

May/June 2014

www.TechWell.com

BETTERTM SOFTWARE

A TECHWELLTM PUBLICATION

CLOUD APP
DEVELOPMENT

What it takes to build for
the new enterprise

DATA WAREHOUSE QUALITY

Best approaches to attack
quality issues

AVOIDING

•The•

prioritization
Trap



CONTENTS



18



22

in every issue

- Mark Your Calendar **4**
- Editor's Note **5**
- Contributors **6**
- From One Expert to Another **13**
- TechWell Spotlight **21**
- Product Announcements **30**
- FAQ **31**
- Ad Index **33**

Better Software magazine brings you the hands-on, knowledge-building information you need to run smarter projects and deliver better products that win in the marketplace and positively affect the bottom line. Subscribe today at BetterSoftware.com or call 904.278.0524.

features

14 COVER STORY AVOIDING THE PRIORITIZATION TRAP

With incoming priorities being requested by just about everybody, how in the world can you and your team prioritize? Brandon shows you some innovative techniques that you can use to turn chaos into order. One surprising approach is simply handling priorities on a first-in, first-out basis.

by Brandon Carlson

18 IS SOFTWARE CONFIGURATION MANAGEMENT TECHNOLOGY REGRESSING?

There are ever-growing ways to organize your project assets with public domain configuration management tools. There's a mistaken belief that these free software configuration management (SCM) alternatives can be just as powerful as leading commercial tools.

by Joe Farah

22 DEVELOPING CUSTOM APPS FOR THE CLOUD

With the cloud providing tremendous freedom like instant deployment of updates, you're definitely going to have to adjust how you develop and deploy apps. Pete and Matt have created a list of things you need to consider when developing apps for the cloud.

by Pete Morano and Matt Stratton

26 ATTACKING QUALITY ISSUES IN DATA WAREHOUSING

To fully detect, isolate, and resolve quality issues in a traditional, large-scale data warehouse requires that several approaches be used together. Wayne identifies types of data quality issues and then illustrates how to best attack and resolve those pesky issues.

by Wayne Yaddow

columns

7 TECHNICALLY SPEAKING WHAT'S A PROFESSIONAL?

by Johanna Rothman

For years we've all heard how software development and IT are a mixture of art and science. As our industry matures and becomes more mainstream, Johanna wants to upset the apple cart by suggesting that there's a missing and sorely needed ingredient—professionalism.

32 THE LAST WORD WHY AM I ALWAYS GETTING BAD NEWS IN THE ELEVENTH HOUR?

by Kenton Bohn and Ryan McClish

This article is a departure from previous columns. Kenton and Ryan role play the stress and friction between a typical product manager and an engineering team lead. This article may make you squirm, but it brings out the issues of teams attempting to do the best thing from completely different perspectives.

MARK YOUR CALENDAR

software tester certification

<http://sqetraining.com/certification>

Foundation Level Certification

May 4–6, 2014

Orlando, FL

May 6–8, 2014

Vienna, VA

May 12–14, 2014

San Diego, CA

May 20–22, 2014

Dallas, TX

June 1–3, 2014

Las Vegas, NV

June 9–11, 2014

Chicago, IL

June 17–19, 2014

Tampa, FL

Advanced Tester Certification

August 25–29, 2014

Chicago, IL

October 27–31, 2014

Washington, DC

training weeks

<http://sqetraining.com/trainingweek>

Testing Training Week

May 12–16, 2014

San Diego, CA

June 9–13, 2014

Chicago, IL

Agile Software Development Training

June 1–3, 2014

Las Vegas, NV

conferences

STAREAST

<http://stareast.techwell.com>

May 4–9, 2014

Orlando, FL

Rosen Centre Hotel

STAREAST Virtual Conference— Free Event!

<http://stareast.techwell.com/virtual-conference>

May 7–8, 2014

Streaming Live from STAREAST in
Orlando, FL

Agile Development Conference West

<http://adcwest.techwell.com>

June 1–6, 2014

Las Vegas, NV

Caesars Palace

Better Software Conference West

<http://bscwest.techwell.com>

June 1–6, 2014

Las Vegas, NV

Caesars Palace

STARWEST

<http://starwest.techwell.com>

October 12–17, 2014

Anaheim, CA

Disneyland Hotel

Agile Development Conference East

<http://adceast.techwell.com>

November 9–14, 2014

Orlando, FL

Walt Disney World Dolphin

Better Software Conference East

<http://bsceast.techwell.com>

November 9–14, 2014

Orlando, FL

Walt Disney World Dolphin

Publisher
Software Quality Engineering Inc.

President/CEO
Wayne Middleton

Director of Publishing
Heather Shanholtzer

Editorial

Better Software Editor
Ken Whitaker

Online Editors
Cameron Philipp-Edmonds
Beth Romanik
Jonathan Vanian

Production Coordinator
Donna Handforth

Design

Creative Director
Catherine J. Clinger

Advertising

Sales Consultants
Daryll Paiva
Kim Trott

Sales Coordinator
Minda Crosby

Marketing

Marketing Manager
Jonathan Greene



CONTACT US

Editors: editors@bettersoftware.com

Subscriber Services:
info@bettersoftware.com

Phone: 904.278.0524, 888.268.8770

Fax: 904.278.4380

Address:

Better Software magazine
Software Quality Engineering, Inc.
340 Corporate Way, Suite 300
Orange Park, FL 32073



WE VALUE YOUR FEEDBACK

I'm particularly proud of the assortment of articles in this latest edition of *Better Software* magazine. With the trends toward big data and cloud services, I hope you all enjoy Wayne Yaddow's data warehousing article and Pete Morano and Matt Stratton's cloud app development feature. Brandon Carlson's cover story attacks the thorny project prioritization issues we all face. Joe Farah also takes a probing look into the current state of software configuration management.

We're now making it possible for you to comment on our articles. Using the previous issue for March–April 2014 as an example, Jon Hagar's article has a StickyNotes hypertext link at the end.

The screenshots illustrate the workflow: a 'Sticky Notes' box on an article points to a 'References' link, which leads to a 'REFERENCES' section on the StickyMinds.com website. From there, users can navigate to the 'USER COMMENTS' section for the latest article, where they can leave a comment.

Clicking on the StickyNotes References link will redirect your browser to our StickyMinds web page, which allows you, as a StickyMinds.com member, to leave a comment that everyone (including the author) can view. Every week an article from the latest *Better Software* issue is highlighted on the StickyMinds.com main page at <http://www.StickyMinds.com/latest>, where you can also submit comments to engage other readers, subscribers, and the author. We truly value your feedback. Let us and our authors know what you think of the articles by leaving your comments. I sincerely hope you enjoy this issue!

Ken Whitaker
 kwhitaker@sqe.com
 Twitter: @Software_Maniac

Contributors



SANJIV AUGUSTINE is an entrepreneur, industry-leading agile and lean expert, author, speaker, management consultant, and trainer. For more than thirteen years, Sanjiv has served as an advisor to executives and management at leading corporations. He is the author of *Managing Agile Projects*, founder of Lean Startup in the Enterprise Meetup, and a founding member of PMI's Agile Community of Practice. Sanjiv still personally manages and coaches agile projects, serves as product sponsor for LitheSpeed products, and is an active agile workshop and conference presenter. Sanjiv can be reached at sanjiv.augustine@lithespeed.com.



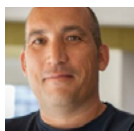
As vice president and executive partner at Geneca, **KENTON BOHN** provides leadership and strategy for its software product development collaboration with new and existing clients. With more than twenty years of experience as an entrepreneur and technology executive, Kenton is passionate about helping individuals and organizations deliver game-changing business outcomes with custom software. You can contact Kenton at kenton.bohn@geneca.com.



A self-proclaimed nerd, **BRANDON CARLSON** works for Lean TECHniques Inc., an IT consultancy that helps teams deliver high-value, high-quality products to market. Passionate about elevating IT performance, over the past twenty years he has helped numerous organizations from startups to Fortune 100 companies improve their product development and delivery systems. Brandon's interests include behavioral psychology and software development professionalism. He can be reached on Twitter and just about everywhere as [bcarlso](https://twitter.com/bcarlso) and at bcarlso@leantechniques.co.



President and CEO of Neuma Technology, **JOE FARAH** is a regular contributor to CMCrossroads. Prior to cofounding Neuma in 1990, he was a director of software at Mitel. In the 1970s, Joe developed the Program Library System (PLS), still heavily used by Nortel (Bell-Northern Research). He's been a software developer since the late 1960s. Find out more about Neuma at neuma.com.



PETE MORANO is the vice president of application development at 10th Magnitude, where he manages the development team and leads the technology direction. With more than seventeen years' experience leading teams at several innovative technology firms, he supports open development models that stimulate communication, creativity, and innovation. In his free time, Pete enjoys cycling, marathon canoeing, and organizing hackathons. You can reach Pete at pmorano@10thmagnitude.com.



As a client partner for Geneca, **RYAN McCLISH** provides guidance and support to Geneca's project teams. He is an experienced and insightful technology leader in delivering custom software to market. Drawing from his broad range of roles, Ryan is key in building lasting client relationships, engagement management, solution development, and team mentoring. Contact Ryan at kenton.bohn@geneca.com.



JOHANNA ROTHMAN, known as the Pragmatic Manager, helps organizational leaders see problems and risks in their product development. She helps them recognize potential "gotchas," seize opportunities, and remove impediments. Johanna is working on a book about agile program management. Johanna writes columns for [StickyMinds](http://StickyMinds.com) and ProjectManagement.com, writes two blogs on her jrothman.com website, and blogs on createadaptablelife.com. Contact Johanna at jr@jrothman.com.



