precision content

Looking Beyond Agile: Structured Content Development and the Iterative and Incremental Development Methodology

Keith Schengili-Roberts ● March 3, 2021

Agenda

precision content

1. Introduction
2. Business Reasons for Using Agile within Technical Documentation
3. The Reasons Why DITA Works So Well with Agile
4. Intro to the Iterative and Incremental Development Methodology
5. Q/A

Keith Schengili-Roberts

Who's This Guy?

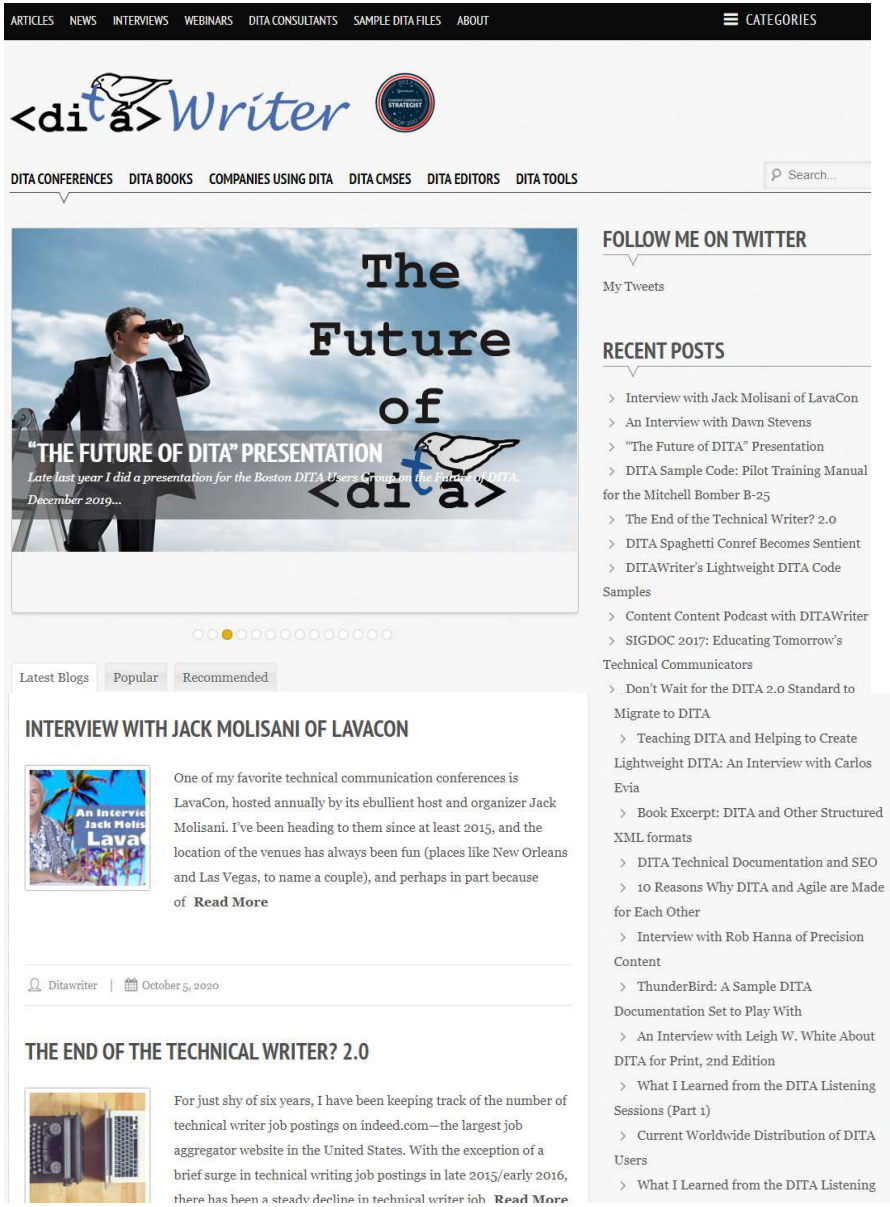
precision content

What I do ?

- Senior Content Strategist with Precision Content
- Liaison with OASIS; Chair of DITA Adoption Committee and member of the DITA Technical Committee and LwDITA Sub-committee
- Also a occasional professor on Information Architecture at the University of Toronto's Faculty of Information
- Have 14+ years of experience with DITA XML

Am Also “DITAWriter”

- Industry blog started +10 years ago
- Over 385,000 views (not bad for a niche website)
- Have regularly updated info on DITA Conferences, DITA Books, Companies Using DITA, DITA CMSes, DITA Editors, other DITA Tools, and DITA Consulting Firms
- News and views on DITA use
- Also features interviews with those making a difference in the world of DITA



precision content

Another Thing People Might Not Know About Me…

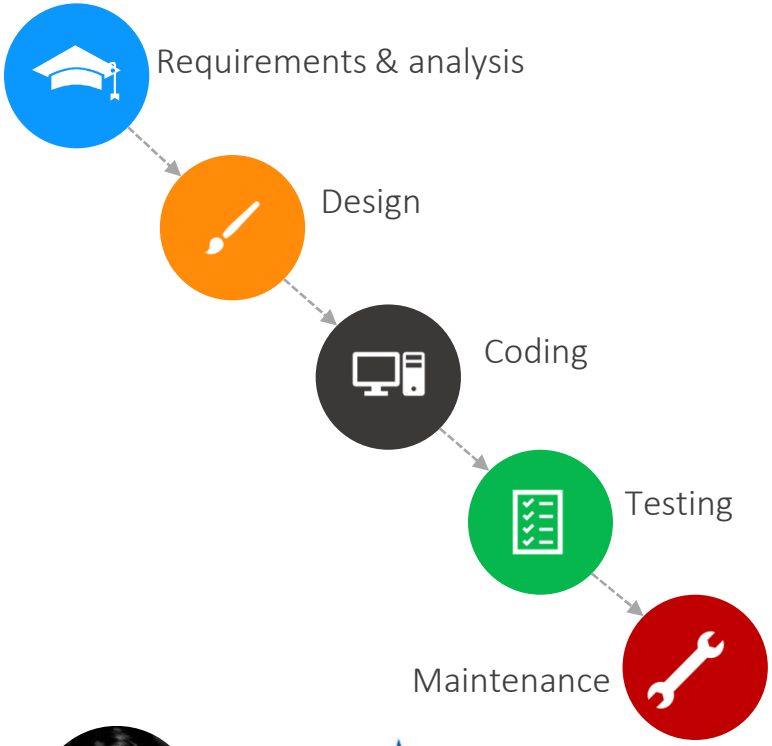
- I have always had an interest in Management studies and hold a Masters in Public Administration (MPA)
- Have been exposed to many management trends over the years, and have taught Lean techniques in my IA course
- Am particularly interested in effective management of documentation groups
  - Since 2015 have been interviewing various tech writers and managers about their Agile + DITA experiences

precision content

Business Reasons for Using Agile within Technical Documentation

Waterfall Management and Technical Writing

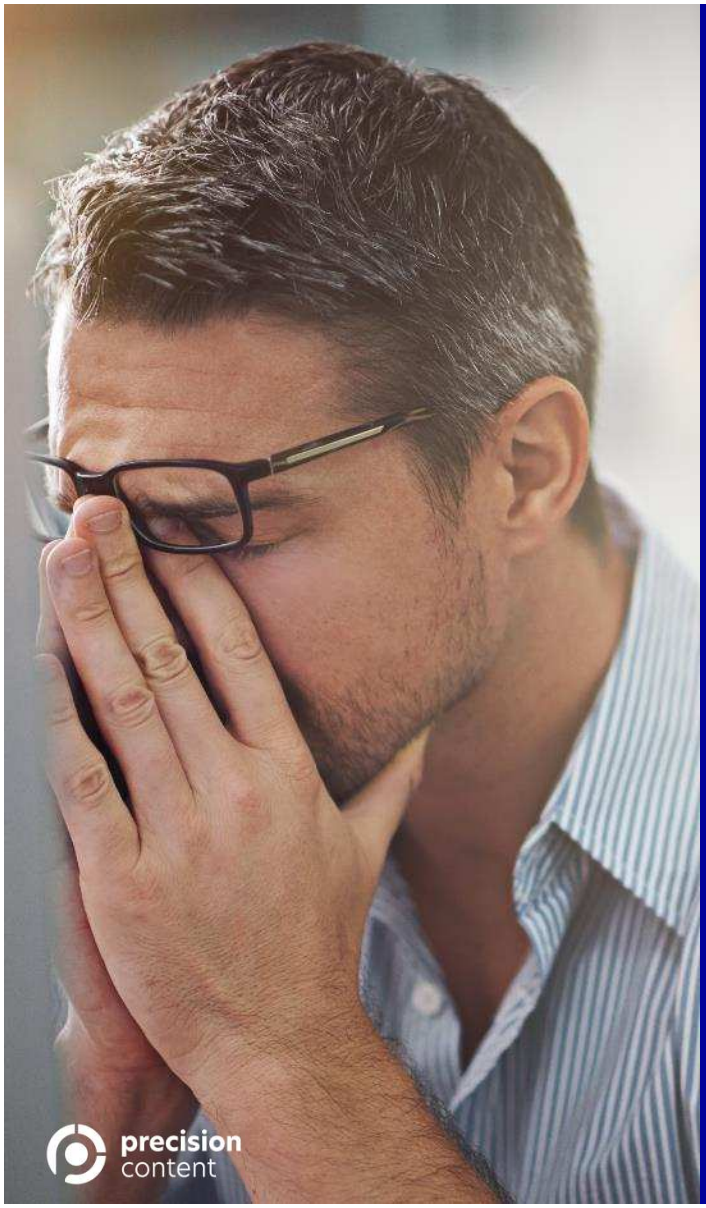
- The waterfall model also began in the software development realm, first described in detail in early 1970s
- Sequential design process, starting with analysis and ending with maintenance (updates)
- Any technical writing typically fell between the Coding and Testing phases, well after Requirements and Design phases
- “Just document what’s there (or will be there).”
- Majority of documentation teams still follo this model (and there’s nothing necessarily wrong with that)



