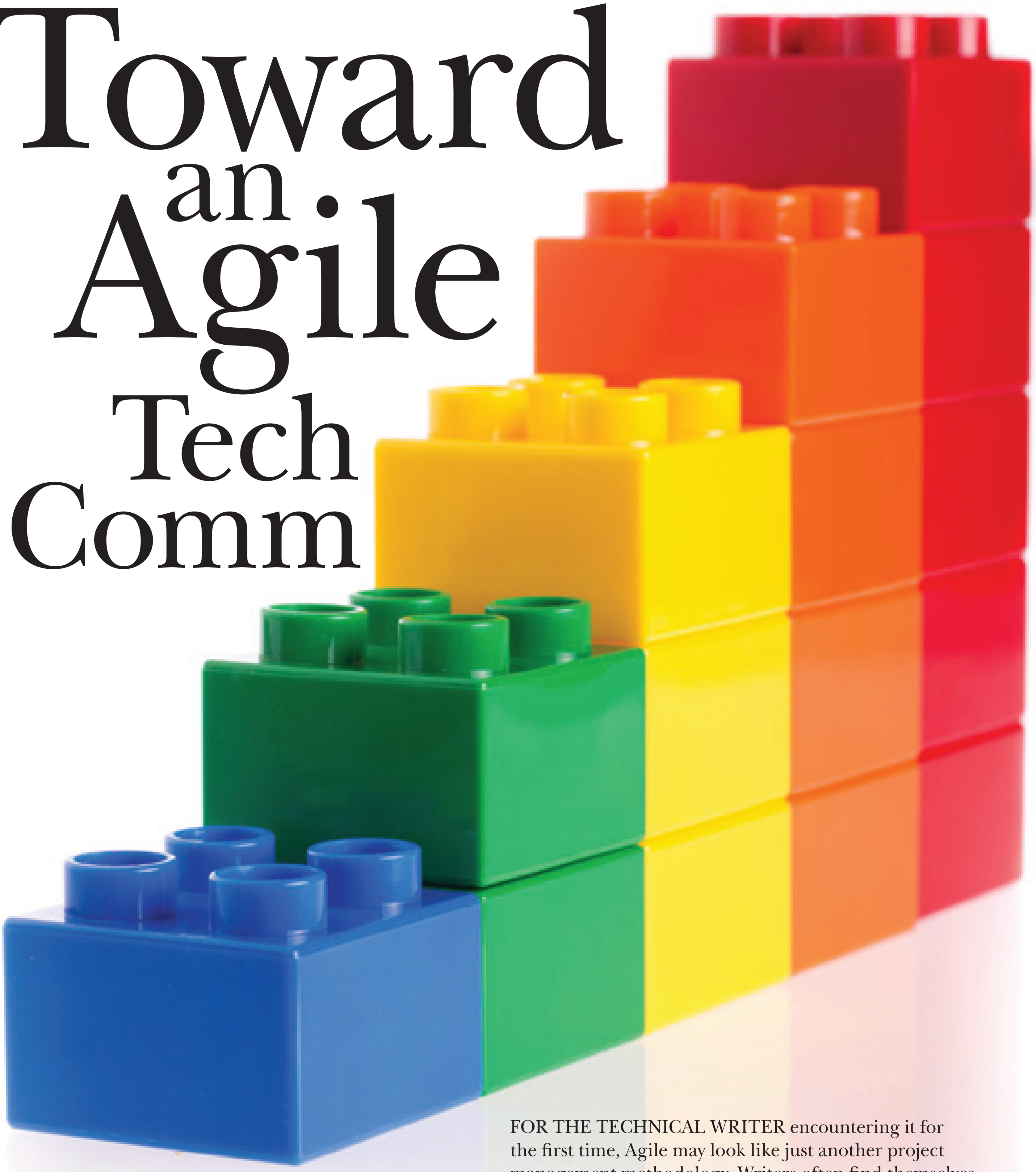
I use this same skill (being okay with looking stupid) as a Scrum master. I know if I don't understand or if I have a question, someone else is going to have the same question. The person might be a VP who doesn't understand the business case or it might be a team member who doesn't quite understand the product. But if I can be brave enough to show my ignorance, and to allow someone else to have the answers, it can really benefit the team.

Change

I know the move from technical communicator to Scrum master is not for everyone. Becoming a Scrum master has allowed me to use my favorite parts of being a technical communicator and spread those skills to benefit more people. Now that I'm a Scrum master, I can really see myself growing and helping those around me grow. That's the best feeling in the world! i

SARAH BACA *is the Scrum coach and Scrum master at Pentaho Corporation, a BI/DI software company. She has her degree in technical communication from the University of Central Florida and lives in Orlando, FL, with her husband and three boys. She enjoys being involved in the community so she can teach and learn from others. She is a former vice president and treasurer of the STC Orlando Central Florida Chapter and currently acts as co-leader of Agile Orlando. You can contact her on Twitter at* @sjbaca3.

Toward an Agile Tech Comm



shutterstock.com/tehcheesiong

By MARK BAKER | *Member*

FOR THE TECHNICAL WRITER encountering it for the first time, Agile may look like just another project management methodology. Writers often find themselves assigned to Scrum teams and attending daily standup meetings. They may not be exposed to, or even aware of, the much deeper changes that have taken place—or should have taken place—in the development organization.

The principles of Agile development include:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Achieving these goals requires a much more profound change than merely creating shorter development cycles. To deliver working software early and continuously, and to welcome change late in the development cycle, requires a different approach to designing and building software.

As the authors of *Service Oriented Architecture For Dummies* explain, traditional software systems had a common flaw:

> They weren't designed for substantive change. One pervasive characteristic that makes ... applications especially brittle—prone to break easily—is the extent to which the various programs that constitute the software applications are intertwined. These programs depend upon each other for little bits of code; and without such bits, they can't run.

To meet its basic goals, Agile needs to create software that *is* designed for substantive change. An Agile approach to technical communication should have similar goals and similar principles. To achieve this, it will need to create content that is designed for early and continuous delivery and for substantive change.

