AGILE SOFTWARE DOCUMENTATION

Photo courtesy of www.mojomechanics.com

Mike Ziegenhagen

mikeziegenhagen@yahoo.com

North Bay STC meeting

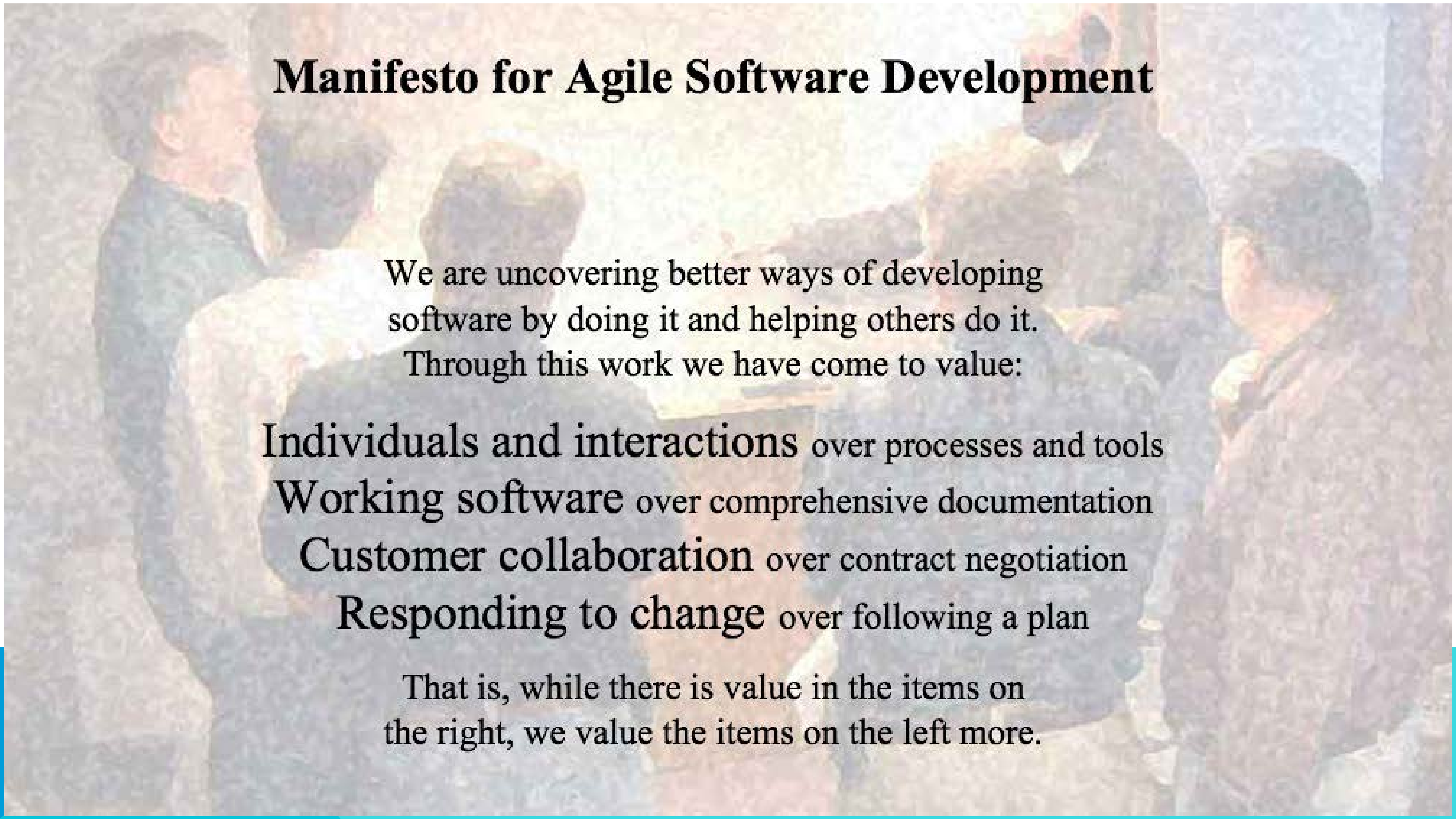
March 15th, 2012

HERE'S THE PLAN

- Short overview of ***Agile Software Development***
  - *Wikipedia Definition:* Agile software development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.
- Introduction of ***Agile Software Documentation***
  - *Wikipedia Definition: The page "Agile software documentation" does not exist.*

AGILE SOFTWARE DEVELOPMENT

(A mini overview)



Manifesto for Agile Software Development

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.