DELRINA

A certain young tech writer at
Delrina, circa mid-1990s
Pre-Agile, definitely Waterfall

precision content



Problems with Waterfall Management

- Waterfall-based software projects were prone to failure
- In 1995 DoD found that of $35.7 billion spent by the organization on software, only 2% of the software was usable as delivered, and that 75% was either never used or was cancelled prior to delivery.
- Waterfall does not deal with changing/adapting to customer needs gracefully

The Agile Manifesto was written in 2001:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions *over processes and tools*

Working software *over comprehensive documentation*

Customer collaboration *over contract negotiation*

Responding to change *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more."

A clear difference from the traditional waterfall approach to docs

Agile Implications for Documentation Processes

- Content creators have to work more closely with developers
- Documentation may support broader communication, such as between teams, customers, audit process, etc.
- Work cycles are faster, feedback more critical
- Efficient documentation tools make things easier (single sourcing, structured content, CCMS)

precision content

What Agile Means in Practice for Tech Doc Teams

1. Work more closely with developers

   Tech writer assigned to work with one or more development teams; does regular reports on progress (Scrum with Stand-up meetings, Kanban priorities, etc.)

2. Provide early feedback on product

   Through active use of product tech writers often become advocate for users; helps define realistic user stories

3. Constant change/iterations of content

   Incremental releases, and a change of focus from “document everything” to “document only what the user needs”

precision content

Agile in Technical Documentation

- I have been interviewing tech doc members at various companies who said they were Agile.
- But when I ask them about their processes they are often in fact:
  - Largely Waterfall-based, or
  - Do not want to be perceived as not being Agile (whatever that is), or
  - "Rest of company is Agile so we must be too" (i.e. "Agile by osmosis"), or
  - Transitioning to some flavor of Agile.
- I found "pseudo-Agile" doc teams to be surprisingly common, and might even be the norm. However, *all* agreed that DITA made their jobs easier.

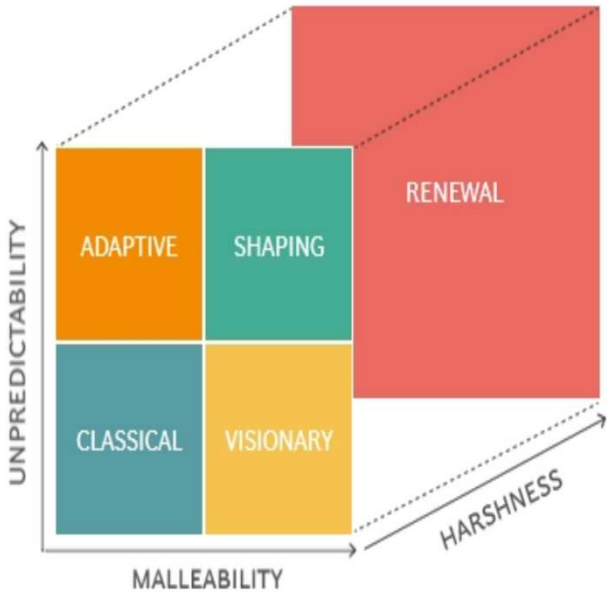
Agile is Not for Every Business Environment

Agile thrives in environments where short release cycles are possible; appears to be rare in highly regulated environments or heavy machinery for example.

- Have found one case of Agile being used in Pharma, though only after and their software division successfully adopted it
- I have found instances of Agile + DITA in Medical Devices, Heavy Machinery sectors, but in these cases it was by the software divisions within these firms

In environments where business factors are pushing for rapid change in product development, Agile methodologies are likely to be introduced

Where Agile Thrives



- In general, software is in the “Shaping” quadrant of industry types, where environment is unpredictable but where expectations of customers can be modified (or “shaped”)
- Rapid testing of releasable product helps shape the market/user’s expectations; Agile approach makes this possible
- Need an equally “agile” way to manage documentation

precision content

Evolution of DITA within Technical Writing

DITA

DITA is arguably the natural evolution of the need for many writers to work and reuse content more efficiently

Agile

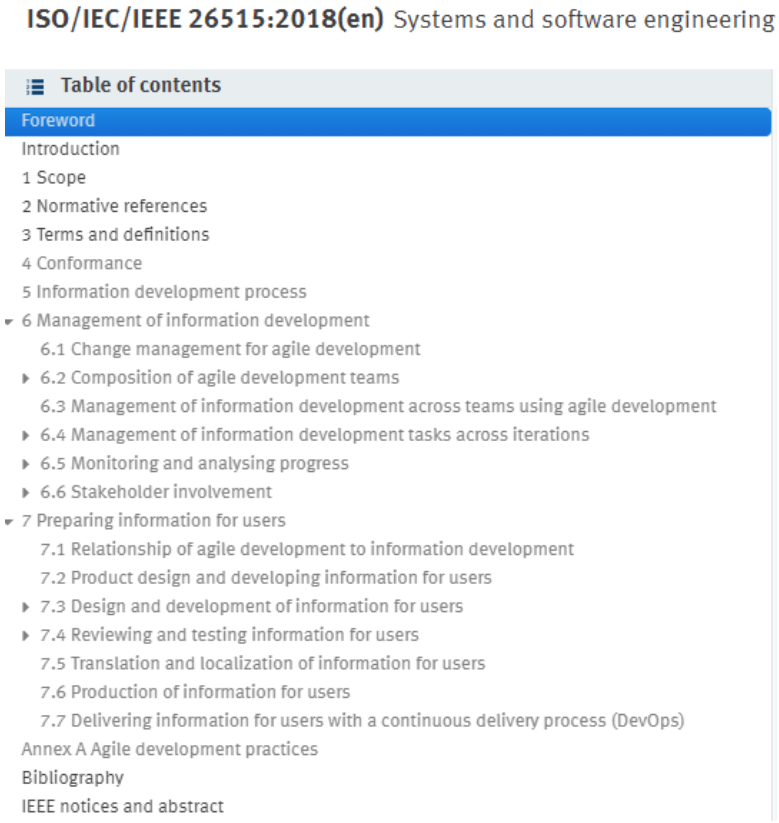
Similarly, Agile can be seen as meeting the need of fast-moving development teams to meet the needs of their customers

The two methodologies compliment each other (when done right)

precision content

ISO Standard for Agile Documentation

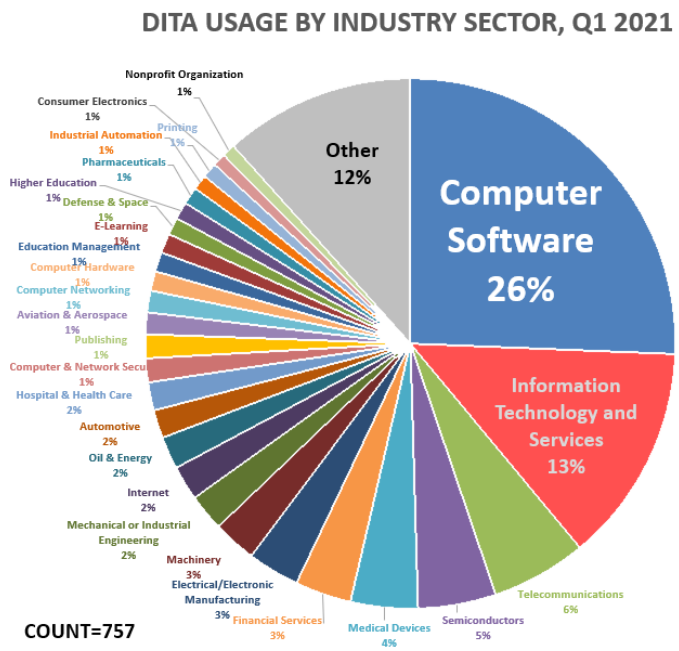
- ISO/IEC/IEEE 26515:2018 is an ISO standard describing how to develop user documentation in an Agile environment
- It “provides guidance on processes appropriate for information developers of information for users in software and systems projects that are using Agile development methodologies”
- Neatly outlines everything you need to know about Agile + documentation
- Most recent version is from 2018