Tight cohesion and loose coupling are two of the most important principles used to build software that can be changed easily. An object is tightly cohesive if it is self-contained and handles its main functions internally. It is loosely coupled if it does not depend directly on the behavior of other objects to function. Jigsaw puzzle pieces are loosely cohesive and tightly coupled. Lego blocks are tightly cohesive and loosely coupled.

Loose coupling has become essential to software development on the Web. As Forrester notes:

> *Amazon.com* and its web-native cousins are the envy of many traditional organizations feeling pressure from competitors that innovate more rapidly. What's the critical enabler for rapid software releases? They require loose coupling of components and services, both within and between applications.

Indeed, the reason many companies are moving to Agile is so they can keep up with the rapid pace of software development and deployment required in a networked world.

The Web is also driving us into a world of loosely coupled content. In fact, the very word Web reflects the tenuous threads that connect content on the Web. The Web's great information sources from, Wikipedia to Stack-Overflow to Amazon, are not organized in rigid hierarchies but through the loose connections of links, likes, tags, votes, filters, and searches.

For loose-coupling to work, you need the individual components to have a high level of cohesion. Pete Goodliffe writes:

> Cohesion is a measure of how related functionality is gathered together and how well the parts inside a module work as a whole. Cohesion is the glue holding a module together.... Each module must have a clearly defined role, and not be a grab-bag of unrelated functionality.

If components do not have cohesion, they start to rely on each other in irregular and complex ways. In doing so, they become tightly coupled to each other. You cannot change one without compromising the functionality of another. And thus the ability to change the system quickly and substantially is lost.

Unfortunately, traditional technical communication deliverables and tools are loosely cohesive and tightly coupled. A typical manual, for instance, tends to have a tight coupling between its chapters and sections, but dive into an individual page and you will often find it is not cohesive. Rather than being self-contained and doing one thing really well, it is often hard to understand by itself and depends on what has come before and what comes after. Implementing substantive change in such a manual at short notice is almost impossible.

Tools also tend to encourage the model of a single deliverable organized in a single hierarchy: a tightly coupled model that is hard to change. Even tools that are intended to encourage topic-based writing often tightly couple those topics into specific structures that are hard to change without breaking other relationships.

The forces that are driving development toward Agile are affecting technical communication just as much as development. Readers in a networked, mobile, always-on world expect instant, up-to-date answers about products that are constantly changing and updating. We need documentation designs and architectures that are capable of substantive change in response to new demands. Going Agile, therefore, is something technical communicators should be doing independent of the development team's decision to do so.

To do this, we need to create content in which individual topics are highly cohesive and information collections are loosely coupled. Nicholas C. Zakas writes:

> Loose coupling is achieved when you're able to make changes to a single component without making changes to other components. Loose coupling is essential to the maintainability of large systems for which more than one person is responsible for the development and maintenance of code. You absolutely want developers to be able to make changes in one area of the code without breaking what other developers are doing in a different area of code.

Substitute “topic” for “component”, “writer” for “developer”, and “documentation” for “code” and this accurately describes what we need to do to maintain large documentation sets in a rapidly changing environment. To implement truly Agile technical communication, it is not enough to implement Scrum and stand-up meetings. We need documentation designs and systems that are loosely coupled and tightly cohesive.

How do we produce technical communication that is highly cohesive and loosely coupled? The answer, of course, is topic-based writing. But not all approaches to topic-based writing produce tight cohesion and loose coupling. Achieving these things requires a specific approach to topics that I call Every Page is Page One.

Increasingly, search is the primary means by which people find content. Social media also play a significant role, with people asking for help on forums and in social networks. Even when your content is not online, people bring their Web-trained information finding habits to your content, and go straight for the search box.

Whether people find your content through search or social media, they don’t land on your table of contents; they land on an individual page. How well that page works for the reader will depend on how internally cohesive that individual page is. If that page depends heavily on other content, the reader will quickly become frustrated.

This preference for small highly cohesive units of content mirrors what we have seen in the software world, where small cohesive apps and software-as-a-service have replaced monolithic desktop applications with the Web providing the glue that—loosely—couples them together.

To create an Agile technical communication process—one that can keep up with the demands and the pace of change, and that can harmonize with development’s Agile process—requires us to create content as small, highly cohesive units that are loosely coupled. Happily, that is exactly the sort of content that works best for the new ways people seek and use content.

Principles that Ensure Cohesion

The Every Page Is Page One information design pattern is intended to help you create content with high cohesion and loose coupling—content the works with the way people seek and use content today. In my book, *Every Page Is Page One: Topic-Based Writing for Technical Communication and the Web*, I describe the seven principles of Every Page Is Page One information design. The first four deal mostly with cohesion. They are:

A topic should be self-contained. It should do one thing for the reader, do it thoroughly, and do it well. For instance, a recipe is an Every Page Is Page One topic that contains everything a cook needs to prepare exactly one dish. If you are creating a recipe book, testing each recipe with diners will certainly lead to a better book—exactly the early and often user testing that Agile advocates. Also because recipes are self-contained, a failed recipe can be dropped from the book without affecting the overall book, requiring changes to other recipes, or delaying the project.

A topic should have a specific and limited purpose. It should serve a specific and well-defined reader need, and should not contain content that is not essential to that purpose. For instance, a recipe does not attempt to teach you basic cooking techniques. It focuses on the specific and limited purpose of preparing one dish. Defining a specific and limited purpose for each topic is the foundation of the discipline that keeps your topics cohesive and loosely coupled. Think of this definition as the user story that drives and defines your topic.

But be careful not to fall into the trap of simply documenting development’s user stories.

Developers’ user stories are sometimes more machine-oriented than business-oriented. Your topics may need to do more to help users bridge the gap between business needs and product features. Write your own user stories that define the specific and limited purpose topics must serve for the user.