MATT STRATTON is a managing consultant and DevOps evangelist who leads the infrastructure team at 10th Magnitude. Matt brings more than fifteen years of technology operations experience from a wide range of companies, including large financial institutions and e-commerce sites. When he's not helping clients improve their agility and velocity, he's either cohosting the biweekly Arrested DevOps podcast or watching *Doctor Who*. You can email Matt at mstratton@10thmagnitude.com.



CHUCK SUSCHECK is a nationally recognized agile leader specializing in agile software development adoption for the enterprise. He is one of only eleven trainers worldwide and three in the US certified to teach the entire Scrum.org curriculum. With more than twenty-five years of experience, Chuck has held key positions at some of the most recognized companies in America. He has spoken at national and international conferences such as Agile 200X, OOPSLA, and ECOOP on topics related to agile project management and is a frequent author in industry and academia. He can be reached at charles.suscheck@juniperhillassociates.com.



WAYNE YADDOW is a computer testing professional who began his career with IBM, where he worked as a Z/OS mainframe developer and tester for fifteen years. He became a QA consultant working primarily in the financial industry in NYC with Standard & Poor's, Credit Suisse, Citigroup, and JPMorgan Chase, where he focused on business intelligence, data warehousing, data migration, and data integration testing projects. Wayne can be reached at wyaaddow@aol.com.

What's a Professional?

You're an expert at your job, always considering the needs of the customers, an expert on how to collaborate with the team and all of a project's stakeholders, and motivate teams to deliver on time.

Is that really enough?

by **Johanna Rothman** | jr@jrothman.com

As I work with teams and organizations transitioning to agile, they say things like:

“I don't see how we are going to have the time to do all of this testing.”

“We have to start coding. We have no time for planning.”

“How can we start without knowing all of the requirements up front?”

“How can we start without designing a full architecture up front?”

For many people, agile approaches challenge everything about what it means to be a working professional. When I teach my project management workshops, I build in experiential practice. We work on the team's stories. We work on the team's releases. We practice on their work.

By the time the workshop ends, participants have had practice on live products. They experience how to create user stories and estimate in their product in real time. They experience what agile feels like.

With the trend toward software craftsmanship, software developers are re-examining what it means to be a software professional. That's good. That examination often results in concern. That's OK, too.

Agile asks a lot of the technical staff. It also provides the technical staff room to be the best they can be. If you are a product contributor on an agile project, your job is to get the story to completion. That is your entire job. It doesn't matter if your part is done. That's irrelevant. Your job is to get the story to done. And that changes the entire game.

That means it's fine to say, “I looked at the code in this area and I see this technical debt. I can code around it for this story. And I'm putting a card on the backlog, because when we do the next piece of the feature, we are going to need to

address this technical debt.” In effect, you opened the closet door, looked at the mess, and closed the door, but made a note of it for later.

Or, if you're a developer and you can see that the testers don't have enough test automation; rather than twiddle your thumbs at the end of the iteration, why not ask, “What actions would help you now? Do you need anything from me or the team?”

The testers might say, “Hooks, please, here and here.” Or it could be something like, “We don't know how to start with automated testing with this beast. Can we pair for twenty minutes and you can suggest something?”

If the team members aren't sure what to ask, consider offering suggestions as starting points. That act demonstrates leadership. I consider it to be professional.

If you're a tester you can say, “How can I help write acceptance

criteria for a story that would help the developers get to done faster?” or “What tests can I write that would help the developers know that their code works?”

In agile, there is no “them-developers” versus “us-testers.” If you need a separate audit department, that's fine. But that group is typically not the testers who are part of the product development team. That helps speed the development of the product. The developers may never see the product from the “what can go wrong” perspective, as testers often do. The testers may never see the product from the “here's how it should work” perspective, as developers often do. Working together, they produce a working product—as good as it can be.

If the developers, testers, writers, UI, database folks, and everyone who needs to be there are working together to move stories across the board and they are working professionally, don't they need oversight to be told to work professionally?

“With the trend toward software craftsmanship, software developers are re-examining what it means to be a software professional. That's good.”

Uh, no. That's why agile teams have a definition of done. Done defines what professional means for your project.

No one can make you work less than professionally if you have defined done.

Let's say your team has defined done as the code is checked in, has been reviewed by at least one other person, has automated unit tests, has automated system tests, and those system tests are checked in. Now, along comes someone who wants you to shove "one more story" into an iteration. I know this

doesn't happen to you, but it might happen to your buddy. Play along with me here.

If that feature doesn't get to done, you can't shove that feature into the iteration. No one can force you to do so—not if everyone works as a team.

You don't have to be agile to work this way—any team can work to complete features as a cross-functional team. In agile, however, the discipline of completing the feature means the entire team stands unified and firm and says, "No, this feature doesn't meet our definition of done. You can't have it yet."

When this response occurs, you know you have met the promise of agile.

Keeping your iterations and features short enough that you get everything to done on a regular basis allows your product owner and your managers the ability to change what they want to do fast enough.

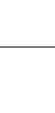
Agile is all about the ability to change. It's not about the ability to release faster, and it's not about predictability.

Agile is about the ability of a product owner to say, "You know, I thought I wanted that feature next. I was mistaken. I want this feature next. Now that I'm at the beginning of the next iteration, this is the ranking I want," or "Now that I see the demo, this is the ranking I want for the kanban."

How do you get that? By being professional.

This means having the courage to complete features all the way by not unnecessarily adding features (gold-plating), by helping your fellow teammates, and by moving one story at a time across your visual board.

That, my friends, is what being agile is all about: being professional. **{end}**



Ken Schwaber

Years in Industry: 45

Email: ken.schwaber@scrum.org

Interviewed by: **Dr. Chuck Suscheck**

Email: charles.suscheck@juniperhillassociates.com

"I see the edges being eaten away by people wanting more certainty—even within Scrum there are conversations about how do we get more predictability of velocity, and the effort that people put into the sprint plan meeting trying to be more certain of what we are going to deliver at the end of the sprint."

"The ability to both deal with the technology difficulties and the constant request for changes led me to start coming up with the ideas in Scrum and agile."

"I think the biggest danger to Scrum and agile thinking is the desire for predictability and that habit in middle management of IT development organizations. They still haven't gotten on board because we haven't really given them a role in this whole thing."

"Back then we were building the first ALM tools in Burlington, Massachusetts, but the tools weren't working very well. They were online and they weren't adopted. So we looked at a way to connect the methodology to the tools so that they

"I think the next big thing is evidence-based management (EBM), to show the value of doing work one way or another—not only for businesses to see the value, but also if they don't act on the value, it should be clear that they are doing things that are not beneficial to their corporation."

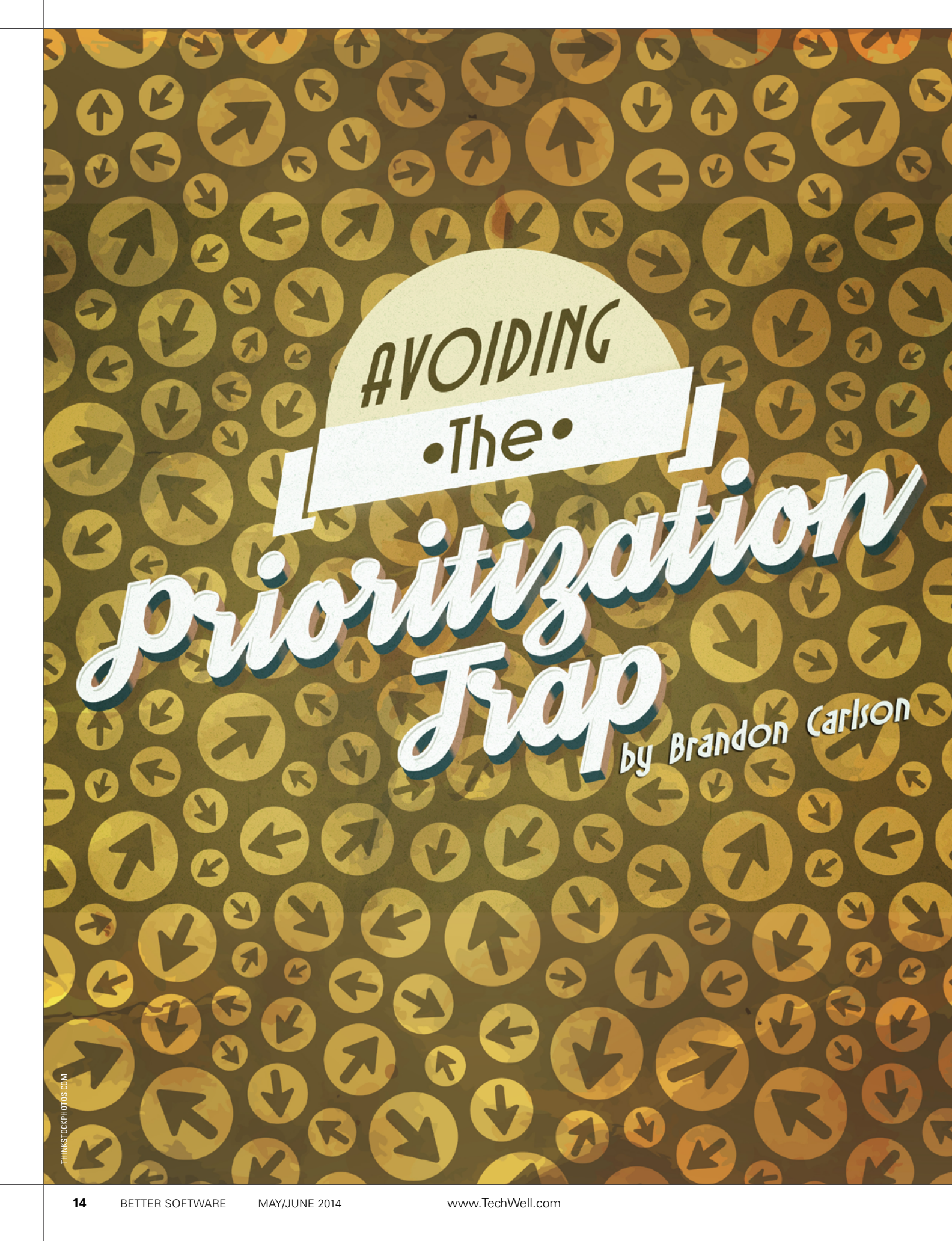
“

You can't say, 'Oh, this is 17.4 ounces of value,' but you can start setting baselines so you can compare trends and predict based on our actions what will happen over time.