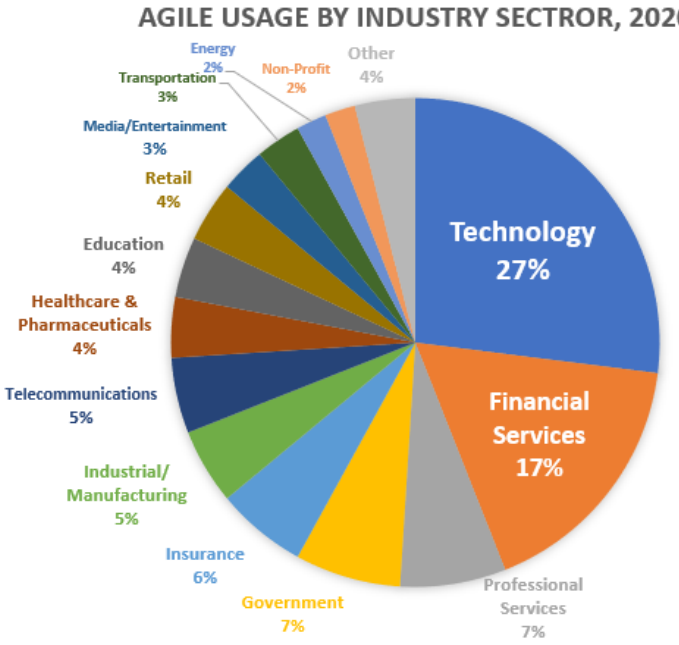
precision content

Similarities Between Agile and DITA Sectors

- Though not a direct correspondence, there is at least some cross-over between DITA- and Agile-using firms



Source: DITAWriter.com

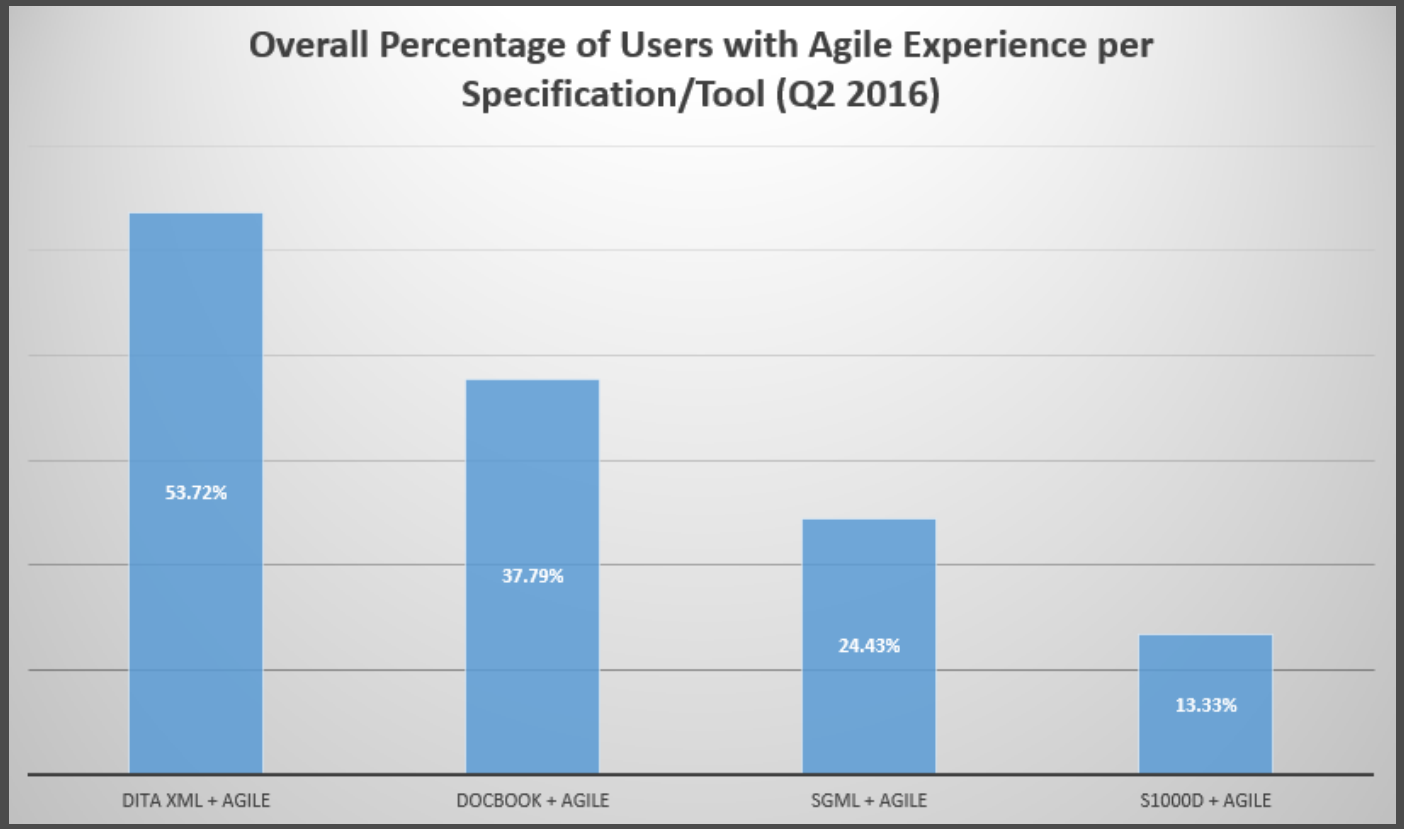


Source: VersionOne 14th Annual State of Agile Report.

precision content

DITA is Clearly Popular Among Agile Teams

- DITA is the most popular form of structured content used by Agile teams (data from LinkedIn):



precision content

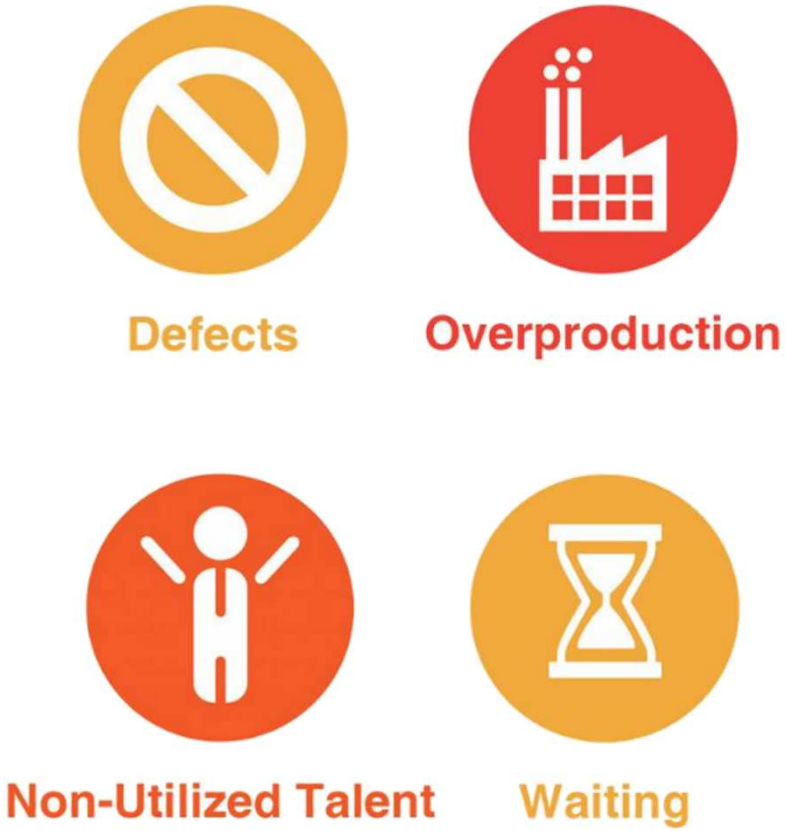
Lean and the 8 Types of (Documentation) Waste

1. Defects: these are the errors, omissions, and information that is simply irrelevant to the customer.

2. Over-production: this describes “kitchen sink” docs, which includes everything users are thought to need to know.

3. Non-utilized talent: this boils down to under-utilizing people’s time, skills and knowledge. This could include pigeon-holing team members to a single responsibility, or requiring everyone to attend meetings they are not truly needed to be there.

4. Waiting: includes the time wasted for budget approval, authorization to begin, and for reviews when documentation work could be done.