A topic should establish its own context. It should not depend on the reader having reached it by following a table of contents or reading in sequence. It should be written as if it were the first topic the reader sees. It must make sure the reader knows exactly where they are and what the topic can do for them.

Look at almost any topic on Wikipedia and you will find that the opening sentences (and often the data sidebar as well) focus on locating the subject in time and space. This lets the reader know clearly where they have landed. A topic that establishes its own context cannot be broken when another topic is changed or removed. This allows you to make changes rapidly and with confidence when the product changes.

A topic should conform to a well-defined type. Topics that do one thing and do it thoroughly tend to have the same content and largely in the same order. The specific and limited purpose of the topic leads naturally to its specific topic type. Recipes, of course, conform to a well-known type. But look at short content on all kinds of subjects and you will find the same thing. The same kind of content is treated very much in the same way: similar fields, similarly expressed, in a similar order.

Having well-defined topic types helps to keep your topics cohesive by ensuring that they do the job they are supposed to do. It helps you develop topics faster, because it reuses design patterns and tells you exactly what needs to said in each case. It can also help with iterative or collaborative development, with different elements of the topic being created at different times or by different people.

Principles that Ensure Loose Coupling

The final three Every Page is Page One principles deal mostly with loose coupling. Even the most cohesive topic is not going to meet every reader’s needs completely, if only because readers often don't have all the background knowledge and experience they need to comprehend the task they are attempting. These readers will need to find other topics to fully understand their task. But each reader will need different topics and in a different order.

A loosely-coupled topic set does not impose a single reading order on all readers. It supports readers in choosing their own path through the content, secure in the knowledge that whatever topic they read next will work for them as a standalone topic in its own right—because all of the individual topics have a high degree of cohesion.

The principles that ensure loose coupling are:

A topic should assume the reader is qualified. A cohesive topic is written for a well-defined reader and assumes that its reader meets that definition. Imagine a recipe book in which every recipe told you how to whisk, crack an egg, or boil water. Every recipe would be 30 pages long and impossible to use. A recipe is written for someone who knows how to cook. But if you need to know how to do a basic cooking technique, like sweating onions, a quick Web search will find Every Page Is Page One topics that teach you just that skill.

Of course, your audience will include people with different levels of qualification. A tightly-coupled information set needs to know a lot about its audience, because it offers one principal route through the content which must be adapted to a particular audience level. But a loosely-coupled information set does not have this constraint. People with different levels of qualification will have different research strategies and will navigate through a content set differently using different topics to fill their personal information gaps.

This actually works best when each individual topic is written for the type of person who most commonly does the task it describes—when recipes are written for cooks. This allows for a natural progression and skill building without imposing a tightly coupled curriculum. This, in turn, preserves the ability to make changes in the content set with confidence at short notice.

A topic should stay on one level. Books often change levels, from the very general to the minutely detailed. A full documentation set will need topics on all of these levels, but each individual topic should stay on one level. It is up to the reader to decide when they need to change level in order to further their understanding. A recipe that requires sweating onions and merely mentions it in an instruction is on a different level from the topic that teaches you in detail how to sweat onions, but each stays on its own level.

An information set that plans when the reader will change levels is inherently tightly coupled. It is also making a decision for the reader that the reader would rather make for themselves. Staying on one level helps ensure the cohesion of individual topics. Loose coupling provides the means for reader to change levels when they please. Avoiding a mix of levels in individual topics ensures that the content is easy to change when changes occurs at a particular level of the system.

A topic should link richly. A loosely-coupled information set allows the reader to move in the direction of their own choosing. Cohesive topics make it possible for readers to come from anywhere and understand the topic. Links provide the loose coupling between topics that enables the reader to make that journey with ease. In an Agile environment it is important to have an effective link management strategy in place to ensure you can change content quickly without breaking links. However, ensuring that links express only a loose coupling between topics means that losing a link does not break the cohesion of a topic. High topic cohesion, therefore, makes managing links much easier.

By themselves, these principles describe how to make content useable for modern readers. That they also make it possible to create an Agile technical communication process is not an accident. Just as Agile software development is a response to the user's need for software that meets their changing needs quickly and effectively, Agile technical communication is about meeting rapidly changing information needs in a timely and efficient manner.

Scrum and standup meetings are simply a way of managing the design process that produces this kind of software and this kind of documentation. The real key to becoming an Agile technical communication organization is to change the way you design and architect content to support high cohesion and loose coupling through an Every Page is Page One information design.

MARK BAKER *is a 25-year veteran of the technical communication industry, with particular experience in developing task-oriented, topic-based content, and technical communication on the Web. He has worked as a technical writer, a publications manager, a structured authoring consultant and trainer, and as a designer, architect, and builder of structured authoring systems. He is currently president and principal consultant for Analecta Communications Inc. in Kitchener Waterloo, Canada.*

REFERENCES

Baker, Mark. 2013. *Every Page Is Page One: Topic-based Writing for Technical Communication and the Web*, XML Press.

Forrester Research, *Software Innovation Requires A Loosley Coupled Applications Architecture, www.forrester.com/Brief+Software+Innovation+Requires+A+LooselyCoupled+Application+Architecture/fulltext/-/E-RES116565.*

Goodliffe, Pete. 2014. *Becoming a Better Programmer*, O'Reilly.

Hurwitz, Judith. 2006. *Service Oriented Architecture For Dummies*, 2d ed. John Wiley & Sons, Inc.

Principles of Agile Development, *http://agilemanifesto.org/principles.html.*

Zakas, Nicholas C. 2012. *Maintainable JavaScript*, O'Reilly.

Community Achievement Award Applications Due 27 January

What Is a Community Achievement Award (CAA)?

THIS AWARD HONORS STC communities that have achieved success. There are three levels of recognition: Merit, Excellence, and Distinction. An award of Merit means that a community is a) running properly, and b) providing programs and services that further the mission of the Society. Excellence and Distinction awards are given to communities that go above and beyond the minimum requirements.