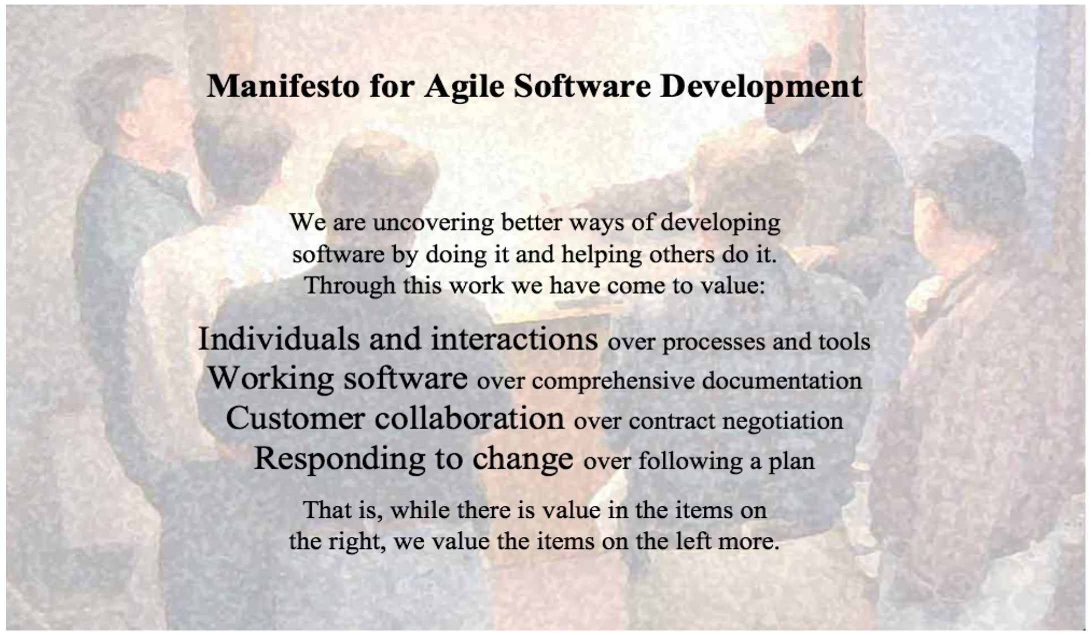
THE DETAILS

Individuals and Interactions – in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

Working software – working software will be more useful and welcome than just presenting documents to clients in meetings.

Customer collaboration – requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.

Responding to change – agile development is focused on quick responses to change and continuous development.



Manifesto for Agile Software Development

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.