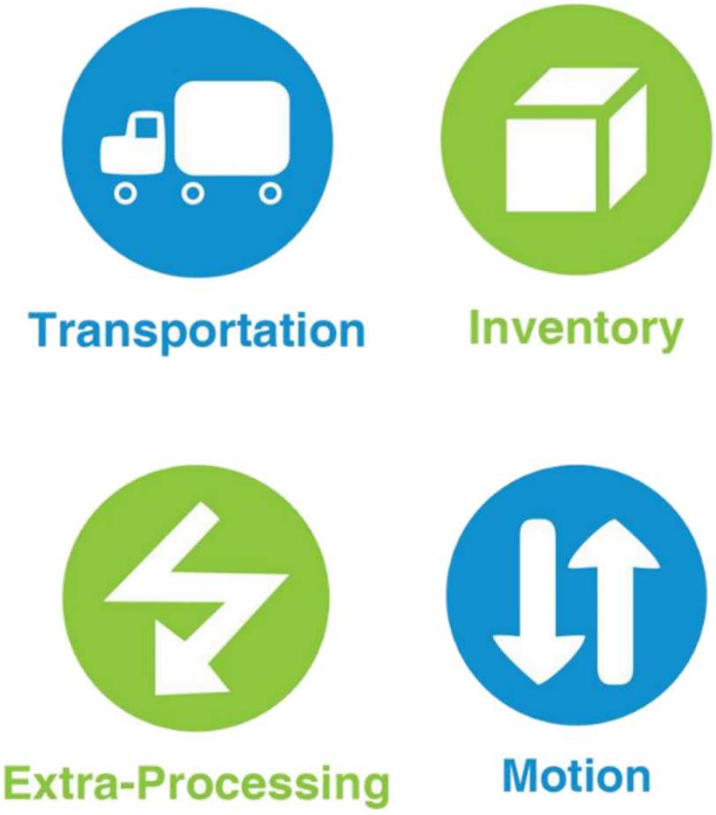
Lean and the 8 Types of (Documentation) Waste

1. Transportation: this includes sending files for review in print or non-native formats, moving content from one program to another for differing outputs, or copying and pasting content from file to file.

2. Inventory: this is when content doesn't meet the need of end users, so for example topics are either never accessed by end users because they are not useful, or because users cannot find them.

3. Excess processing: is the duplication of effort where the same content is developed by multiple groups.

4. Motion: when a person has to travel (i.e. to a meeting physically located elsewhere), the time that it takes for that person to travel is never recovered.



The Reasons Why DITA Works So Well with Agile

Agile and Content Reuse in DITA

Content reuse: “write once, use many”

- No need to re-write what already exists
- Content consistency
- Single-sourcing is built in

“[DITA] handles the reuse of small information chunks brilliantly. My engineers reused functions and objects constantly as they developed new features. I found it invaluable to be able to conref (reuse by reference) previously written tables, sections, paragraphs, procedure steps, etc.. During that last long night at the end of a sprint I was never too proud to reuse available writing.”

– Stan Doherty

precision content

Topic Reuse Improves Content Consistency

- This is an additional benefit of content reuse

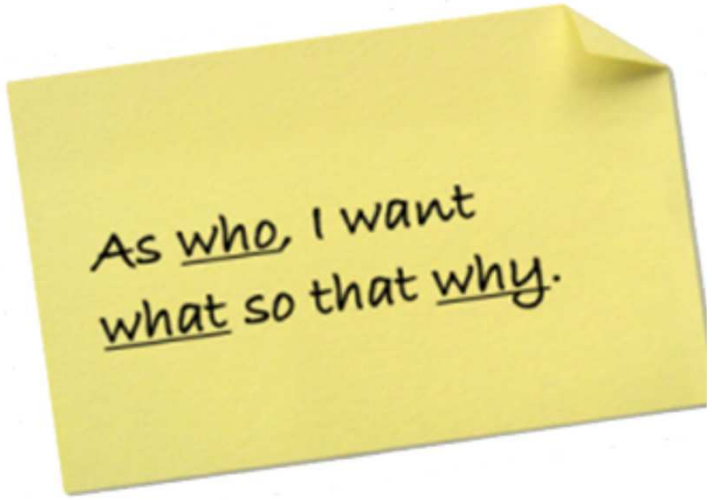
Examples:

- Shared “key warehouses” between maps can hold document “variables” (like product name, model number, trademark terms, etc.) which can easily be shared between maps
- Phrase and topic-level reuse convey the same messaging to users, improving the perception of the brand
- Content that has been reviewed by SMEs already can be flagged as such (within a CCMS) so they do not have to be reviewed again

precision content

Agile User Stories Maps Well to DITA Task Topics

- Scrum-based Agile often calls upon User Stories to craft development
- Often take form of various procedures that users will want to accomplish; this fits well with DITA's task topic type



"DITA allows correlating user stories to specific procedures much easier than other less granular formats. This can be utilized in some pretty creative ways to apply principals of continuous integration, and testing to documentation."

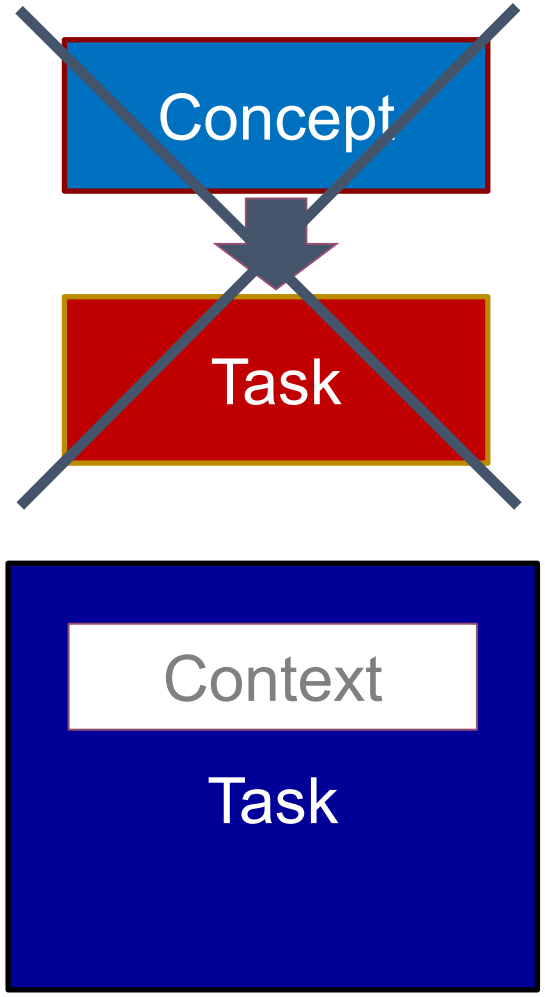
- Casey Jordan

precision content

Agile User Stories Maps Well to DITA Task Topics (cont.)

A possible DITA + Agile best practice for writing tasks emerged from my interviews:

- Instead of writing a concept to be followed by a task, encapsulate that concept as the context for a task instead
- Depending on scenario, describe expected outcomes for individual steps/conclusion
- Use concept topics to link between tasks



precision content

Agile Epics and DITA Audiences

- “Epics” are a collection of related user stories that comprise the complete workflow for a type of user
- From a DITA standpoint, epics can be used to help refine conditional processing for audience
- User types may change during development; the “agility” and flexibility of DITA makes it possible to change quickly

precision content

DITA Best Practices Advocate Focusing on the User

- This follows from how user stories map to DITA task topics
  - Leads to an emphasis on core tasks
  - User does not have to wade through irrelevant content (to them) in order to "get things done"
- Similarly, the other main topic types (DITA's concept, reference, troubleshooting, and PCAS' additional process and principle) are deliberately structured "boxes" that should make a good technical writer think about what information the user needs

Tech Writers Become Part of Feedback Loop

Tight integration of tech writers with development team opens possibilities for early feedback on product development

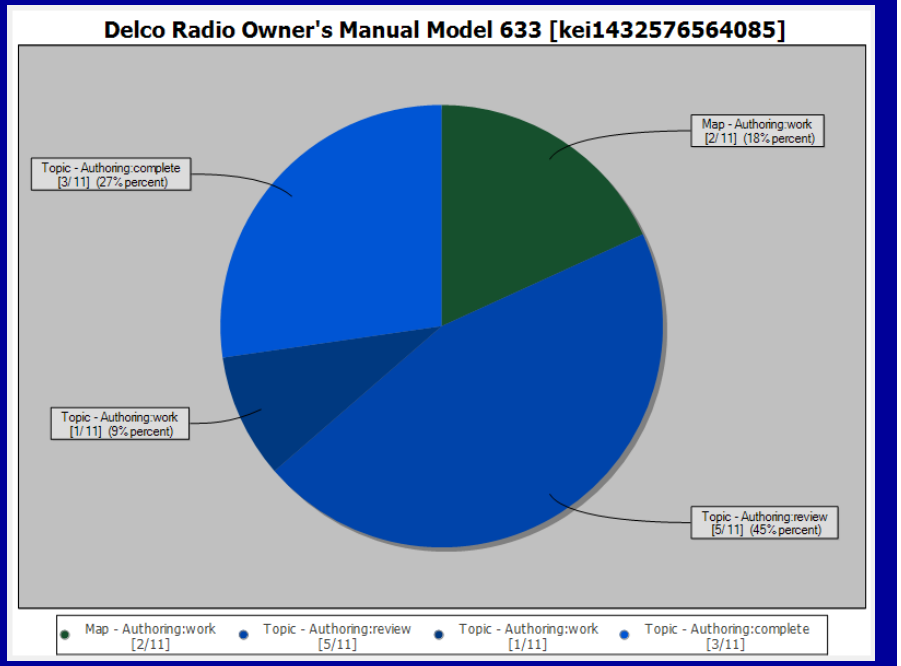
"The goal of technical communicators is not to explain confusing product features, but to prevent them."