The CAAs will be presented at the 2015 STC Summit in Columbus, Ohio. Outstanding CAA honorees will also be awarded "Most Improved Community" and "Community of the Year."

Who Should Apply for the Award?

ALL STC COMMUNITIES (chapters, SIGs, and student chapters) should apply for this award!

Earning a CAA means that you have a successful community. You can post the award banner on your website, brochure, or other communications. It gives your community **credibility**. People are more likely to join your community if you have a good track record.

Why Should You Apply?

Take advice from community leaders who have applied for CAA and used it for community planning:

"Completing the CAA for our community is important for the following reasons:

- It helps us keep track of our accomplishments
- It helps us identify opportunities for improvement
- Completing the CAA helps us formally recognize **all** our volunteers who work hard to make our community successful."

—Jamye Sagan, past co-manager of IDL SIG

"NEO STC has earned a Community Achievement Award from the Society for over 10 years for its outstanding service to members.

These distinctions:

- Are impressive to spotlight on our NEO chapter website, include in marketing and promotional materials, and use for volunteer recruitment.
- Can attract nonmembers to check out chapter programs and make them into potential future members!
- Give the NEO chapter members a level of pride about their chapter. No matter what your level of participation is in the chapter, each NEO member plays a role in making these achievements come to fruition."

—Janean Voss, past president, Northeast Ohio STC Chapter

Tips for Applying

Plan Ahead: While it's too late to begin planning for the 2014 year, use the award criteria as a success plan to run your community in 2015. Assign a community member to keep track of accomplishments, and have council/board members report to that individual when they have accomplished something. An immediate past president/manager is often a good choice for keeping track of accomplishments.

Use Innovative Activities: If your community is doing activities you believe should count on your application, send a description of those items to the committee at least a week **before** the deadline, and **before** you submit your application—and allow the committee enough time to review them.

Review Excellence Criteria: Most successful communities are doing these activities. Take a look! You may be able to fill in more criteria than you know.

Review Distinction Criteria: Although there are only a few "Required Distinction" criteria, communities achieving distinction will have numerous activities completed in the optional categories. Communities of Distinction are voted in by the committee as truly outstanding from other communities.

What's New This Year?

The application has been simplified for your convenience**.** It now contains five sections instead of six. An up-front section tells you how to complete the application and where to send it.

We now have "Show Us" comments for each item. This explains what the committee wants to see for supporting materials and takes the guesswork out of interpreting the item.

Get More Information

As of the date of this article, the applications are still under review and need to be approved. In the meantime, keep using the 2013 application, because the criteria will not change much. Be prepared to cut and paste your answers into the 2014 document when it is ready.

The 2013 Community Achievement Award Guidelines and applications are available on the STC website at *www.stc.org/membership/recognition/awards/1221-community-achievement-award.*

Questions?

Contact the Community Achievement Awards committee (Chair, Tricia Spayer) at *stc.caa@gmail.com.*

Details on the 2015 Summit Official Hotel

THE 2015 SUMMIT takes place 21–24 June in Columbus, OH. This year the official conference hotel is the Hyatt Regency Columbus with all events taking place inside the hotel.

STC has negotiated a special conference room rate of $164 (plus 17.5% tax) for single or double accommodations at the Hyatt Regency Columbus (visit *https://resweb.passkey.com/go.soctech15*). All attendees who make reservations in the STC room block will receive complimentary Internet access in their guest room (a value of $9.95 per day). **IN ADDITION**, every room reserved in the STC's block will receive two vouchers for continental breakfast served just for STC Summit attendees. The continental breakfast will be served on Sunday, Monday, Tuesday, and Wednesday mornings.

According the hotel's website, the Hyatt Regency Columbus "is in the heart of the city, connected to the Greater Columbus Convention Center and … within walking distance of Columbus' best attractions. Hyatt Regency is one of the closest Columbus hotels to the trendy Arena District, a lively neighborhood with numerous bars, restaurants and shops. You can also walk to Huntington Park and Nationwide Arena, home of the Columbus Blue Jackets NHL team. Visit the Ohio State University Campus, State of Ohio Capital Building and Columbus' popular Short North Arts and Entertainment District."

Why stay at the official conference hotel?

Staying at the official conference hotel benefits both you and STC. Here are a few reasons to stay in the Hyatt Regency Columbus for the Summit.

- **Convenience.** All education sessions and Summit events are within the hotel, so no cabs or long walks needed to participate.
- **Networking.** There are so many informal networking opportunities beyond the official conference hours. Impromptu dinner parties start in the hotel lobby, hotel restaurants are full of Summit attendees, and the hotel bar is packed well into the night with all sorts of conversations, debate, and networking. You'll bump into technical communicators from around the world in all parts of the hotel.
- **Help keep Summit prices down.** Meeting our contracted room block enables STC to continue to negotiate affordable room rates for future meetings. Based on the number of rooms blocked, hotels will provide complimentary meeting space, which enables STC to offer discounted registration fees for a longer period of time.
- **Assistance from STC.** By staying at the official conference hotel, you'll have STC's assistance if there is a problem with your hotel reservation.

Registration Opens 1 December! Be sure to check out the summit website, *www.summit.stc.org*, for full details on Summit registration rates, including the low, Early Bird rate! Register early to get the best rates possible and ensure your spot at the premiere technical communication conference!

See you in Columbus!

TECHNICAL COMMUNICATION
SUMMIT '15
STC's 62nd Annual Conference
21–24 June 2015 • Columbus, Ohio

Share Yourself with *Intercom* Readers

WE'RE LOOKING FOR MEMBERS to contribute a first-person column for a future issue of *Intercom*. The magazine has a trio of member-focused columns: My Job (your day-to-day work), Off Hours (discussing your hobby or side gig), and Looking Back (historical pieces of interest). My Job takes a look at the day-to-day work of an STC member and what makes the job interesting, fun, or unique. Off Hours is a look at the side jobs and hobbies our members have. And Looking Back focuses on senior members providing perspective earned throughout their career. Would you like to share your story with *Intercom* readers? Email Marisa Seitz, *marisa.seitz@stc.org*, for more information, samples, and to volunteer!