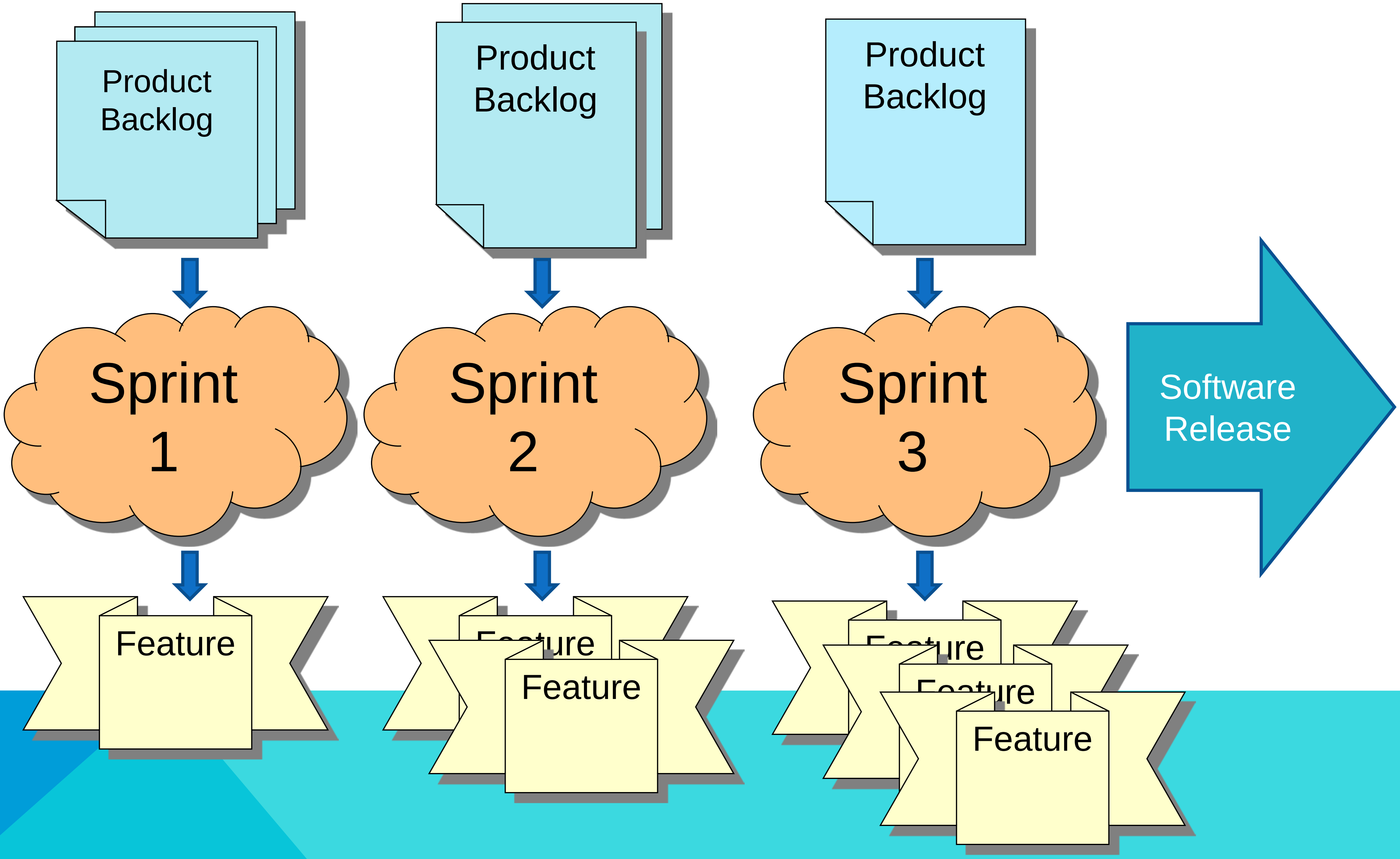
PRINCIPLES

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

PRINCIPLES (CONT'D)

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

ITERATIVE SOFTWARE DEVELOPMENT



HELPFUL JARGON

- **ROLES**
  - Scrum Team
    - Cross functional
    - Many small scrums
  - Scrum Master
    - Runs meetings
  - Product Owner
    - Represents user
    - Makes Use Cases
    - Final Say on Product Features

MEETINGS

- Release planning
  - Use Cases, Scenarios
  - Time Boxed estimate (backlog)
- Sprint (iteration) planning
  - Capacity
  - Points (1, 2, 3, 5, 8, 13, 20)
  - Velocity
- Daily standup
  - What did you do
  - What will you do
  - Are you blocked?

MORE HELPFUL JARGON

- **PRODUCT BACKLOG**
  - Backlog Grooming
  - Velocity
  - Burn down
- **USER STORY (US)**
  - Tasks (TA)
  - Acceptance Criteria (AC)
- **FOR PROGRAMMERS**
  - Pair programming
  - Refactor
  - Demos
- **FOR WRITERS**
  - Capacity per scrum team
  - Definition of Done (DoD)

WHAT WORKS IN AGILE SOFTWARE DEVELOPMENT

- Agile prevents the spec monster approach,
  which creates bloat ware or phantom ware
- Agile prevents "turn the prototype into a product"
  by embracing that approach
- Changing needs and markets are no problem
  Add or remove iterations to meet change
- Bug backlog is addressed early and often
  preventing unplanned delays in product release