– Tim Grantham

precision content

DITA Topics Allows for Doc Project Measurement

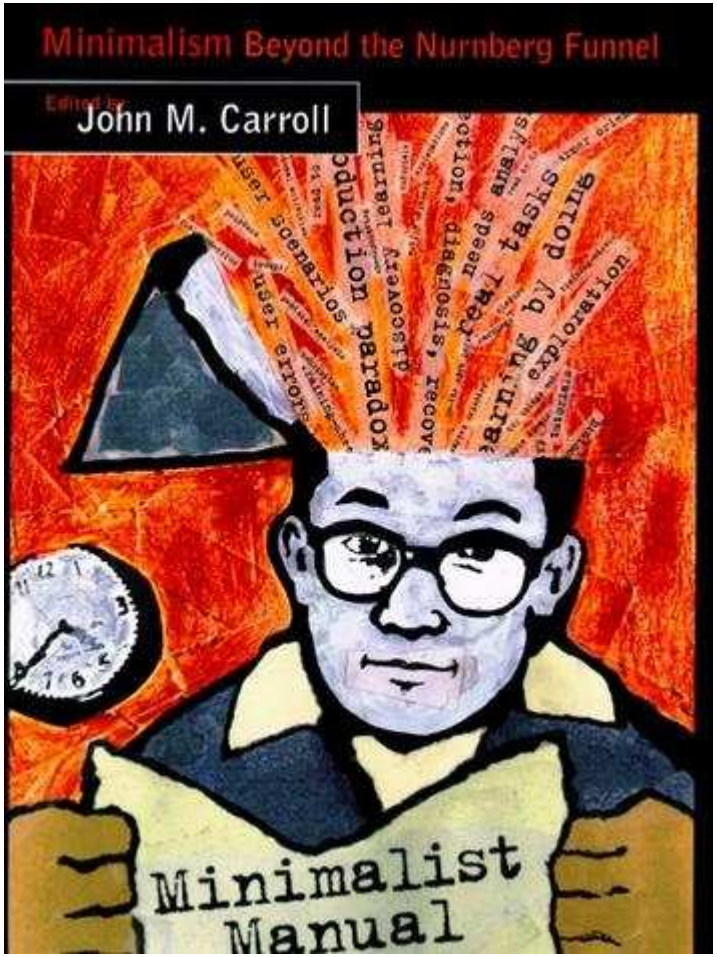
- DITA's topic-based approach also makes it easy to measure content
- Within a CCMS it is also possible to track how "done" topics within a map are
- Non-structured docs are much harder to track due to lack of this level of granularity



"Our project managers could track progress of documentation deliverables within our DITA-based CMS on a daily basis."

- Jason Owen

precision content



DITA Best Practice of Minimalism Reduces "Waste"

According to John M. Carroll, genuinely useful information for users consists of the following components:

- Available when and where it is needed
- Easy to find
- Immediately useful
- Concise and to-the-point

When done right, DITA content + minimalist writing hits all of these points

precision content

Separation of Content from Formatting Saves Time

- Time is spent writing rather than formatting
- Separating content from formatting saves considerable time

In an informal survey done on a team of technical writers working with non-structured content, roughly half of their time was spent formatting that content. That time can now instead be put towards writing more Agile content in a structured XML environment.

precision content

Feedback is Part of Agile

- Documentation feedback is a developer requirement under Agile
- Using DITA, turnaround of topic-based review with SMEs much reduced
  - SMEs can provide feedback in a more timely manner
  - Easier to review a few topics than a whole book!

"Developers would review topics on the spot in the Agile team room. Agile also left no room for procrastination, so this was an easy way for them to check this off their own task list."

- Jason Owen

precision content

Agile Documentation and the “Definition of Done”

- From a practical perspective, in order to make documentation review work, it needs to be considered part of the definition of “done”
- This means that the tech writer/manager needs to be fearless at a scrum meeting and say when something is not “done”
  - Anecdotal evidence suggests that when SMEs are made accountable in this manner, they are more likely to work with you in the future

precision content

Only Document What is Necessary

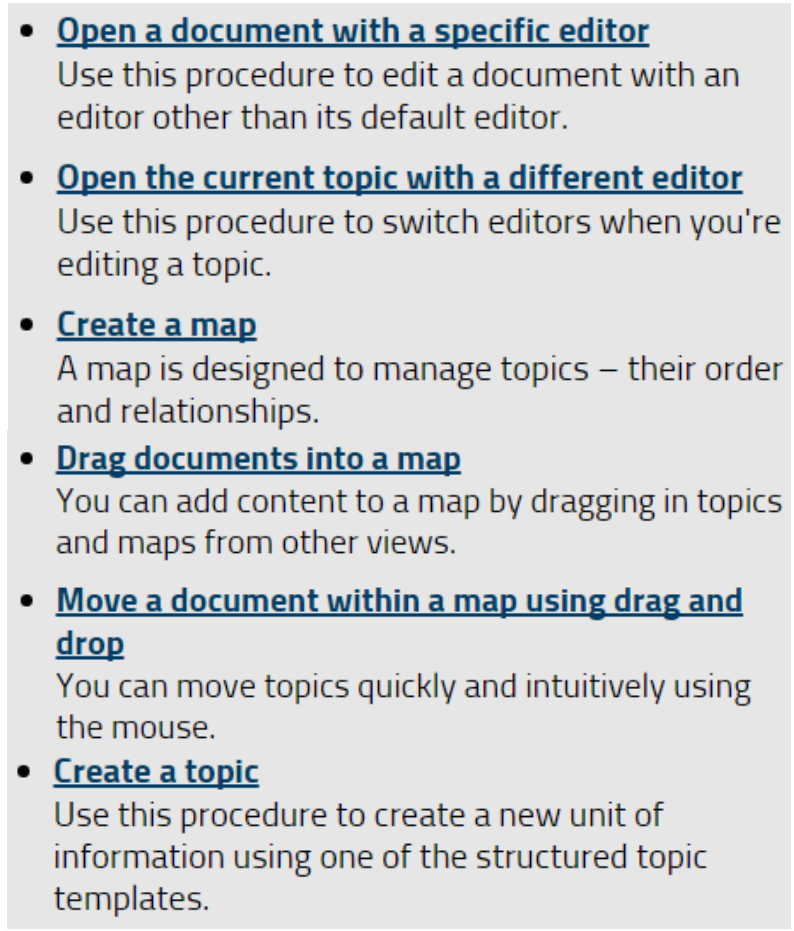
- Not only based on feedback from developers, but also from users
- Fits with minimalist writing principles; ditto Lean
- When possible, track online usage from published docs, and prioritize user-favored content
  - One interesting example: UI-related content “how to” style content is reduced and UX is improved by writer feedback, ensuring UI is more usable

precision content

Short Descriptions Direct Users to Content

Writing short description for DITA topics is already considered a best practice

- Arguably more so for Agile-based content, as it provides a means of progressive disclosure as to the relevancy of content to users
- Can be similar in intent to a user story: “User x can do y based on z”



- **Open a document with a specific editor**
  Use this procedure to edit a document with an editor other than its default editor.
- **Open the current topic with a different editor**
  Use this procedure to switch editors when you're editing a topic.
- **Create a map**
  A map is designed to manage topics – their order and relationships.
- **Drag documents into a map**
  You can add content to a map by dragging in topics and maps from other views.
- **Move a document within a map using drag and drop**
  You can move topics quickly and intuitively using the mouse.
- **Create a topic**
  Use this procedure to create a new unit of information using one of the structured topic templates.

precision content

DITA Was Built with Multi-channel Publishing in Mind

DITA Open Toolkit (DITA-OT) was designed with this in mind

- Origins: prior to DITA at IBM, multiple processes and tools were needed to create documentation in more than one format.
- DITA + DITA-OT was the solution

Reduces wasted time/resources that would otherwise be spent using additional tools

precision content

DITA 1.3 Troubleshooting and Agile

- DITA 1.3 added troubleshooting as a new topic type
- Designed to provide specific solutions to scenarios that are likely to arise, and how to solve them
- Welcomed by Agile writers who are looking for a troubleshooting option for user stories and where a task may not be an appropriate solution

"The troubleshooting pattern of condition > cause > remedy is essentially a scenario."