”

"The latest data from Forrester Research says 92 percent of agile organizations are using Scrum. It doesn't say how many organizations are developing software just with Scrum and how many are using modern engineering techniques too. I think if we ask the other question, the answer might be closer to 30 or 40 percent, which is still pretty reasonable."

For the full interview, visit <https://well.tc/FOETA16-3>



AVOIDING
•The•
prioritization
Trap

by Brandon Carlson

THINKSTOCKPHOTOS.COM

Many organizations adopting agile run into the prioritization trap, where work is significantly more difficult—if not impossible—to plan due to the increased variability introduced into the backlog by the prioritization process.

Using the First-In, First-Out Approach

One organization I know escaped the prioritization trap using a first-in, first-out model for one of its shared services teams.

The sustained engineering group was overworked and customer satisfaction was trending the wrong direction. The primary complaint was the inability of the organization to reliably provide a release date for any new issues entered into the system. Something needed to be done to improve the situation, but what? They tried almost everything: creating a triage sub-team for up-front estimates, multiple prioritization schemes, regular update calls with customers, and countless other tactics. Unfortunately, none of these approaches worked. At one of the regular customer service/IT meetings to touch base, a conversation took place that would change the direction of sustained engineering and customer satisfaction for good.

It all started when one of the members of the IT team suggested developing a value stream map for the entire customer service process to help generate ideas for a solution to the predictability problem. The team mapped the flow of work from first customer contact to production release, initially focusing on handoffs and queues. While mapping the flow was easy, putting wait times between each workflow step was more difficult. As the team discussed each queue, the phrase “it depends on the priority” was repeatedly coming up, and they decided to explore the idea further.

What would their value stream look like if they did away with the concept of priority? The idea sounded preposterous, but it kept coming back. Could it really work? The team decided to take a look at recent customer problem escalations. In almost every case, the customer was upset because the team didn’t deliver the fix according to the expectations identified by customer service. Upon further investigation, the schedule was typically missed because a higher priority request took precedence. The team definitely found a key component of the predictability picture, but they didn’t know what to do about it.

At the next meeting, a customer service team member brought along some additional information that seemed to substantiate the impact of prioritization on customer satisfaction. He even brought a direct quote from the customer: “I don’t know why I should trust you to get this done—you guys can’t even fix spelling errors!” (Of course everyone on the team knew that cosmetic errors such as this were considered low priority and typically never actually got worked on until developers stumbled upon them.)

In light of this additional evidence, the team decided to tackle this issue head-on rather than treating it merely as interesting information. One developer asked, “What if we worked the tickets in a first-in, first-out model?” The response was expected: Customer support continued to emphasize very real

scenarios of customer down and management escalation. As glorious as it was in theory, a first-in, first-out (FIFO) model could not withstand the first contact with an upset customer. The team began discussing using a hybrid approach—one that would eliminate most prioritization but allow for exceptions when necessary. After some additional discussion, they decided that the best way to do that was to introduce a new queue.

The queue was a contentious topic within the team, and for good reason. During the value stream mapping exercise, the team had already identified at least three queues that were being used in the current process. Without proper queue management, the new queue could become just as unwieldy as the others that led to this mess in the first place. After a bit of discussion, the team agreed that the priority queue could not easily be managed using the FIFO model, so the traditional prioritization process would be the best approach for items placed there.

In addition, to prevent the model from becoming flooded with requests, the team agreed to control the size of the queue by applying a work-in-progress limit. This limit would serve two purposes. First, it required the customer service group to be judicious about which items to prioritize, and second, based on the team’s current capacity, using a limit of three provided a good balance between FIFO and priority work. The team decided to give this new queue a name, and the escalation queue was officially born.

Rolling Out the New Approach

With the FIFO and escalation queues in place, it was time to discuss the rollout of the new process. The team decided on a simple solution using a hard schedule date to start the new process. All tickets that were entered after the cutoff date would be placed in the FIFO queue and the entire team would start working from it. This is a suitable approach for new tickets, but another approach was needed for items that already existed in the system. The team discussed a number of options and came up with a simple rule: If a customer calls to discuss a current ticket in the system, then a new ticket would be created in the FIFO queue and linked to the original, and the new issue would be worked from the FIFO queue.

Tickets that existed in the system but were dormant for some time weren’t considered important enough to be fixed. This was obviously a controversial decision, but the team agreed to try it. A clear path was set and only a few questions concerning tracking and monitoring remained outstanding.

The team agreed that providing customers a planned release date for the fix was the primary goal of the new process. Everyone agreed that cycle time and lead time were the best indicators to determine which release the ticket would be deployed in. They also decided that a cumulative flow diagram (CFD) would provide all the information the team needed.

Spreading the Word

The next task at hand was to determine a lightweight and shareable way to capture and update CFD information, making it available across the organization. The simple spreadsheet shown in figure 1 was chosen. It contained a column for

Date	In Production	Release Ready	Released to QA	In Progress (Escalation)	In Progress (FIFO)	Blocked	To Be Worked
8/19/2010	19	2	14	1	11	10	50
8/20/2010	22	2	16	1	9	11	47
8/23/2010	23	5	24	2	7	13	47
8/24/2010	30	5	23	1	7	13	43
8/25/2010	33	11	15	2	9	13	50
8/26/2010	38	13	16	3	9	16	50
8/27/2010	38	14	18	3	7	15	52

Figure 1: Updating the numbers for the cumulative flow diagram

each workflow state, and the sustained engineering manager updated it daily with the latest ticket information immediately following the daily standup.

Using the CFD, they tracked the average cycle time of the tickets and, when that was combined with the number of items in the FIFO queue, were able to determine the lead time, shown in figure 2, for any new tickets added to the system.

Lead times were slightly inaccurate at first but started to

would be. This allowed them to plan around issues they encountered in the software.

Eventually, as the lead times continued to fall from a dismal 180 days down to a much more workable thirty days, customer service presented an idea to development that would reduce the escalation queue size down to two items. This was a remarkable improvement considering that early on in the process, customer service was facing enormous pressure to increase the escalations in the wake of customer demand. They knew the fewer items in the escalation queue, the more items from the FIFO queue could be worked on. As it turned out, the new process was showing an overall reduction in customer escalations from within the call center. This reduction was directly attributed to the increased predictability gained since implementing the FIFO process.

The predictability provided by eliminating prioritization

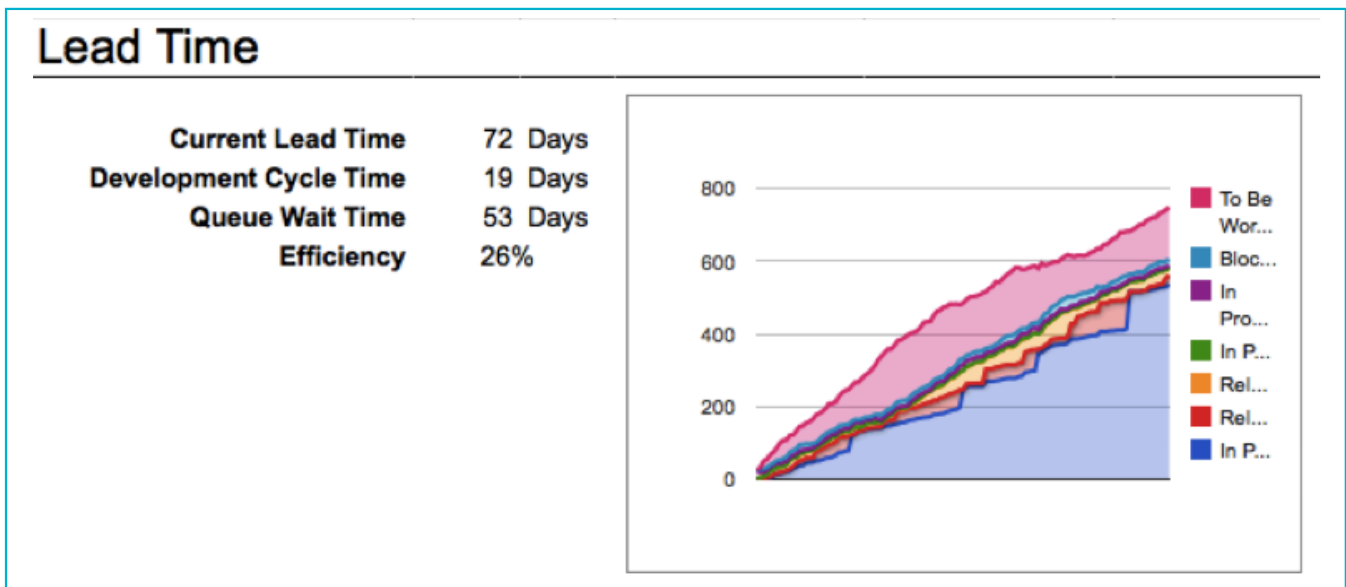


Figure 2: Customer service dashboard showing lead time

improve as more data was collected. In order to make communication of the current lead time easier on customer service, IT created a dashboard that provided the current expected release based on the latest CFD and queue information. Using the dashboard, customer service was able to provide an accurate release date to the customer at the time the customer called in, providing the highest level of service on first customer contact.