WHAT'S DIFFERENT FOR THE WRITER IN AGILE SOFTWARE DEVELOPMENT

- **The Writer needs to:**
- Understand the big picture during release planning
- Understand features from US not from a functional spec
- Estimate your own effort required, in points or hours
- Communicate to others on your team why it takes that long
- Test the feature and document it within the same iteration
- Write just enough

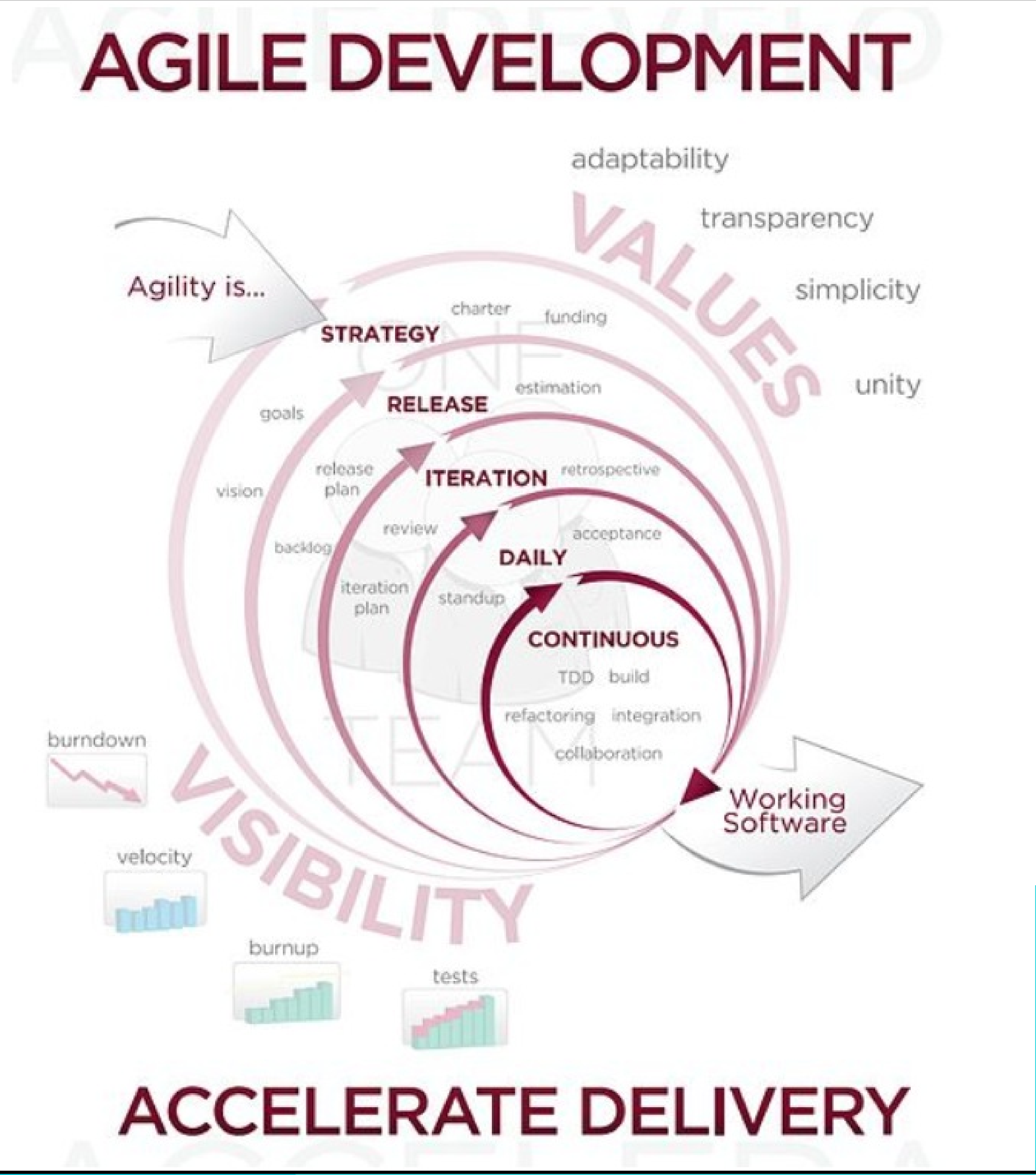
FIGHT CLUB PHILOSOPHY

The most important thing to know about Agile methods or processes is that there is no such thing. There are only Agile teams. The processes we describe as Agile are environments for a team to learn how to be Agile.