- Bob Thomas

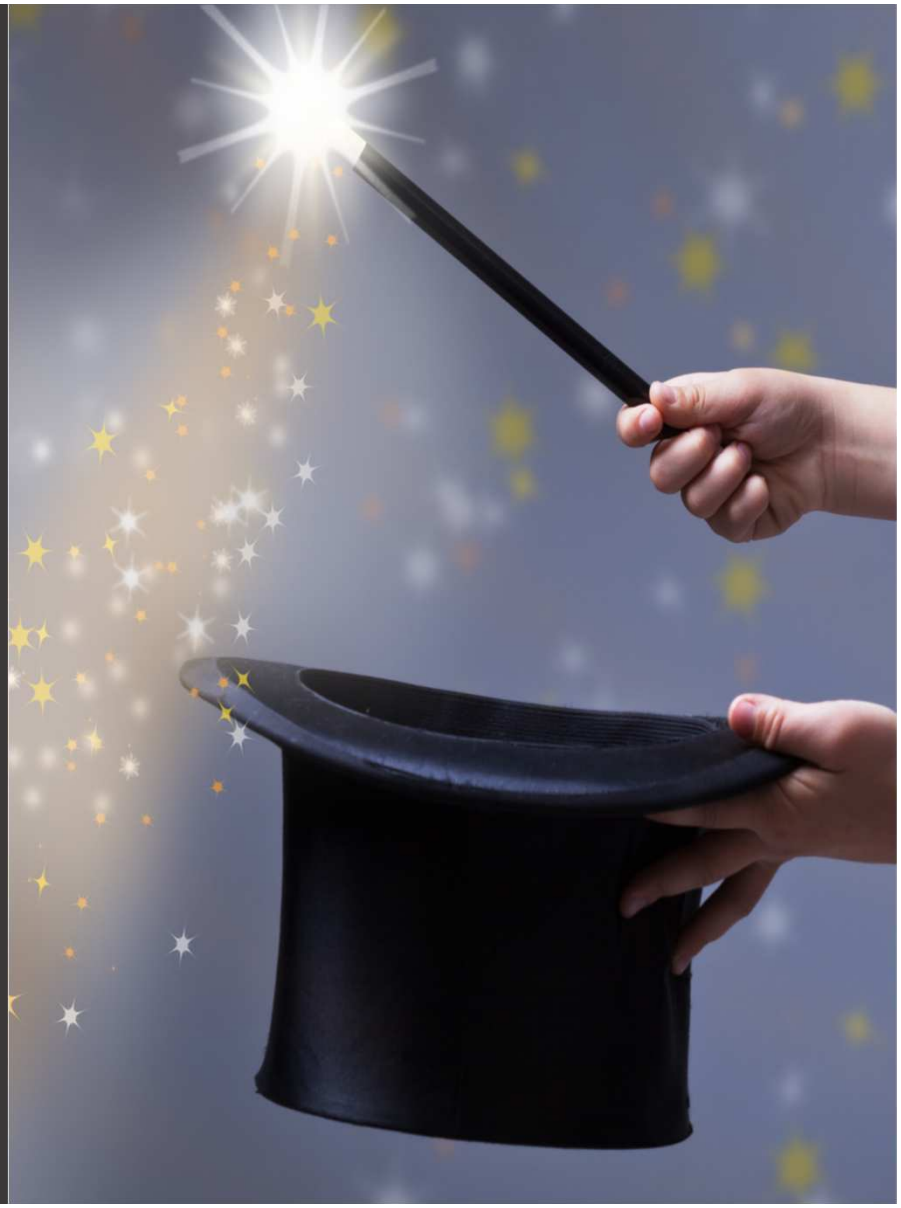
precision content

Documentation Does Not Happen by Magic

Problem #1: Agile will not solve understaffed tech doc teams

Symptoms:

- Writers cannot attend stand-up meetings due to scheduling conflicts
- Writers perennially falling behind on assigned topics to write
- A good Agile process manager will recognize the problem and either throttle back work or bolster effort for new hires



precision content

An Unexpected Need for Documentation “Glue”

Problem #2: Need to make documentation “glue” for publications

- Applies to cases where a full manual is expected
- High-level introductory or conceptual material not typically accounted for in a sprint
- There’s still a need to answer the “why would you use this?” type of question

Solution is to recognize this need up front, and allow for it in the overall documentation plan.

An Example of How DITA Can Enable Agile

1. Lean methodology employed at AMD; early on localization was a focus:
   - Under old toolchain could only localize software (with 1 month cadence) once every 6 months
   - Using non-structured content based processes, it was costly, slow and process did not allow for feedback
2. DITA + CCMS made localizing on a monthly cadence possible
   - Demonstrated considerable costs savings
   - Localization staff could focus on quality and provide developers with feedback

Localization Process Pre-Lean:

Localization Process After Lean + DITA + CMS:

precision content

DITA + Agile Case Study #1

Semiconductor industry example:

- Prior to move to DITA, used traditional Waterfall method
- A “doc build” could take as long as a day; longer if the program crashed
- DITA opened up possibility of moving tech docs to a more Agile approach

Results:

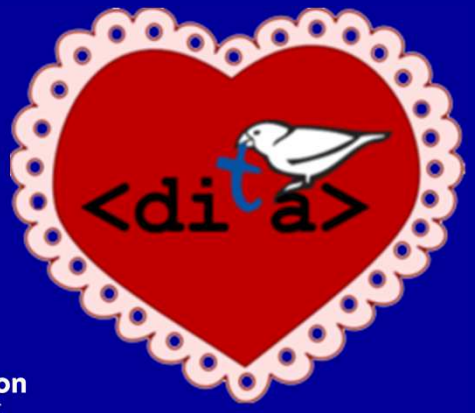
- Considerable time saved by no longer dealing with formatting issues
- Topic-based review process + CCMS workflow has improved SME feedback
- Can now do daily “publication builds” of their content; early access given to tier 1 clients

precision content

DITA + Agile Case Study #2

Software sector example:

1. Writers were already embedded in software development teams, but with their existing, inefficient documentation tools they were always playing “catch up”
2. Lack of granularity meant that the non-structured content they produced was hard to track



precision content

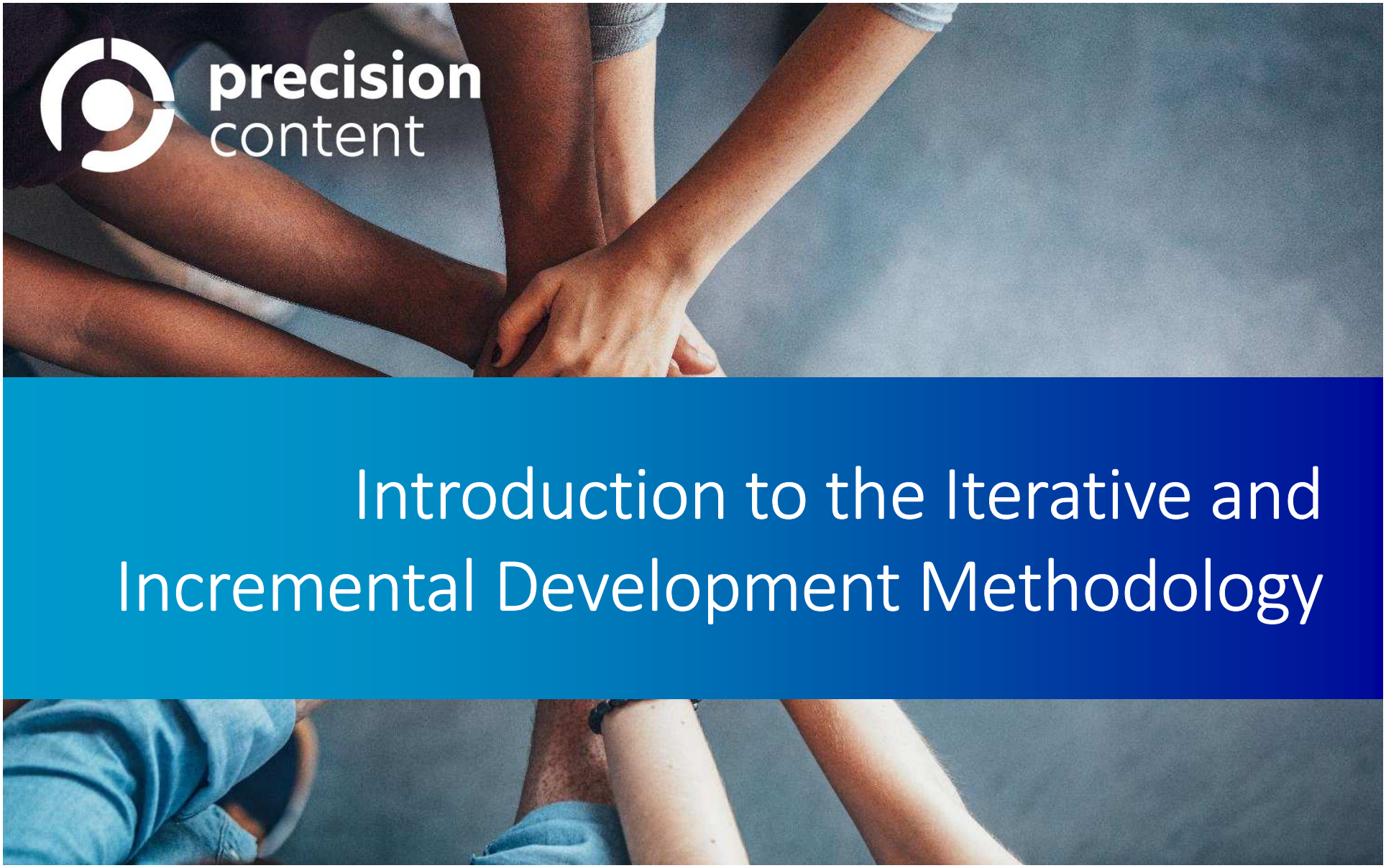
Results:

- DITA + CCMS means that writers now have the time to both create content and to participate fully in the Agile process
- Per topic progress reports now possible; now a regular part of scrum meetings, and can even be done on-the-fly on request

Why DITA + Agile = A Great Partnership!