Reminder of Deadlines for Awards and Honors

THE DEADLINES FOR NOMINATIONS for many of STC's awards and honors are upcoming. Please see the STC website, *www.stc.org*, for more information or to find out how to nominate someone.

- Community Achievement Awards: 27 January 2015
- Community Pacesetter Awards: 25 March 2015
- STC Election: 9–20 March 2015

Visiting Columbus

BY CHRIS HESTER | *Conference Chair*

LIKE MANY, I was surprised to learn that Columbus is the destination for the 2015 Technical Communication Summit. Columbus? Ohio? Because what could a small Midwestern city possibly offer us during our event? I was delighted to learn the answer is *quite a lot.*

Columbus is small but mighty—and mighty charming. The city has invested a great deal in money and resources to vitalize this city in the last 20 years, and it's evident in the cultural opportunities and activities that can appeal to everyone. During the Summit, take a break and enjoy the following:

- Art walks, for self-guided tours throughout various art and educational areas
- North Market, for a public market with offerings from local artists, jewelers, and merchants
- Jeni's Splendid Ice Creams and other food specialties
- Arena District, for sporting and entertainment (and karaoke!) venues
- German Village, for bookstores and art stores full of treasures, as well as German food, beer, and polka! Schmidt's Sausage Hause is a centerpiece.
- The Brewery District, for fantastic microbrews
- Columbus Zoo, Conservatory, and Museum of Art

Getting around Columbus is easy. Port Columbus International Airport serves the area, and the cab ride to the Hyatt is only 15 minutes. CBUS is the free downtown "circulator," with a convenient stop near the Hyatt. And for those more adventurous, Columbus offers CoGo, a bike-sharing program. You can rent a bike for touring on your own or follow one of their themed rides.

To learn more about Columbus, visit these sites:

- Experience Columbus, *www.experiencecolumbus.com*
- Downtown Columbus, *http://downtowncolumbus.com/*
- Columbus Arts, *http://columbusarts.com/*
- North Market, *www.northmarket.com*
- COGO Bike Share, *www.cogobikeshare.com*

The Summit in 2015 promises to be a great event, in a great city, and I'm looking forward to seeing you all there! i

Go for the Gold! Renew Your STC Membership with the Gold Value Package

RENEW YOUR STC membership for 2015 today with the Gold Value Package to take advantage of quick and built-in access to extra education and networking. New this year, Gold membership now includes 10 hardcopy issues of *Intercom* magazine (a $60 value) delivered directly to your door at no additional cost. That's right! STC is returning printed copies of *Intercom* magazine to Gold members only.

Receive free and discounted education, increased networking opportunities, a locked-in low rate for the 2015 Summit, and more. Pay once and have access to the resources you need when you need them. The Gold Value Package includes a wealth of benefits for the busy professional, all in one bundle.

Benefits of the Gold Value Package:

- **NEW!** Print copy of *Intercom* magazine
- Five free live webinars
- 20% off all online courses
- Early Bird pricing for the Summit regardless of your registration date
- All of the Classic membership benefits
- Membership in any or all of our special interest groups (SIGs)
- Membership in one chapter of your choice

When renewing your membership, simply click on the edit option next to your current membership type. This will allow you to select the Gold Value Package and enjoy all the benefits. You receive every benefit listed above, more than an $800 value, for only $200 more than a Classic membership. Go for the gold and renew with the Gold Value Package today! Contact *membership@stc.org* for more information. i

"Float"—CSS Feature and Symbol of Change

BY NEIL PERLIN | *Fellow*

SINCE I STARTED the Beyond the Bleeding Edge sessions at the Summit in 1999, I'm often asked how quickly something new *really* emerges to affect technical communication. It doesn't happen *that* often, but it does happen. For example, Microsoft's introduction of HTML Help at the WinWriters conference in 1997 turned online help in a new direction in an hour. This column discusses another such example of change, the CSS "float" property. The float property's impact won't be as quick or dramatic as that of HTML Help, but I think that what it symbolizes for technical communication is just as far-reaching.

The W3C introduced float in CSS1 in 1996 (see *www.w3.org/TR/REC-CSS1/#floating-elements*), but it remained little known in technical communication until responsive design, promulgated in 2010, came to tech comm through help authoring tools in early 2014. In this column, I'll briefly describe the float property's function before turning to the larger issue of what I think it symbolizes for technical communication.

What Is "Float"?

Float is a CSS property that lets us position elements on HTML pages. No big deal, you say. We've done that for years using simple inserts. True. What if we need multiple graphics side-by-side on a page? Again, no big deal. Create a table, hide its row and cell borders, and insert the graphics in adjacent cells in a row. Philosophically, using tables for page composition is wrong but it's worked fine for years. Until responsive design arrived.

Responsive design lets us create *one* (the crucial word) online output that automatically reformats itself for the device on which it's displayed. For example, say you have to create online help to run on large-screen devices like desktops and laptops and small-screen devices like smartphones. You can create two outputs, each designed for its target device. But now add support for 10" tablets? That's three outputs. Then 7" tablets? That's four outputs. At some point, you won't have enough resources to create and maintain all your different outputs. Responsive design solves this problem.

This column presents overviews of new technologies that may affect technical communicators in the near future. If you have feedback, or would like to suggest topics for subsequent columns, please contact Neil Perlin at *nperlin@concentric.net*.

Responsive design lets you create one output that can detect the properties of the device on which it's displayed and tailor itself accordingly. For example, three graphics laid out horizontally when displayed on a large screen automatically shift to a vertical format as the screen gets narrower. On the first screen in Figure 1 below, I inserted the first group of three graphics in a table. I inserted each graphic in the second group normally—not in a table—but specified a left float for it. On a wide screen, there's no difference.