from *Agile Software Development: a gentle introduction.*

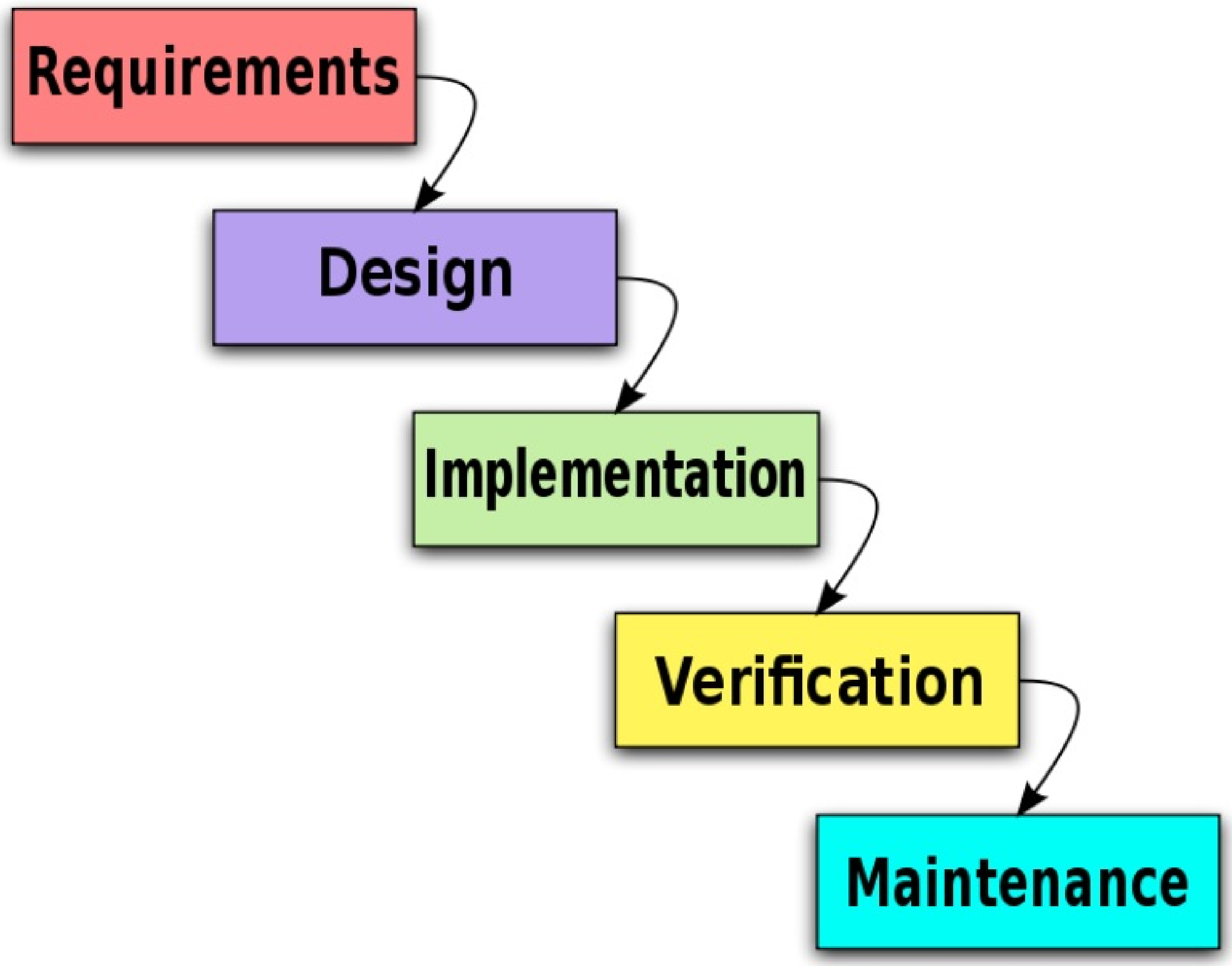
EVERYTHING ELSE YOU NEED TO KNOW ABOUT AGILE SOFTWARE DEVELOPMENT

- Yeah, it's on Wikipedia....this is what you'll see


AGILE DEVELOPMENT
Agility is...
adaptability
transparency
simplicity
unity
VALUES
STRATEGY
charter
funding
RELEASE
goals
estimation
vision
ITERATION
release plan
retrospective
review
backlog
acceptance
DAILY
iteration plan
standup
CONTINUOUS
TDD
build
refactoring
integration
collaboration
ONE TEAM
Working Software
burndown
velocity
burnup
tests
VISIBILITY
ACCELERATE DELIVERY