DITA	Agile
Topic-based approach	Incremental development; can update topics as required; self-contained
Task topics	User stories; content is focused squarely on what user needs to do
Individual topics can be counted; in a CMS workflow can be measured	Need to track development progress; easy to demonstrate progress
Best practice of minimalism	Document only what needs to be documented; Keep It Straight and Simple (KISS) and Keep It Light (KIL); reduces waste
Reuse improves content consistency	Continuous feedback from developers and users
Iterative publication to multiple formats on demand	No holdup for incremental product releases

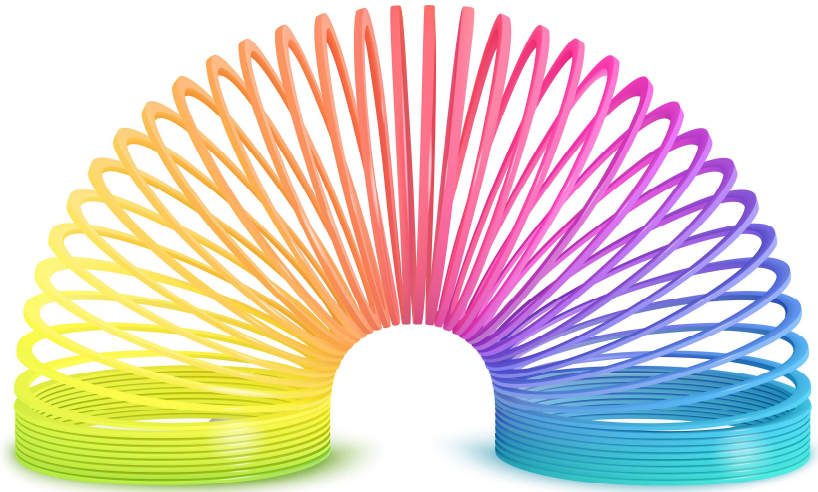
precision content

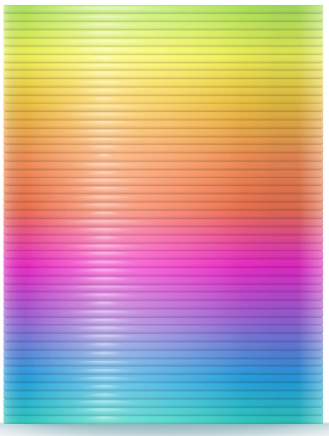


Introduction to the Iterative and Incremental Development Methodology

Some Issues Particular to TechDoc Groups

- Need for a methodology more obviously built around deadlines.
- Need earlier and more frequent deliveries and user feedback loops so requirements can be refined sooner rather than later.
- Need a plan that allows for flexibility to absorb changing requirements, while still driving towards a larger set of goals and deliveries.
- *Key thing is a need for flexibility while also retaining a structured process*





precision content

Introducing…

Iterative and Incremental Development (IID)

1. From Wikipedia:
   - "The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental)" allowing teams to learn from development of earlier parts or versions of the system, and user feedback.
2. From Techopedia:
   - "Iterative and incremental development is a method… that is modeled around a gradual increase in feature additions and a cyclical release and upgrade pattern."

Where Does Agile Fit?

- Scrum methods and Agile values and principles are both incremental and iterative in approach. They are
  - Iterative because the work of one iteration is improved upon in subsequent iterations, and
  - Incremental because completed work is delivered throughout the project.
- Trends over the past few years show us that Agile teams thrive when using both.

  "When it comes to complex software, each technique fills in the gaps created by the other. In using the two together, you can complete your software in increments while delivering completed work throughout the course of the project."

  From: https://backlog.com/blog/importance-iterative-incremental-software-development/

Precision Content has started using IID in our development and documentation processes.

precision content

IID Planning Cadence

Project				
Increment 1			Increment 2	
Iteration 1	*Iteration 2*	*Iteration 3*	*Iteration 1*	*Iteration 2 …*

1. A project starts with a Roadmapping exercise.
   - In this meeting, the project leaders prioritize and sequence the work to be done.
   - Expected outcome is a Project Roadmap with milestones and portioned deliveries per *increment*.
   - The roadmap guides us into the iteration planning meeting.
2. In the Iteration Planning meeting, we identify a the Minimum Viable Product ("MVP", which is a portion of the full increment goal) which involves all aspects of the increment, and that can be handed to the client for review and feedback.
   - Before the end of the iteration, we deliver a MVP (product feature and docs).
3. The *next* Iteration Planning meeting pulls in previous feedback and any lessons learned, and identifies the next small MVP we deliver, and that progresses us towards the larger *increment* goal.

Once the increment is completed, we are left with a finished "something" that we don't revisit or reopen further down the project. Our first milestone will have been met. The same approach is followed for subsequent iterations and increments.

precision content

What's in an Increment

Project (SOW)										
Increment 1			Increment 2				Increment 3		Increment 4	
Iteration 1	*Iteration 2*	*Iteration 3*	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*	*Iteration 1*	*Iteration 2*	*Iteration 1*	

- In each *increment*, a slice or chunk of (completed) software (and documentation) is delivered through cross-team/multidisciplinary work. We build and deliver in incremental pieces until the whole project is complete.
- Each increment is divided into one or more iterations, each of which solicit user feedback and build on what's been learned from previous iteration.
- Each increment is a subset of the functionality, with contributions from all departments, and the deliverable at the end is fully tested (end-to-end testing).
- By the end of an increment, several rounds of user feedback and adjustments will have competed (at the end of each iteration). Therefore the delivered subset of documentation at the end of an increment should be deemed to be complete and not require later revision.

precision content

What's in an Iteration

Project (SOW)										
Increment 1			Increment 2				Increment 3		Increment 4	
Iteration 1	*Iteration 2*	*Iteration 3*	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*	*Iteration 1*	*Iteration 2*	*Iteration 1*	

1. An iteration is time-boxed, similar to a sprint, but typically longer. They are numbered.
   - Iterations can vary in length, but commonly fall between 3-8 weeks.
   - Iteration lengths do not need to be equal (Iteration 1 can be 3 weeks, Iteration 2 can be 5 weeks, etc.)
2. The full team contributes to each iteration delivery; other teams do not wait until a later iteration to contribute.
3. Each iteration contains the same Line Items (aka departmental buckets of work) that run through the entire project e.g. Transformation, Conversion, Modelling, Publishing. Tasks will vary accordingly.
4. A set of documentation delivered at each iteration.
5. Progress is made through successive refinement. For example:
   - During the first iteration, HTML pages are coded to support only a very simple type of search.
   - During the second iteration, additional metadata is added.
   - During the third iteration, search is tested against content and metadata.