As the screen narrows, however, like shrinking to tablet size, the graphics in the table can't shift because the table controls the layout. The result is the horizontal scroll bar. But the graphics in the second group, controlled by the float setting, automatically start shifting to a vertical format shown in the next image (see Figure 2).

As the screen narrows further, to smartphone screen size, the graphics in the table still can't move but the second set of graphics, controlled by the float setting, automatically shift to a full vertical format as shown in Figure 3.

Float also lets us move whole columns, change how text aligns next to images, and more. This is what lets us create the "fluid grids" that give responsive design much of its layout flexibility.

The float property can get very complex but it's easy to understand and apply for most online help and documentation projects. For example, to get the graphics to behave as shown in the figures above, the code is simply this:

```
<p>
   <p>
      <img src="Resources/Images/
lambo 1.png" style="float: left;" />
      <img src="Resources/Images/
lambo 2.png" style="float: left;" />
      <img src="Resources/Images/
lambo 3.png" style="float: left;" />
   </p>
</p>
```

Your help authoring tool's stylesheet editor lets you add the floats without getting into the actual code. You can start experimenting with it now.

What's the *Symbolic* Importance of the "Float" Property?

Why spend an entire column on a little-known piece of CSS code? In my opinion, float symbolizes four shifts taking place within technical communication.

1. *Online superseding print*—If you only output print, float is irrelevant. But if you output print *and* online or online only, you should optimize your content for online. Some techniques for doing so have been available to technical communicators for years, like using relative size units (em, rem, or %) for text rather than the familiar points, or embedding index entries. Float is the latest such technique to reach us. The more

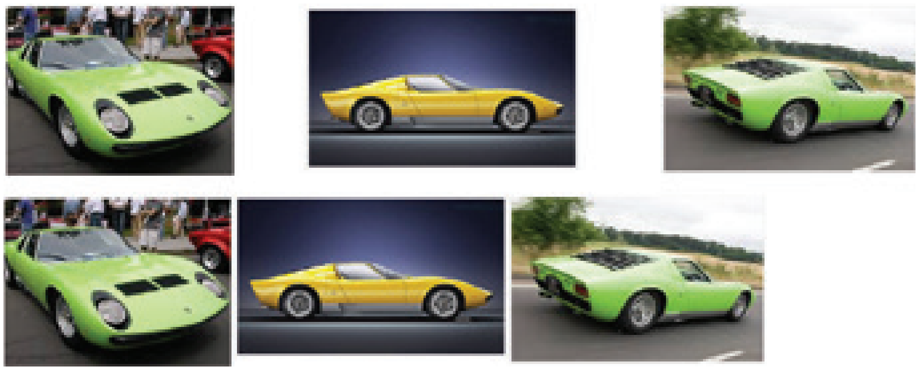


Figure 1. Float Demo for Wide Screen

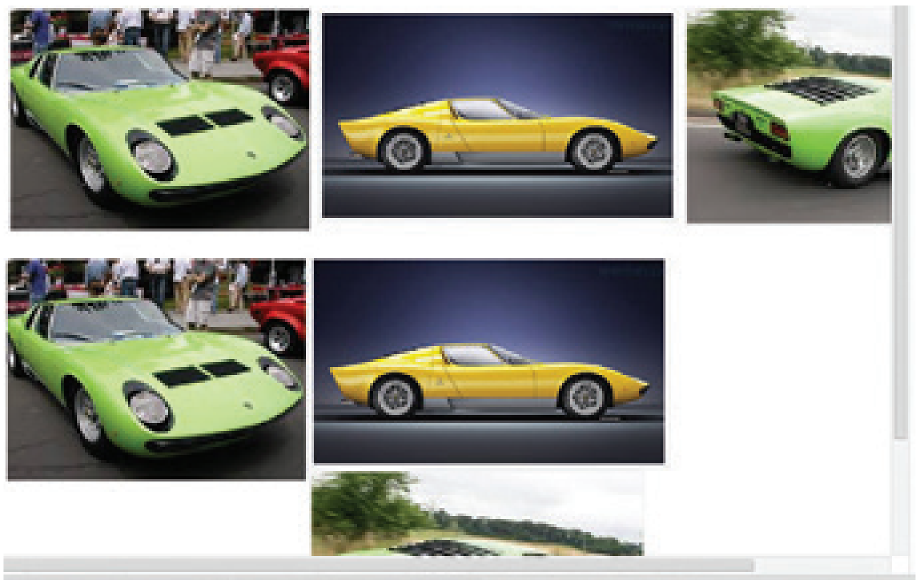


Figure 2. Float Demo for Tablet

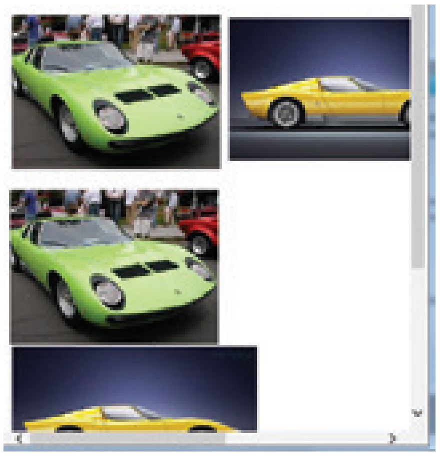


Figure 3. Float Demo for Smartphone

you think "online" first, the more in tune you'll be with changes in technical communication.

2. *Growing interest in output to mobile*—For years, computer screens were so large that we rarely had to worry about screen real estate. But growing interest in mobile as a single source output option makes it increasingly important to find automated ways to use the same content on different-sized screens. Programmatically controlled layout modification techniques like float do that and let us adapt quickly as new output requirements appear.
3. *HTML5 output*—HTML5 output doesn't need floated graphics. But one of HTML5's benefits is responsive design, and you'll want to look at using float to make the most effective use of responsive design.
4. *Increased development rigor*—Poor development practices like HTML tags with no end tags and tables created by using the tab key to create columns have largely vanished as our authoring tools and practices improved. Others, like local/inline formatting, are slowly vanishing as more authors use styles. The result is that improvements in programming practices are reaching increasingly esoteric areas like graphic positioning control. We'll never totally eliminate bad practices and "hacks," but we'll minimize them through techniques like float.