While things were better, they did encounter some objections from customers who were impacted. When customers were provided an expected release that was three months out, some became quite upset. The team had two choices: escalate the fix or keep the item in the FIFO queue. They chose the latter. By this time, the team realized the impact that escalating an issue had on the entire queue. Rather than optimizing attention for a single customer, the team decided to optimize the whole system. Most customers didn't necessarily want everything fixed as soon as possible—they just wanted to be able to count on issues being fixed when the organization said they

had a staggering effect on lead times and collaboration between customer service and IT. IT was providing the most up-to-date information to customer service on a daily basis, and customer service didn't feel as if all tickets were being sucked into a black hole, never to be seen again. Items placed in the escalation queue went to the sustained engineering manager and, with only three at a time allowed to be in progress, the manager was more informed about the details of each. IT also found that tracking the historical average cycle times per ticket was more accurate than the traditional estimation for forecasting lead times to the customers. With the predictability problem solved, the team could turn its attention toward reducing the lead times even more.

Tips and Techniques for Your Organization

If you would like to see how a limited prioritization system works for your team, here are some implementation tips.

Create a safe environment by creating a rollback plan:

Switching to using a FIFO queue for shared services requests rather than the more traditional model is quite a change in mindset. As with any large change, create an environment of safety surrounding it by making the change reversible. Determine what the expected outcomes are and how long you need to validate your hypothesis. In our example, the team decided that it could take up to three months to truly know if the change was successful. They had a plan to roll back the changes if the process had not shown improvements after the first month.

Create an escalation queue: Using the escalation queue with a work-in-progress (WIP) limit allows the team to prioritize some items without allowing a full prioritization of the entire backlog. In addition, the WIP limit was kept small enough that the primary FIFO queue was not starved and continued advancing forward. Your WIP limit for escalations should not exceed 50 percent of your total capacity; otherwise, it is not likely to work efficiently.

Treat the escalation queue as sacred: Items that are placed in this queue should be considered sacred. Once the team starts working on an item from this queue, it should be completed. No starting and stopping of the items is allowed. This helps ensure only customer emergency issues are placed in the queue. When combined with the WIP limit, it usually only takes once or twice for a lower priority item stopping work on a critical item for teams to realize the importance of using the escalation queue for its intended purpose.

Consider using class of service (CoS) for high-variability items: In the example, the items in the list were fairly similar in size and nature. In some shared services environments, this might not be the case. Some items could take a single hour, like a simple database change, while others could take a month to complete, like infrastructure provisioning. In these scenarios, a CoS model can be used, creating a FIFO queue and managing lead time for each CoS. This introduces some complexity with staff allocation, but using kanban can help balance the amount of time spent on each queue to achieve a better flow.

Keep your lead time information up to date and accessible: Nothing makes communication simpler than visibility of information and accurate data. Determine how frequently lead time numbers change, and keep this information updated. Updating too frequently can carry too much of a price, but consider updating the information at least weekly so that everyone who needs it has access to timely information. This keeps the expectations consistent with the current team performance.

Keep improving: Tailor your process to what works for you. Continually monitor and improve your process. Consider new metrics that could be tracked, but remember to keep it lightweight and strive for better than you were the day before.

In Summary

Are you stuck in the prioritization trap? Even though it seems counterintuitive, reducing your prioritization process to its bare essentials can improve both your predictability and value delivery when it comes to certain types of work streams such as shared services teams. **{end}**

bcarlso@leantechniques.co

WANTED! A FEW GREAT WRITERS!

Better Software magazine is always looking for authors interested in getting their thoughts published in a leading online magazine focused in the software development/IT industry. If you are interested in writing articles on one of the following topics, please contact me directly:

- Testing
- Agile methodology
- Project and people management
- Configuration management

I'm looking forward to hearing from you!

Ken Whitaker

Editor, *Better Software* magazine

kwhitaker@sqe.com

IS SCM TECHNOLOGY REGRESSING?

BY JOE FARAH



THINKSTOCKPHOTOS.COM

Free source code management tools are becoming more and more popular. But is free just as good as commercial offerings? Back in the 1970s, software configuration management (SCM) meant version control. Anything more than that was an in-house, advanced solution. This situation persisted through the 1980s. Two operating systems—Digital Equipment Corporation's VMS and Apollo Computer's Domain platform, a workstation variant of Unix—got into the act, providing various levels of version control. There were a few impressive proprietary, homegrown solutions, but they were largely invisible to the overall software industry.

Finally the 1990s arrived. Yes, there were still version control tools—RCS, CVS, and PVCS, to name a few—but there were also tools such as ClearCase, Continuous (which later became Synergy), STS (later CM+), MKS, and other evolving commercial tools. There was strong competition among the vendors. The UK research firm OVUM performed annual SCM tool product reviews that were highly respected and anticipated. Vendors had to improve their tools in order to stay competitive. Suddenly, the computing world was introduced to real SCM solutions, focusing on the broader software development lifecycle and on process and automation.

However, toward the end of the 1990s and into the new millennium, advances in SCM slowed and mergers took place, such as IBM's acquisition of Telelogic (Synergy) and Rational (ClearCase). The SCM industry continued to advance, but with reduced competition among vendors, the focus shifted more to lower administration costs than to investing in new features.

There was a move to glue together parts of a solution: requirements management, version control, change and configuration management, build control, test case management, document management, and even problem tracking.

Some integrated solutions knitted together two or three of these, others combined the tools into a comprehensive suite, and some provided many parts of the solution in a single integrated tool. With few exceptions, these solutions came with hefty price tags. The industry slowly adopted the term application lifecycle management (ALM), synonymous with computer hardware's product lifecycle management.

There were still new CM tool startups, including Accurev, and Microsoft's VSS—but, in my opinion, these were largely version control tools with a new twist here or there.

Subversion and Git

More recently, Microsoft's TFS and IBM's RTC have shown some real advances. But the software industry has embraced newer version control tools, with Subversion and Git topping the list. Why?

To put this in context, Git and Subversion, both open source version control tools, are battling it out for dominance in the SCM industry, and many organizations are regressing from stronger SCM solutions to more basic Subversion or Git. Some commercial SCM tool vendors have reacted to integrate their tools with these open source version control solutions.

Software teams need advanced SCM or ALM solutions with

real benefits that provide real productivity to all product team members. These benefits include fail-safe reliability and accessibility, near-zero administration, full change package support, a mature SCM process, easy process customization (rather than process buried in scripts), advanced user interfaces, reduced training requirements, comprehensive SCM metrics, generation of required SCM and release documents, data security, and navigation of traceability relationships.

With today's SCM technology, it's possible for users in each role to increase their productivity and for the entire product team to have all required SCM information at their fingertips. Good SCM tools should result in higher quality products with lower CM costs than basic version control tools.

Everyone Is Transitioning to Simpler SCM

I believe there are eight reasons SCM is reverting to version control through the use of Subversion and Git. But are these reasons really justified?

1. PRICE OF COMMERCIAL TOOLS

The reality—or perception—is that commercial tools are always expensive. ClearCase, the dominant commercial solution just a few years ago, carried a price tag ranging from thousands to tens of thousands of dollars per user. IBM's acquisition set the stage for costs holding fairly steady. And even though there are reasonably priced, highly capable commercial SCM tools, the perception is that the capabilities doesn't justify the added expense.

In addition, commercial SCM tools typically consume much customization effort, requiring consulting and training, although some of the new SCM and ALM tools have driven those costs down significantly through easy out-of-the-box installation and simpler configuration procedures.

However, building processes around free version control tools will cost significant resources. And from a training perspective, the biggest cost is in lost salaries. As a result, a mature process and easy-to-use technology are both needed to reduce training costs. These are somewhat lacking in open source solutions and in many commercial tools, too.

2. BENEFITS OF OPEN SOURCE TOOLS

Open source tools are inexpensive to acquire and maintain. The source code is in the public domain, so there is no chance of the vendor going bankrupt, and with so many contributors and experts out there, the tools should continue to receive support and enhancements.

However, it is difficult to make architectural changes to an open source product, as this is disruptive to the community and knowledge base. It also requires dedicated resources to see such changes through, which can take a significantly long time to develop.

Speed of updates can also be a problem due to a gradual response to market demand. In contrast, commercial providers pride themselves on rapid customer response for new features and capabilities.

3. INTEGRATION OF COMMERCIAL OFFERINGS BUNDLED ON TOP OF OPEN SOURCE TOOLS

In the commercial SCM and ALM market you'll find commercial vendors selling Subversion and Git solutions. Version control is a very visible component of an ALM solution, and when a vendor bundles one of them into its solution, that appeals to a market that has already adopted one or the other. On the other hand, a free tool bundled in a commercial package results in the perception of the loss of the free benefit.