precision content

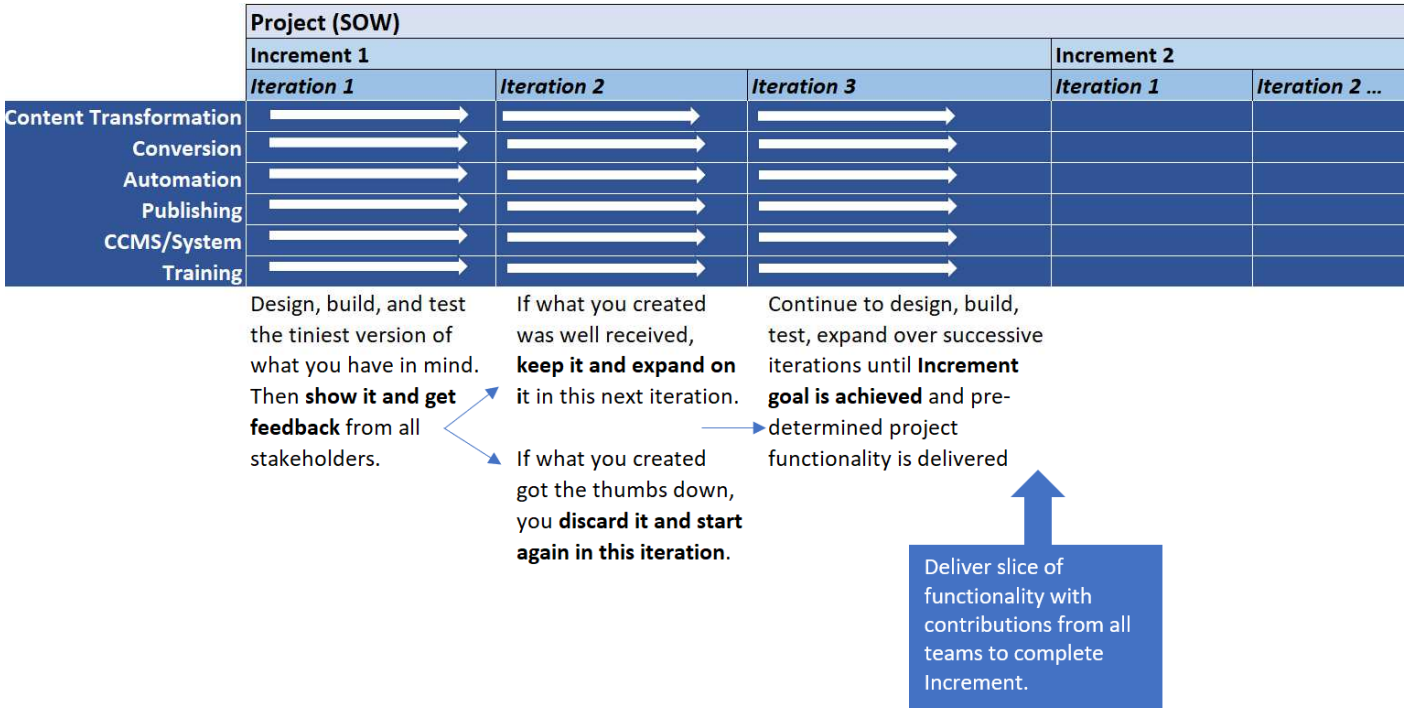
What's in an Iteration, con't

Project (SOW)										
Increment 1			Increment 2				Increment 3		Increment 4	
Iteration 1	*Iteration 2*	*Iteration 3*	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*	*Iteration 1*	*Iteration 2*	*Iteration 1*	

1. If iteration commitments can't be met, ***reduce the scope*** – don't move the iteration end date.
   By the original planned iteration end date, the partial system should be in a stable and tested state.
2. Consider team capacity when planning scope of the iteration – team size, availability, velocity, and provided efforts.
3. Typically, each iteration plans to meet a set of specific objectives and addresses a group of project risks. Tasks are completed within the iteration to meet the iteration goals.
4. "The iteration plan information should be compiled into a simple document. The first section of the plan should reflect the scope and objectives of the iteration. The primary goal of the scope is to make clear the iteration's objectives, priorities, and evaluation criteria. This information can be used to track the iteration's progress and drive the iteration assessment. The rest of the plan captures the detailed plan for the execution of the iteration."

precision content

Typical IID Flow



precision content

Iterative and Incremental Development is Ultimately:

- Built around achievable milestones
- Better engages with internal resources every step of the way
- Provides earlier and more frequent deliveries, giving the development team (and tech docs) more opportunities to give user feedback, perform user testing, and accept delivered functionality, and
- It remains flexible and can absorb changing requirements while still driving towards a larger set of goals and deliveries.

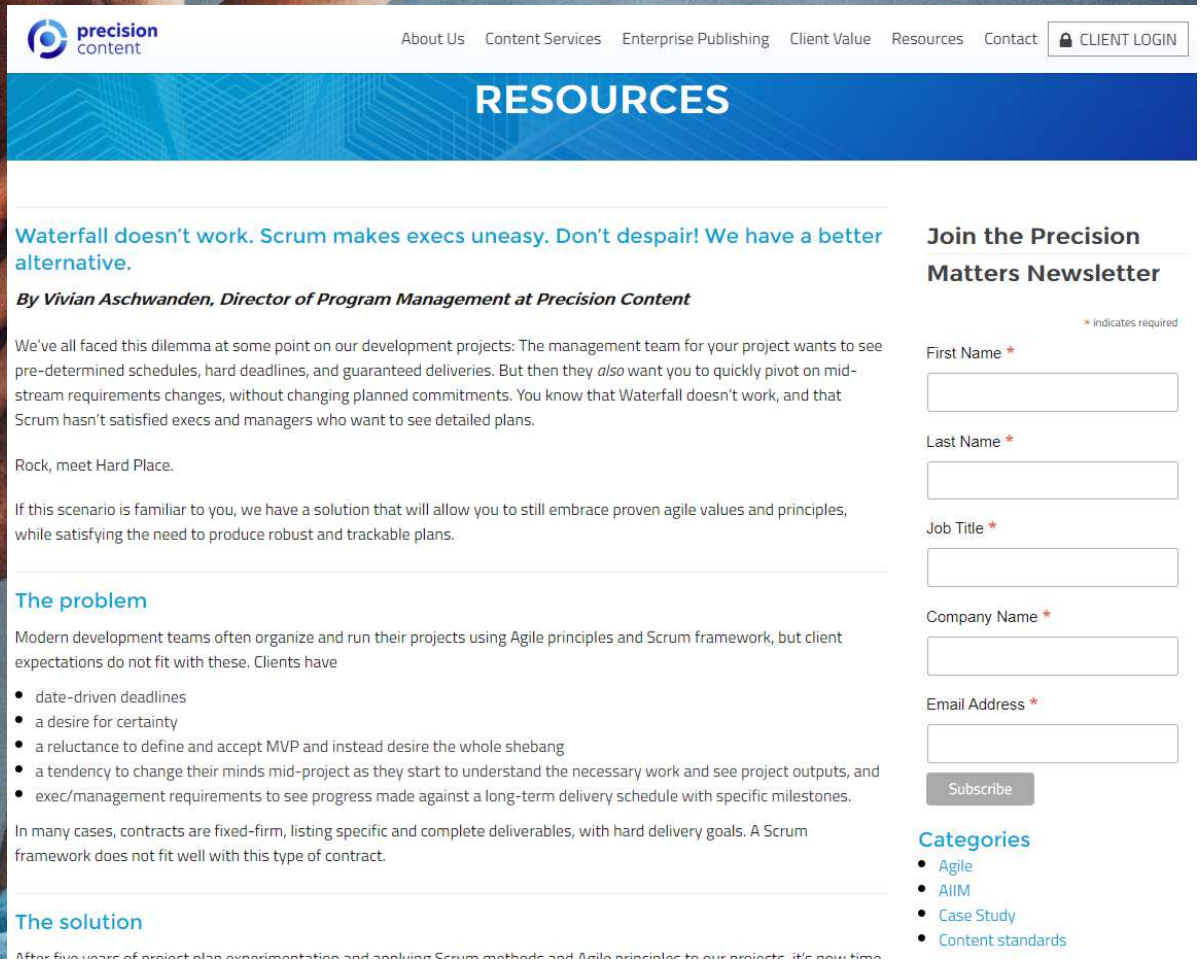
precision content

precision content

Questions?

keith@precisioncontent.com

For further reading, see Vivian Aschwanden's blog post on the Precision Content website at:
https://www.precisioncontent.com/blog/iid/



precision content

About Us | Content Services | Enterprise Publishing | Client Value | Resources | Contact | CLIENT LOGIN

RESOURCES

Waterfall doesn't work. Scrum makes execs uneasy. Don't despair! We have a better alternative.

By Vivian Aschwanden, Director of Program Management at Precision Content

We've all faced this dilemma at some point on our development projects: The management team for your project wants to see pre-determined schedules, hard deadlines, and guaranteed deliveries. But then they *also* want you to quickly pivot on mid-stream requirements changes, without changing planned commitments. You know that Waterfall doesn't work, and that Scrum hasn't satisfied execs and managers who want to see detailed plans.

Rock, meet Hard Place.

If this scenario is familiar to you, we have a solution that will allow you to still embrace proven agile values and principles, while satisfying the need to produce robust and trackable plans.

The problem

Modern development teams often organize and run their projects using Agile principles and Scrum framework, but client expectations do not fit with these. Clients have

- date-driven deadlines
- a desire for certainty
- a reluctance to define and accept MVP and instead desire the whole shebang
- a tendency to change their minds mid-project as they start to understand the necessary work and see project outputs, and
- exec/management requirements to see progress made against a long-term delivery schedule with specific milestones.

In many cases, contracts are fixed-firm, listing specific and complete deliverables, with hard delivery goals. A Scrum framework does not fit well with this type of contract.

The solution

After five years of project plan experimentation and applying Scrum methods and Agile principles to our projects, it's now time

Join the Precision Matters Newsletter

* indicates required

First Name *

Last Name *

Job Title *

Company Name *

Email Address *

Subscribe

Categories

- Agile
- AIIM
- Case Study
- Content standards