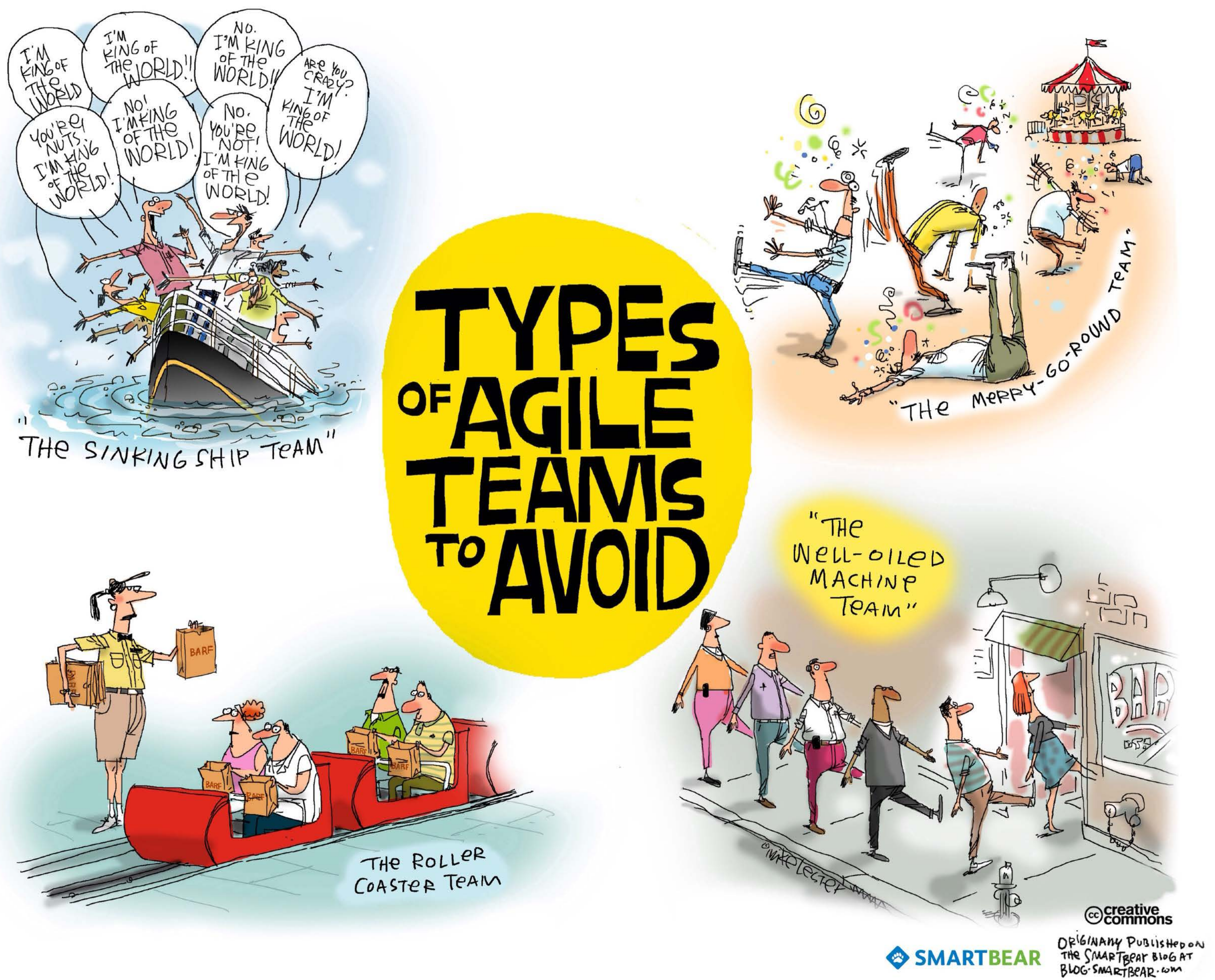
THE WATERFALL

(OK to forget)



LESSONS LEARNED WHEN REPLACING WATERFALL WITH AGILE

- Management buy-in is critical
- Implementation depends on company culture and varies by the personalities of the participants
- Multiple scrum juggling is the norm for writers


TYPES OF AGILE TEAMS TO AVOID
I'M KING OF THE WORLD
I'M KING OF THE WORLD!!
NO. I'M KING OF THE WORLD!!
ARE YOU CRAZY? I'M KING OF THE WORLD!
YOU'RE NUTS. I'M KING OF THE WORLD!
NO! I'M KING OF THE WORLD!
NO. YOU'RE NOT! I'M KING OF THE WORLD!
"THE SINKING SHIP TEAM"
"THE MERRY-GO-ROUND TEAM"
BARF
THE ROLLER COASTER TEAM
"THE WELL-OILED MACHINE TEAM"
BAR
©MIKELESTER
creative commons
SMARTBEAR
ORIGINAny PUBLISHED ON THE SMARTBEAR BLOG AT BLOG.SMARTBEAR.COM

AGILE SOFTWARE DOCUMENTATION

(Hi Honey, I joined a cult!)

AGILE SOFTWARE DOCUMENTATION MANIFESTO

(just my manifesto, so far)

- Documentation is not needed for discoverable tasks
- Simple is only good when it is complex enough to accomplish work
- Complex tasks need simple documentation
- Documentation is part of the process of users doing their work, it is not *the* process (communication not documentation).
- Write the documentation you would like to read
- Occam's razor, make it as simple as possible, but no simpler