4. STARTUPS WITHOUT THE EXPERIENCE OF FULL CONFIGURATION MANAGEMENT CAPABILITIES

Developers don't like administration, and if you put a group of unseasoned developers together, the last thing they want is to put some SCM administration in place. This is actually a selling feature for open source solutions. There is no need to contact vendors or evaluate solutions—just download what everyone else is using. You can find the minimum feature set you need right now.

Experienced developers, on the other hand, recognize the benefits of full ALM solutions. They know it's best to start out with all the capabilities at hand, especially if that solution increases developer productivity.

5. MARGINAL BENEFITS OF COMMERCIAL OFFERINGS

Just as the cost of some commercial tools can establish the perception that commercial tools are expensive, the functionality of some tools can paint the perception that commercial tools are only marginally better than open source counterparts. And in some cases, that's true. So why pay?

In my experience, there are several commercial tools out there that will pay for themselves within a few short months and then continue to accrue benefits. It doesn't take a lot of marginal benefit to cover the license costs of a commercial tool.

6. A LACK OF CUSTOMIZATION CAPABILITIES IN COMMERCIAL OFFERINGS

Open source version control tools have very limited customization capabilities, including scripts, triggers, and settings—perhaps sufficient, considering version control is a small part of the SCM and ALM puzzle. SCM and ALM tools, on the other hand, must support a greater variety of users, process, and data. Whereas version control needs may slightly differ between one organization and the next, this is not the case for SCM and ALM.

And while some commercial tools support large process variations that fit many projects, other offerings are much less configurable. SCM and ALM tools need to support significant customization and configuration, including the definition of metadata, tuning of the user interface for specific roles, defining the presentation and navigation of data, defining custom information links to to-do lists, and modification of process. In addition, the tool should provide documentation support, report and dashboard creation, and metrics required for a project.

With a high level of customization capability, each user can look at the complexity of SCM and ALM through views spe-

cific to his roles and requirements. The easier to customize, the more value the tool adds, resulting in increased productivity.

7. AN OVERALL POOR UNDERSTANDING AND POOR MARKETING OF THE TRUE BENEFITS OF FULL ALM

There are plenty of inexperienced team members out there. But they are going to remain inexperienced if the benefits of a full ALM solution are not easily and readily explained. The software SCM industry has not done a good job of educating the industry or marketing ALM.

Proper marketing of true benefits might take the form of annual tool competitions, where real-world SCM and ALM issues are addressed by all commercial tool suppliers, and even open source solutions.

SCM product reviews can help, but the complexity of SCM may preclude a thorough review and result in comparing only the basic common elements of each tool. You cannot compare an open source tool such as Git to an advanced, modern SCM and ALM tool. It would be like comparing a bicycle to an automobile.

8. THE PERCEPTION THAT BUILDING AROUND OPEN SOURCE TOOLS IS EASIER TO SELL TO MANAGEMENT THAN CAPITAL EXPENDITURES OF ALM TOOLS

"How much does the tool cost?" is usually the first question. And if the answer is that it is free because it's an open source tool, then the response a software manager will most likely give is "Great! No cost? Go for it!" However, a decision like this would never pass a business case review. The cost of licenses is not the largest cost of SCM. Training, process implementation, scalability, and integration with existing systems can be very costly.

Every company needs an ALM solution. How much of that is manual or done piecemeal is a separate question, but the cost of ALM is the cost that has to be measured in a business case.

Free version control, no matter how good, is not an ALM solution. Solutions may be built around it and engineered for cost-effectiveness, but it's even better to have a version control component specific to a full ALM solution. Then certain things become more obvious: You don't check in files; you only check in changes. You don't just type in comments; you reference and link to approved problems and feature activities.

Advance!

SCM technology appears to be regressing. It's customer-driven. But look again and perhaps you'll see that this shouldn't be the case. The software industry, in its adoption of version control instead of SCM, is charting a course that is not much different from what we witnessed back in the 1980s.

What do you do to fill in the holes in your SCM? Or is version control your only third-party tool? How do you justify the cost of maintaining your own scripts to support the version control tool you use? Are you considering changes to how you perform SCM or version control? Take a good look at what's available out there. It's time to advance! **{end}**

farah@neuma.com

Featuring fresh news and insightful stories about topics that are important to you, TechWell.com is the place to go for what is happening in the software industry today. TechWell's passionate industry professionals curate new stories every weekday to keep you up to date on the latest in development, testing, business analysis, project management, agile, DevOps, and more. Here is a sample of some of the great content you'll find. Visit TechWell.com for the full stories and more!

Four Reasons to Stick with Daily Scrum Meetings

by Cameron Philipp-Edmonds

A scrum meeting, or daily scrum, should last no longer than fifteen minutes and should happen every day at the same time and place during a project's duration. The daily scrum should communicate project progress, clarify product direction, and identify any issues or impediments. Then, those factors are all taken into account by the ScrumMaster and product owner, who remove barriers to keep the project on course.

Although not everyone speaks at these meetings, every team member is required to attend. Pro tip: The ScrumMaster and product owner are also team members, despite their titles of leadership. If you have come to a position in your career where you feel you don't need to attend every daily scrum, then consider these four reasons why you should stick with it.

Continue reading at <https://well.tc/GYF>

The Tech Industry's Problem with Ageism

by Steve Berczuk

A hallmark of many tech companies, particularly those practicing agile, is being a flat organization with a company culture based on a meritocracy. When hiring, however, this meritocracy is inconsistent with the importance some companies place on a person's age, as I heard on an NPR interview with New Republic senior editor Noam Scheiber.

In the interview, Scheiber says that an engineer is considered old at thirty-five and being over forty is too old to be an entrepreneur in the tech industry. The surface reason for this bias seems to be that age is a proxy for being out of date and seemingly less able to come up with new ideas.

Continue reading at <https://well.tc/GmS>

Avoid These Sneaky Time Wasters at Work

by Naomi Karten

Ask people what the biggest time wasters are that they face at work, and the top three answers are usually endless email, meandering meetings, and Facebookery. Respondents in one survey reported that checking email wastes 50 percent of the workday. And then there's surfing the web, dealing with interruptions, and looking for the piece of paper that's lost somewhere on your desk.

Of course, it's tough to put in a full day of nonstop work. The typical employee works only about six hours of every eight-hour day. The remaining time is taken up with nonwork activities such as breaks and chatting with coworkers.

Continue reading at <https://well.tc/Gs6>

Five Software Testing Myths Busted

by Ulf Eriksson

Testing may not be the most glamorous job in the software world, but it plays an essential role in creating functional software.

This article busts some myths that have been perpetuated about testing and shows how testing is not only very useful but also something worth being passionate about.

Continue reading at <https://well.tc/GWD>

Should You Be Worried about Shadow IT?

by Joe Townsend

If there is one thing IT professionals—especially those involved with security—like, it is control: control of software, hardware, firewalls, and any other thing attached physically or electronically to the network. However, we all know that shadow IT exists, which means IT professionals don't have complete control like they used to. What we don't agree on is the danger—or lack thereof—it creates. Let's try to find some answers.

A document on McAfee.com offers a deep dive into why shadow IT has become so popular and prevalent. From the “everyone does it” argument to the lack of SaaS policies and employees wanting to get the job done, the authors expose shadow IT and the dangers it poses to your organization. So, is shadow IT always a bad thing? Remember, bring your own device (BYOD) is another form of shadow IT, and companies are embracing this phenomenon.

Continue reading at <https://well.tc/GWV>

Strong Competition in Cloud Computing Means the End User Wins

by Mukesh Sharma

Cloud computing is just about a decade old. Despite its being a relatively new entrant to the world of computer networking, its impact across disciplines and the evolution it has had are tremendous.

We often look at whole organizations that have helped revolutionize and embrace new technology, but it is equally important to look at specific individuals who were behind the scenes and the early adopters who together helped the technology reach unprecedented heights.

Along these lines, cloud computing's pioneers are not just the masterminds at Amazon and Google but also people from companies like Salesforce, who gave needed facelifts to varied manifestations such as software as a service and infrastructure as a service.

Continue reading at <https://well.tc/GWB>

DEVELOPING CUSTOM APPS for the CLOUD

by Pete Morano and Matt Stratton

From the developer's point of view, developing custom applications in the cloud offers a great deal more freedom to be creative and forward-thinking because using a cloud platform significantly reduces development time and costs. The market has already responded with cloud-based tools for sales, marketing, finance, and even DevOps—all of which take advantage of the cloud's appealing attributes: instant access, no downloads or maintenance, and pay-as-you-go pricing models. Now, new cloud-based development tools are quickly arriving to drive the cloud development platforms to new heights.

Removing Hidden Development Costs

To understand what a cloud-hosted development environment brings to the table, we have to consider the reduction of development time and cost. An often overlooked expense in any development lifecycle is the time and effort it takes to set up, configure, and maintain development tools. As the development team grows and new members are brought on board, the process repeats itself and is seldom automated. Some teams address this challenge by creating disk images of developer workstations, but this approach often requires that the same hardware profile be provisioned for each developer. Likewise, maintaining the images becomes a task in its own right and, therefore, is often neglected over time, resulting in stale images.

The cost and complexity of maintaining a noncloud development environment further increases when you consider that many developers support multiple project environments, often running different versions of the same integrated development environment (IDE), as well as different versions of software development kits and libraries. At some point, a developer's local environment can become too cluttered and fall out of sync with the supported production environment, creating significant difficulties in tracking down bugs and cross-environment discrepancies in application behavior.