Summary

The symbolic effect of float is that it illustrates how Web coding practices are increasingly available to the technical communication world. As technical communication moves away from "writing" and toward "content" and the lines between Web development and technical communication continue to blur, techniques like "float" push technical communicators closer to Web developers, letting us future-proof our material and our jobs and get into increasingly interesting work. i

Thanks to Allen Beebe, Cheryl Landes, and Deborah Sauer for their comments.

NEIL PERLIN (nperlin@nperlin.cnc.net) *is president of Hyper/Word Services* (www.hyperword.com) *of Tewksbury, MA. He has 35 years of experience in technical writing, with 29 in training, consulting, and developing for online formats and tools including WinHelp, HTML Help, JavaHelp, CE Help, XML, RoboHelp, Flare, and others now almost unknown. Neil is MadCap-certified for Flare and Mimic, Adobe-certified for RoboHelp, and Viziapps-certified for the ViziApps mobile app development platform. He is an STC Fellow and the founder and manager of the Beyond the Bleeding Edge stem at the STC Summit.*

STC 2013 -2014 Salary Database

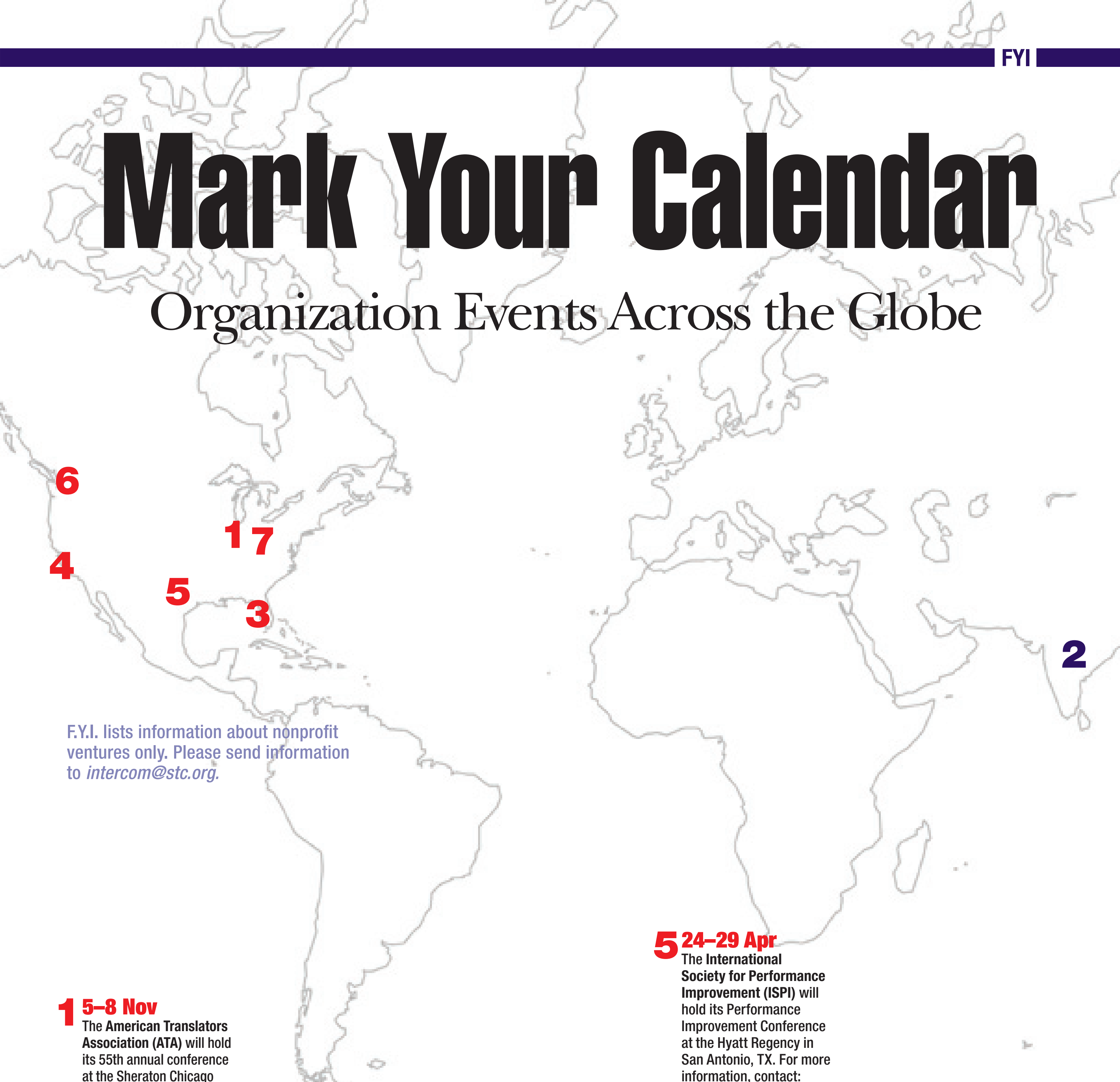
The 2013-2014 Salary Database is now available! The STC Salary Database is a tool that can be used to conduct more powerful job searches, make a strong case for a raise, or prepare department payroll budgets.

Visit www.stc.org/publications/products/salary-database to download today!