- Build documents from the center outward, as time allows. Use the software, take notes, flesh it out. Do not write the outline and fill in with content.

PRINCIPLES OF AGILE SOFTWARE DOCUMENTATION

(IMHO)

- Domain experience is key
- Product owners define use cases (*purpose*)
- Uses cases dictate documentation needs (*message*)
- Persona or role-based docs (*audience*)

BEST PRACTICES FOR INCREASING THE AGILITY OF DOCUMENTATION

FROM *AGILE MODELING* WEBSITE BY ERIC REIS

"IDEALLY, AN AGILE DOCUMENT IS JUST BARELY GOOD ENOUGH, OR JUST BARELY SUFFICIENT, FOR THE SITUATION AT HAND."

In General

- Treat documentation like a requirement
- Require people to justify documentation requests

Writing

- Prefer executable specifications over static documents
- Document stable concepts, not speculative ideas
- Generate system documentation

Simplification

- Keep documentation just simple enough, but not too simple
- Write the fewest documents with least overlap
- Put the information in the most appropriate place
- Display information publicly

BEST PRACTICES FOR INCREASING THE AGILITY OF DOCUMENTATION

FROM *AGILE MODELING* WEBSITE BY ERIC REIS (CONT'D)

Determining What to Document

- Document with a purpose
- Focus on the needs of the actual customers(s) of the document
- The customer determines sufficiency