Another complexity of managing desktop development environments is the support and maintenance of the growing array of plug-ins and extensions that integrate with services that are key to the development lifecycle. These include source code version control, publishing and deployment services, and the ever-expanding collection of developer productivity tools.

Broad Options

Options for developing for the cloud range from hosted IDEs to cloud-based virtual machines running common operating systems such as Windows or Linux.

Some of the more popular hosted IDE offerings (Cloud9, CodeAnywhere, and CloudIDE) effectively turn your browser into a thin-client development environment while still offering many of the features one would expect from a desktop-based development environment:

- Code completion
- Integration with version control systems
- Ease of deployment to a wide variety of server environments

One of the immediate benefits of developing within an IDE is the ease of collaboration with other developers. Pair programming, code reviews, and real-time interaction are all possible without the overhead of traditional screen-sharing tools.

Another option is using cloud-based virtual machines to host your development environment. Going beyond using your browser as the shell for your IDE, these are fully provisioned machine instances accessible through remote desktop tools. Amazon and Microsoft offer the ability to quickly and cost-effectively spin up virtual machines with their EC2 and Azure offerings.

Microsoft has recently made cloud-based development even more cost-effective by offering prebuilt images of common developer environments. For example, in a matter of minutes, MSDN subscribers can create a virtual machine that is pre-configured with Visual Studio Ultimate 2013, Windows Azure SDK for .NET 2.2, SQL Server 2013 Express, and SharePoint 2013 Trial.

Usage-Based Costs

The costs of cloud-based development environments vary. Cloud-hosted IDEs can run as little as fifteen dollars per month for multiple workspaces, and many offer free versions with limitations of some features. At the other end of the spectrum, machine instances in EC2 or Azure can run several hundred dollars per month, depending on usage and technical specifications.

In addition, Microsoft recently changed its pricing model to support by-the-minute pricing with no additional charge for stopped instances. In the past, developers were charged for VM usage even if they turned the VM off or stopped, making VMs cost-prohibitive for development.

Why Is Cloud Development for You?

Developing in the cloud offers amazing flexibility. Cloud development may not be the answer in every case, but if you are building software that will run in the cloud, it makes sense to build and test it there as well. By offloading much of the effort spent maintaining code on one or more development environments, developers can execute more efficiently, resulting in lower costs for clients and more time for developing and releasing solutions. Freed from unnecessary busy work and concern about cost overruns, cloud developers can spend more effort on the actual development process, leading to more robust solutions.

The Relationship between DevOps and Cloud Development

Another advantage of developing in the cloud is that when working in a DevOps environment, developers can focus on their specialties rather than devoting time to system administration, giving the organization a competitive advantage in the marketplace. In a DevOps culture, product delivery teams are cross-functional; silos are broken down and everyone on the team is focused on delivering the product. By utilizing cloud technologies, these silos are further minimized because every-

one's focus is on the product.

In a traditional operational organization, Ops focuses on stability and development focuses on new features. In a DevOps world, both groups have the common goal of delivering first-class software quickly. By removing the need for the core infrastructure using a cloud development platform, the operational teams have fewer competing priorities. In this model, the needs of the software product drive changes to the operations and infrastructure, not vice versa.

One of the key tenets of DevOps, like in agile, is to shorten feedback loops. The faster an idea can get in front of a customer or other feedback provider, the sooner you'll know if the idea meets customer needs. A truly agile organization is one that is willing to fail small, fail fast. The longer it takes to deploy software for the customer to view, the slower it takes to fail.

The cloud allows developers to experiment quickly. It reduces friction and impediments to agility by giving developers the chance to test an idea without having to invest time and capital in building infrastructure. Treating computing as a utility creates an as-needed consumption model: on demand and just in time.

Cloud technologies like platform as a service democratize application deployment and management. By removing things such as patching, OS configuration, and other "mystical" IT items that require extensive system administration expertise, a software organization can focus on its core differentiators. For example, it probably isn't a competitive advantage to excel at patching Windows servers, but it is a competitive advantage to

write a great search algorithm for your data. The more a team can focus on the things that make a business or product special, the more efficient the delivery process is. This helps achieve the level of velocity necessary to shorten feedback loops.

Cloud technologies also facilitate modern software release automation. Manual software release is error-prone and slow. The more software deployment is automated, the more velocity and agility increase. Traditional infrastructure methods and implementations require more effort to hack the automation to work with legacy, noncloud systems, but modern cloud technologies provide support for these tools and practices natively.

In a high-velocity software delivery organization, applications can be approached as services rather than groups of systems, more closely aligning the technology drivers with the business objectives. Modern software infrastructure should approach servers and systems as merely components in the overall service delivery. If servers are considered similar to processes or instances rather than actual physical hardware, we can treat them in an immutable, disposable way. Rather than installing new releases of software to existing boxes, it is becoming increasingly apparent that simply releasing new systems with the updated code is more robust, stable, and manageable. Without the automation and on-demand capabilities that the cloud provides, operating this type of immutable infrastructure becomes substantially more challenging. As an organization embraces the DevOps culture, cloud technologies become essential. **{end}**

pmorano@10thmagnitude.com
mstratton@10thmagnitude.com



ATTACKING QUALITY ISSUES IN DATA WAREHOUSING

BY WAYNE YADOW

Implementation and growth of data warehouses continue to gain attraction as organizations become more aware of the benefits of decision and analytic-oriented databases. Nevertheless, there is often one important obstacle to the rapid development of commercial data warehouses: data quality. Serious problems are often discovered when planning and populating a warehouse that, if not resolved, can delay or eventually result in terminating the project.

During the past twenty years, researchers have contributed to the understanding of data quality issues, yet little research has been collectively compiled to identify root causes of data quality problems that occur throughout major phases of data warehousing.

Based on my experience, the following are primary causes of data quality defects in data warehousing [1]:

- Flaws in the data warehouse modeling and schema design
- Defects in data sources used as input to the data warehouse
- Failure to effectively profile source and target data
- Weaknesses in the design and implementation process for data warehouse staging and extract, transform, and load processes

Using Early-Phase Defect Prevention Methods

This article highlights the reasons for data deficiencies related to the root causes listed above together with timely quality assurance efforts that can be implemented for discovery and correction. It is hoped that data warehouse designers, developers, and testers cooperate and benefit by examining these quality issues before moving forward with data integration into the data warehouse.

Figure 1 displays a high-level view of the common data

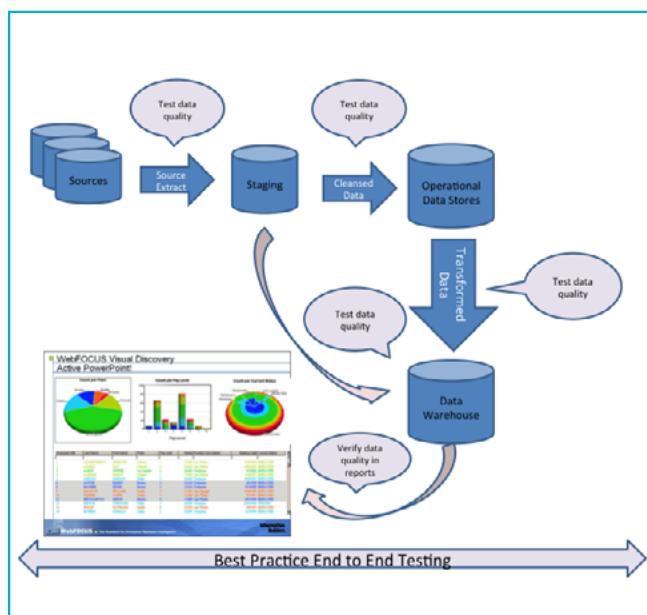


Figure 1: Data warehouse flow and recommended phases for data quality testing

warehousing extract, transform, and load process where data quality and functional testing are recommended.

Data Quality Issues Related to Data Model and Schema Design

Design of the data model for the data warehouse greatly influences the quality of the analysis by programs that use the data. A flawed schema will negatively impact information quality.

Data modeling is the process used to define and analyze data requirements needed to support business processes within the scope of application needs. The data modeling process should involve trained data modelers working closely with business stakeholders, developers, quality assurance, and potential users of the information system. Data modeling defines not only data elements, but also their structures and the relationships between them.

Data modeling methodologies should be used to model data in a standard, consistent, and predictable manner in order to manage the data as a resource. The use of modern data modeling standards and tools is strongly recommended for all projects.

Table 1 shows highlights of how a proper data warehouse design review can make or break your data warehouse. [2]

Quality Issues in the Data Warehouse Source Data

A leading cause of data warehousing and business intelligence project failures is finding and then loading incorrect or poor-quality data. The source system often consists of transaction and production raw data, which is where the details are pulled from and made suitable for the data warehouse. Each of these data sources usually has its own diverse methods of storing data, which may contribute to data quality problems if proper care is not taken.

Data warehouse environments provide the source of information used by business units to make strategic decisions. However, much of that data is created outside the warehouse. That means data quality problems can originate at the source and can therefore persist due to faulty data acquisition and delivery processes, or interpretation and transformation glitches.

Data quality problems in source systems need to be recognized as requiring mitigation. This can be accomplished by either addressing these problems as defects or by getting approval by stakeholders that these issues are acceptable. The QA team must then ensure that data warehouse users are aware of these data quality deficiencies in cases where they are not fixed before being loaded into the data warehouse.