Sponsored by: Adobe Technical Communication

Mark Your Calendar

Organization Events Across the Globe



F.Y.I. lists information about nonprofit ventures only. Please send information to *intercom@stc.org.*

1 5–8 Nov
The **American Translators Association (ATA)** will hold its 55th annual conference at the Sheraton Chicago in Chicago, IL. For more information, contact:
ATA
+1 (703) 683-6100
ata@atanet.org
www.atanet.org/conf/2014

2 5–6 Dec
The **India Chapter of STC** will hold its 15th annual conference at the Vivanta by Taj hotel in Bangalore, India. For more information, contact:
STC India
www.stc-india.org/

3 26–29 Jan 2015
The **Annual Reliability and Maintainability Symposium (RAMS)** 2015 will be held at Innisbrook Golf and Spa Resort in Palm Harbor, FL. This year's theme is "Unleashing R&M Knowledge." For more information, contact:
RAMS
+1 (603) 863-2832
www.rams.org

4 12–16 Feb
The 2015 **American Association for the Advancement of Science (AAAS)** Annual Meeting will take place at the San Jose Convention Center in San Jose, CA. The theme is "Innovations, Information, and Imaging." For more information, please contact:
AAAS
+1 (202) 326-6450
meetings@aaas.org
http://meetings.aaas.org

5 24–29 Apr
The **International Society for Performance Improvement (ISPI)** will hold its Performance Improvement Conference at the Hyatt Regency in San Antonio, TX. For more information, contact:
ISPI
+1 (301) 587-8570
conference@ispi.org
www.ispi.org/AC2015

6 30 Apr–1 May
The **American Society for Indexing (ASI)** will be holding its annual conference in Seattle, WA. For more information, contact:
ASI
conference@asindexing.org
www.asindexing.org/conference-2015/

7 21–24 June
The **Society for Technical Communication** hosts the Technical Communication Summit in Columbus, OH. For more information, contact:
STC
+1 (703) 522-4114
http://summit.stc.org

* STC-related event

50 Years in STC

BY BILL LEAVITT | *Fellow*

AS I APPROACH 50 years in STC and 54 years in our profession, it seems appropriate for me to share what I have learned in that time. Like many of us, I didn't plan to be a technical communicator, but each change in my life moved me on in unexpected ways. For example, I attended Purdue University to become a chemical engineer. I took some aptitude tests to find out my strengths and what major to consider. I found out that while I had an aptitude for practical engineering, I also had an aptitude for writing. My advisor suggested technical writing, and I really enjoyed my classes in this subject. I earned a degree in technical writing with an area of specialty in chemical engineering in 1964. From my very first job, I have always worked in some form of technical writing.

How I got into STC. At Purdue, I was a feature writer for the student newspaper, and my boss, feature editor, Barbara Simmons, was also majoring in technical writing. After I graduated in 1964, she contacted me and convinced me that I should join the Society of Technical Writers and Publishers (STWP). Within two years of joining, she had become an officer in the Chicago Chapter, and she recruited me to become chapter treasurer in 1966. Soon after, I became finance manager for the 1967 STWP International Technical Communication Conference, which was held in Chicago that year.

In 1967, I became the Chicago Chapter first vice chairman, second vice chairman in 1969–1970, and then chairman (title later changed to president) in 1970–1971. I also served as symposium manager, nominating committee manager (1972–1973), and a second term as chapter treasurer (1973–1974). All those positions provided valuable experience for me in becoming an effective and successful leader.

The right time and place for Society-level leadership. In 1974, I attended my second international conference, held in St. Louis, to accept a writing award, which resulted in several fortuitous events. While at the conference, I had the opportunity to meet many STC leaders who convinced me that I was capable of serving at the board level. Because I had gained leadership experience and won an award, my company promoted me into management and gave me a generous raise.

Along with my leadership experience from the financial success of the 1967 international conference, my various chapter roles, and the support of many STC leaders, I ran for STC Treasurer in 1975. I was elected to three consecutive terms from 1975–1978, and STC experienced a period of tremendous membership and financial growth.

You can't win them all. The success of my work as treasurer, even though I was only 37 years old, led me to believe I could be a successful candidate for president. However, I was not nominated because the Nominating Committee thought I was too young and inexperienced. My STC friends convinced me to nominate myself by petition. However, I lost the election.

There is no way to know if I would have been a good president at that time, but losing the election gave me the opportunity to get married, start a family, and develop my career. During that "time off," I served STC as manager of the Nominating Committee and assistant to the president for publications.

Four years later (in 1983), I was nominated for the office of director-sponsor, which was a great development opportunity for me because I was responsible for sponsoring the leaders and members of the chapters in my geographic region. I learned a lot about the variety of problems regional and student chapters face, and I became a well-rounded leader.

After that I served in numerous Society board positions, including assistant to the president for member programs and as a member of the STC finance committee, and took on additional roles for my chapter. My STC work led to being elected an Associate Fellow in 1983.

In 1987, I was elected to the position of second vice president, which automatically acceded to first vice president, and then president in 1989–1990. I had the honor of being the first STC president to be installed in his home city.

Being STC president was definitely a peak in my career. Among our many accomplishments that year were holding STC's first international member reception, conducting formal research into STC's future success in international growth and influence, and helping Canadian members benefit from the new North American Free Trade Agreement (NAFTA).

Between 1975 and 1990, STC changed dramatically due to major shifts in industry. In July 1975, STC had approximately 2,900 members, but by July 1990, its net worth had grown and membership was over 13,000. Demographically, the membership also went from about two-thirds male to about two-thirds female, and our average age dropped by 20 years. And we grew from 48 chapters to 121. I'd like to think I was instrumental in influencing some of those changes, but I know that nearly 100 board members, committee managers, and STC employees worked together to achieve them.

I am often asked how I can devote so much time to STC over such a long period of time (almost every single year since 1964). The answer is that, first of all, I feel very fortunate to be in this profession. It was the right profession for me, and I care a lot about our members, about the success of our profession, and about getting the recognition that our profession deserves. I'm willing to donate a piece of my time every single year. I continue to have active roles in both my chapter and international STC.

Through my STC career I have learned what opportunities there are in STC and how one can take advantage of them. I hope that the success I have achieved will influence other STC members to strive to achieve all that they can.

Adobe

Save more with Adobe FrameMaker 12!

Find out how much you save by using FrameMaker

v/s using Word

Calculate savings

v/s using other XML editors

Calculate ROI

Request: Private demo | More information