Under certain conditions, source files are the product of multiple file consolidations. Consolidated files can, in turn, result in data quality being compromised before being loaded into the data warehouse staging area. Table 2 summarizes a few other possible causes of data quality issues as data sources are staged into the warehouse.

Other reasons for data pollution issues in the data warehouse may be cases where data was never being fully captured by source systems, the use of heterogeneous system integra-

	Questions during data warehouse modeling and schema design	Impacts and data quality risk mitigation issues
1	<p>Have the major data warehouse dimensions been broken down into lower levels of detail?</p> <ul style="list-style-type: none"> • Have the keys been identified? • Have the attributes been identified? • Have the keys and attributes been grouped together? • Have the relationships between groupings of data been identified? • Have the time variances of each group been identified? 	<p>There needs to be a data model that serves as the intellectual heart of the data warehouse environment. The data model normally has three levels: a high-level model where entities and relationships are identified; a mid-level model where keys, attributes, and relationships are identified; and a low-level model where detailed database design can be accomplished. While not all of the data needs to be modeled down to the lowest detail in order for the data warehouse environment to be built, at least the high-level model must be complete.</p>
2	<p>Have levels of data granularity been defined? A high level? Low level? Multiple levels?</p> <ul style="list-style-type: none"> • Will rolling summarization be done? • Will there be a level of true archival data? 	<p>An important design issue in the data warehouse environment is that of granularity of data and the possibility of multiple levels of granularity.</p>

Table 1: A small sample of a data modeling and schema development checklist

	QA checks for source data quality problems	Impacts and risk mitigation efforts
1	<p>Insufficient data source selection and profiling of candidate warehouse data may cause data quality problems (i.e., source data that does not comply to business rules).</p> <ul style="list-style-type: none"> • An age field contains values greater than 150 years • ZIP codes contain non-numeric values for U.S. addresses 	<p>For each required data source field, testers should run queries or profiling tools to verify that the data complies with business rules.</p>
2	<p>Data warehouse business analysts may possess insufficient knowledge of interdependencies among data sources that are used to populate the warehouse.</p>	<p>Review the quality of source data entry processes, loads, and merges used to create each data source. This effort may also include the establishment and monitoring of service level agreements, communication protocols with data suppliers, and data quality assurance policies.</p>

Table 2: Checking for source data quality issues

tions, and a failure to have an adequate policy for data warehouse project planning.

Discovering Data Quality Issues Using Data Profiling Techniques

When potential data sources are identified and then finalized and agreed to, data profiling should be implemented immediately on that source data. Data profiling is the examination and assessment of your source systems' data quality, integrity, and consistency—sometimes known as source systems analysis. [3]

As important as data profiling is, it is often ignored and, as a result, data warehouse quality can be significantly compromised.

The data quality assurance analyst supports an organization's data quality initiatives by analyzing data. Profiling is the primary method for performing a data quality assessment. Data profiling is also used to quantify the extent of problems

surfaced by other means and to measure the impact that data quality remedies have had.

Listed below are examples of problems that are easily uncovered through data profiling:

- Data fields used for purposes other than expected
- Fields that contain no data for any record
- Missing values when a field is defined as NOT NULL
- Violations of business rules

Business analysts can determine problem root causes during data analysis that could result in a substantial number of data quality problems that need to be corrected.

At the beginning of a data warehouse project and as soon as potential data sources are identified, data profiling assessments should be conducted to prepare for a go/no-go decision about proceeding with the project.

Table 3 depicts just a few of the possible causes of data

	Data quality issues discovered through data profiling	Impacts and risk mitigation efforts
1	Counts and types of distinct values in each field are not as expected.	Analyzing the number of distinct values within each field will help identify possible unique keys in the source data, which are referred to as natural keys. Identification of natural keys is a fundamental requirement for database and extract, transform, and load architecture, particularly when processing inserts and updates.
2	Counts of zero, blank, and NULL values in each field may reveal defects.	Analyzing each field for missing or unknown data helps you identify potential data issues. This information aids database and extract, transform, and load architects to set up appropriate default values or to allow NULLs on the target database fields where an unknown data element is unacceptable.

Table 3: Discovering data quality issues through data profiling

	Data quality problems discovered during extract, transform, and load phases	Impacts and early risk mitigation efforts
1	Inadequate source-to-target data mapping preparation.	Data mapping documents should be developed and continually maintained throughout the project. Tools designed specifically for data mapping should be considered over the use of manually created spreadsheets.
2	Improper extraction of data to the required target fields.	Design of extract, transform, and load program logic should be reviewed before development has been completed to assure that data will be loaded correctly to each field in target tables.
3	Failure to generate data flow and data lineage documentation that depicts the extract, transform, and load process	Data integration in warehouses is highly complex for most projects. Assure that modern data flow and data model diagrams are used to illustrate the planned flows.

Table 4: Checks for issues in the extract, transform, and load process

quality degradation discovered at the profiling stage of data warehousing.

Data Quality Issues Discovered During Data Loading

An important design consideration is whether data cleansing should be conducted for each source input during the staging phase, during the extract, transform, and load process, or within the data warehouse. The data staging area is where “grooming” is often conducted on data after it is loaded from source systems. [4]

Data staging and the extract, transform, and load phases are considered to be the most crucial phases of data warehousing, where maximum responsibility for data quality efforts occurs. These are prime phases for validating data quality from sources or auditing and discovering data issues. There may be several reasons for data quality problems during the staging and extract, transform, and load phases. A few of those are listed in table 4.

When data quality problems are encountered while importing data into the data warehouse, there are four viable actions that can be taken: exclude the data, accept the data, correct the data, or insert a default value. These are some of the design decisions that must be faced while working to improve data quality in early phases of data warehouse projects.

In Summary

There are many causes of data quality problems that may be found throughout all phases of data warehouse development. Data quality issues have been classified and described in a way that should help data warehouse practitioners, implementers, and tool providers find and resolve these issues as they move forward with each phase of data warehousing. **{end}**

wyaddow@aol.com



Click here to read more at StickyMinds.com.

■ References

Quantitative Software Management Inc. Announces SLIM Suite 8.2

Quantitative Software Management, Inc. (QSM), a software process improvement and systems development estimation company, announced SLIM Suite 8.2, which provides the ability to perform enhanced top-down estimation for capacity planning.

Unlike other resource-demanding management tools that rely on bottom-up estimates, QSM is the first in the industry to provide detailed resource breakdowns, utilizing a more accurate top-down approach. Top-down estimation accounts for even the unpredictable aspects of IT project implementation that a bottom-up approach does not, such as unrealistic project goals, miscommunication among team members, and rework, which may account for up to 60 percent of the total effort on a project.

<http://www.qsm.com>

Cloud Technology Partners Announces General Availability of PaaS Lane

Cloud Technology Partners, a cloud solutions company, announced the general availability of PaaS Lane, the first software solution that analyzes Java and .NET application source code to uncover problems that could impact cloud readiness and accelerates the migration of applications to public or private clouds. PaaS Lane assesses existing Java and .NET apps quickly, finding coding errors, hardcoded dependencies, security issues, scalability problems, and other factors, which could quickly derail an application migration project. By reducing the need for manual code reviews, which are labor intensive and time consuming, PaaS Lane can shorten migration times by months and eliminate hundreds of thousands of dollars in cost.

PaaS Lane includes complete support for Java and C# .NET and assesses cloud readiness for all major private and public cloud platforms. Building on support for Amazon Web Services (AWS), PaaS Lane 2.0 includes support for other leading cloud platforms, including Google Compute Engine, Microsoft Azure, Pivotal CloudFoundry, and Apprenda. Additional new features include: a redesigned reporting interface, live drill-down into alert detail, interactive charting, and new options for delivering reports in PDF and Excel formats.

<http://www.paaslane.com>

CollabNet's CloudForge Development Platform Available in Cloud Foundry Marketplace

CollabNet, a provider of cloud-based application lifecycle management (ALM) solutions for agile software delivery at scale, announced that its CloudForge development platform is currently available as an add-on service in Pivotal's on-demand PaaS service, powered by Cloud Foundry.

CloudForge availability in the Pivotal Web Services add-on marketplace provides developers with an integrated development and PaaS offering. It will accelerate development and deployment of enterprise applications to any Cloud Foundry-

based platform, whether it is hosted in a private or public cloud. CollabNet's CloudForge helps developers build websites, mobile/cloud/web applications, and rapidly prototype and deploy business software. Built for the enterprise, it offers a fast and practical path to grow and scale organization-wide.

<http://www.collab.net>

Atlassian Releases Atlassian Connect

Atlassian, a provider of collaboration software for teams, released Atlassian Connect, an extensible new framework for developers to build add-ons that deeply integrate with Atlassian JIRA OnDemand and Atlassian Confluence OnDemand, the company's cloud-based issue tracking and collaboration offerings. Developers can offer these add-ons to Atlassian's 33,000 customers via the Atlassian Marketplace, a proven ecosystem of more than 1,500 commercial and free add-ons.

The new Atlassian Connect framework, built on common web standards like REST APIs and JWT authentication, offers additional modules for developers to integrate add-ons directly into Atlassian applications using the technology stack of their choice. Third-party partners like Zendesk, Lucidchart, Giffy, Zephyr, and Comalatech are among the early adopters of Atlassian Connect.

<http://www.atlassian.com>

Parasoft Unveils Latest Release of API Testing Solution

Parasoft unveils latest release of its API testing solution, which introduces enhanced support for testing the RESTful APIs that have become the backbone of mobile transactions.

The latest release of Parasoft's enterprise-grade API Testing solution focuses on ensuring that rich regression suites for JSON can be more rapidly constructed, continuously executed, and effectively managed. By providing increased control over assertion management and test data management, Parasoft eliminates the need for complex scripting and allows less technical resources to test business-critical transactions.

<http://www.parasoft.com>

Zeenyx Software Launches AscentialTest Version 6.5

Zeenyx Software, a testing solutions company, announced the release of AscentialTest version 6.5, which features an automatic data table generator. AscentialTest allows domain experts to design and create manual and automated tests based on reusable steps that are built by point and click, as actions and data objects are automatically generated by interacting with smart images called snapshots.

V6.5 completes the new step-based testing paradigm. The test editor automatically generates data tables with fields defined in the data type expected by the test and binds them to test parameters.

V6.5 also includes testing support for 64 bit applications and for Siebel, PowerBuilder, and Dev Express.

<http://www.zeenyx.com>

FAQ

expert answers to
frequently asked
questions

My Team Is Agile, but My Organization Is Not! What Can I Do?

Has your team adopted agile, but you are still struggling to deliver customer value because of organizational inertia? Your organization's portfolio management probably needs upgrading. Here's how to spread the agile love beyond your team by addressing inefficient, outdated, and value-killing portfolio management processes.

1. **Terminate zombie projects.** Zombie projects are ongoing, wandering projects that continually miss deadlines, lower morale, and simply cost too much. Don't vacillate! Terminate these zombie projects immediately. Purge your portfolio to reduce project inventory and redirect the effort of team members to more valuable initiatives. This is the best way to speed overall throughput, reduce waste, and maximize value.
2. **Create stable teams.** Studies show that teams work best by focusing on one project at a time. So, create stable teams with team members from cross-functional departments, including business analysts, designers, developers, testers, and a project manager (or ScrumMaster). Dedicate core team members at least 80 percent to the project.
3. **Break large projects into small increments.** A minimum marketable feature (MMF) is a key component of marketable value. Group product features into increments of MMFs to deliver early value to end-users. Break up large projects into smaller projects organized around MMFs to reduce work in process and, in turn, reduce lead time.
4. **Stop starting, start finishing.** Most organizations start more projects than they finish. To manage the project on-ramp, create and follow a lightweight, disciplined project prioritization process to decide which projects are started. Then, start only those projects that can be properly resourced. If a project starts to falter, terminate it before it becomes a zombie project.
5. **Create a portfolio backlog.** Prioritize your projects based on business value to create a portfolio backlog. Now, have your teams pull the highest priority project from this backlog. Focus on a single project at a time and work closely with the business sponsor to deliver it. After the team completes the project and delivers the system into production, pull the next highest priority project from the backlog.
6. **Track and control the flow of in-flight projects.** Institute a portfolio prioritization and control process that is based on the delivery of business value first and use it to relentlessly re-evaluate projects in flight. Keep work in process for in-flight projects to a minimum. Don't use just the triple constraints—scope, schedule, and resources—to measure progress. Instead, track delivery of MMFs in increments by milestone to measure the achievement of product goals. Conduct quarterly inspections, and if a project falters, consider either terminating it or breaking it down into smaller increments.

Use the steps above to make your portfolio flow and thus lay the foundation for lower variability, faster throughput, and higher business value.

Good luck spreading the agile love to not just your team but your entire organization! **{end}**

by Sanjiv Augustine
sanjiv.augustine@lithespeed.com

Why Am I Always Getting Bad News in the Eleventh Hour?

The needs of the business don't often match the expectations of the team. With the following narratives, who's going to come out on top?

by **Kenton Bohn and Ryan McClish** | kenton.bohn@geneca.com, ryan.mcclish@geneca.com

Industry data states that 50 percent to 80 percent of software projects fail to satisfy predetermined business objectives, budgets, or timelines. Spend some time reading through the many surveys that have been done and you'll realize that surprisingly few of these projects fail from technical challenges.

This is especially troubling given the ever-increasing role of software in the enterprise.

Very smart people are planning and executing these projects. So why are they struggling? Why is it that when business people and technologists collaborate, bad things seem to happen? Are we speaking different languages? Are we trying to accomplish different things?

Are IT people and business people from different planets?

Sound Familiar?

Kenton, a product manager and business partner, and Ryan, team lead, have been partnering on a project that Kenton believes will be very valuable to the business. Kenton is getting pressure to have the product ready for an upcoming trade show.

Months before when the two colleagues met, Kenton described the scope of the project. Ryan needed more time with Kenton to detail requirements, but other priorities were always getting in the way. Because Ryan was so familiar with the business, he felt comfortable making assumptions to build a plan and a budget—and Kenton trusted him.

Although some of the unknowns that had been identified early on started to cause delays, Ryan remained fairly confident that his team could make up the time. Everything appeared to be going well, until ...

I'm Right and You're Wrong

Ryan: Hi, Kenton, do you have a minute? I need to talk to you. You'll want to sit down. I have some bad news. We're going to be late.

Kenton: What happened? I thought we were on track to finish on time.

Ryan: We thought we could complete testing in two weeks, but now it looks like it's going to take six instead.

Kenton: Ryan, this isn't acceptable. I told my boss and the executive team that the product would be ready for the October trade show based on your project plan and completion date.

Ryan: Wait a minute. We agreed that we would try to hit the October date, but we also knew that there were risks. And we've run into some unexpected bugs and complexity.

Kenton: You really put me in a tough position this time. I can't believe you did this.

Ryan: I did this? You committed everyone to this before even talking to me. If you had just worked with me to put together a real plan from the beginning, we would have both known there was no way to get this done in that time frame.

Working Toward Success

Kenton: All right, let's remember we are both on the same team here. I realize I made external commitments on your team's behalf without talking to you first.

Ryan: OK, so what do we do now?

Kenton: First, we need to figure out what it's going to take to get this thing done. Let's not worry about the trade show right now. We need a real plan that we can both commit to. Then, we can see if we can make any adjustments to bring in the timeline.

Ryan: OK, I will work with my team to build the new plan—but we need more of your participation this time around. Are there some things you would be willing to compromise on to help bring in the time line?

Kenton: Well, I guess we don't need that mobile app yet.

“Lack of alignment and misplaced ownership often takes a project in the wrong direction or motivates people to say one thing and do another.”

Would it help if we focused our time on the website?

Ryan: Yes, that would help a lot. Then I can use my mobile testing team and focus them on the site with the rest of the team. I'll put all hands on deck for the development team, and we should be able to get the website done much sooner.

Kenton: Would you be able to deliver that for the trade show?

Ryan: Let me talk to the team and make sure that's something they would be able to commit to.

Kenton: Sounds like a plan.

The Last Word

When projects fail, it is rarely because of the complexity of technology. Most often, it is because of the dynamics between business and IT. Things like misplaced ownership, distrust, misalignment, or a lack of tools are commonly in play but often overlooked as the underlying root causes of the failure. What can we do to increase our chances for success?

We often let our pride and our relationships with people get in the way of progress. We need to take a step back and figure out how to get out of trouble and move forward with a solution. Ask yourself, "What is going on with me personally? What is going on between my colleagues and me that is causing me to feel threatened and defensive?" This is something that

needs to happen on both sides of the organization.

Start to think about what guiding principles are missing or being violated that contributed to this situation. Ask yourself, "Did we define success up front, and do we have a way to communicate that to each other in a language that everyone understands? Do we have the right tools to get the job done? Does the organization empower people to make commitments and hold them accountable to keeping those commitments?"

Oftentimes, we start projects without a clear understanding of what success is for the project. We have commitments, but they were handed to us. Lack of alignment and misplaced ownership often takes a project in the wrong direction or motivates people to say one thing and do another.

Lastly, but possibly most importantly: What is the culture of the organization? Does the culture support the value "We succeed together; we fail together"? Ask yourself, "Do we promote people taking ownership?" "Do business and IT work together or against each other?"

The next time you find yourself in that difficult situation where the project is not going as planned, take a step back and ask yourself some questions. You might find out that some small changes in how you work with your team might make all the difference. **{end}**

index to advertisers

Agile Development & Better Software Conference West	http://adc-bsc-west.techwell.com	9-12
ASTQB	http://www.astqb.org	8
Capgemini	http://www.hp.com/discover	25
Ranorex	http://www.ranorex.com/whyBSM	2
StickyMinds	http://www.stickyminds.com	24
SQE Training	http://www.sqetraining.com	1
STARWEST	http://starwest.techwell.com	Inside Front Cover
Virtusa	http://bit.ly/1ePFw03	Back Cover

Display Advertising
advertisingsales@sqe.com

All Other Inquiries
info@bettersoftware.com

Better Software (ISSN: 1553-1929) is published six times per year: January/February, March/April, May/June, July/August, September/October, and November/December. Back issues may be purchased for \$15 per issue plus shipping (subject to availability). Entire contents © 2014 by Software Quality Engineering (340 Corporate Way, Suite 300, Orange Park, FL 32073), unless otherwise noted on specific articles. The opinions expressed within the articles and contents herein do not necessarily express those of the publisher (Software Quality Engineering). All rights reserved. No material in this publication may be reproduced in any form without permission. Reprints of individual articles available. Call 904.278.0524 for details.

Virtusa QA Revolution

The future of quality is here

Social and millennial QA, 'connected quality' and managed testing services driving innovative quality outcomes



Virtusa is proud to partner with Per Scholas to build US based QA resources leveraging their Software Testing Education Program (STEP) in underserved communities