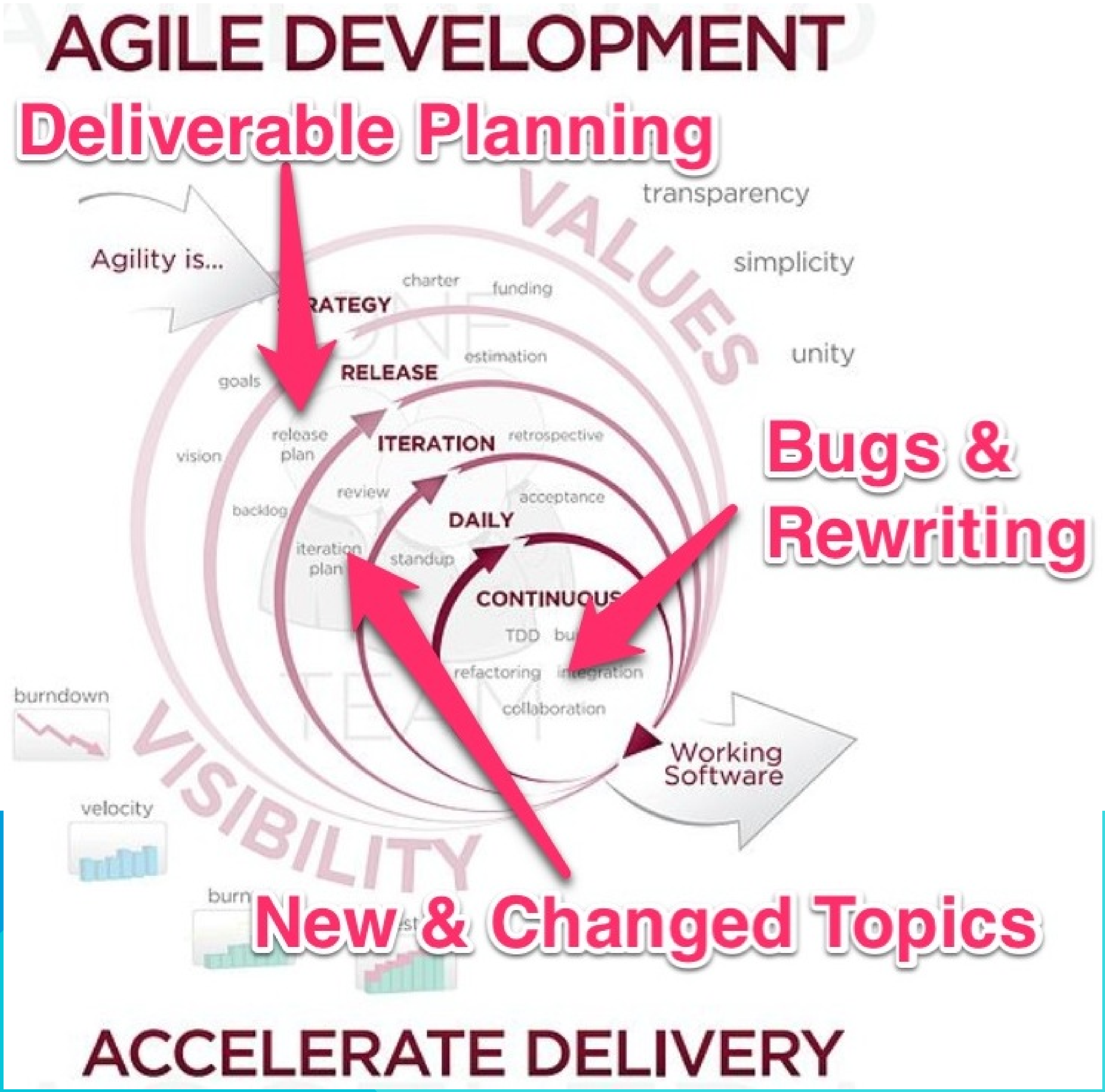
Determining When to Document

- Iterate, iterate, iterate
- Find better ways to communicate
- Start with models you actually keep current
- Update only when it hurts

HOW TO DO AGILE DOCUMENTATION

(IMHO)

- Use cases, persona based tasks, focus on the customer
- Collaborative reviews
- Product owners trump engineers
- Iterative writing, don't hold up the user story
- Rewrite during the hardening iteration and have it reviewed
- Rewriting *is* iterative
- Write overview info last
- Integrate doc planning with Agile planning


AGILE DEVELOPMENT
Deliverable Planning
Agility is...
VALUES
transparency
simplicity
unity
STRATEGY
charter
funding
RELEASE
estimation
goals
release plan
vision
ITERATION
retrospective
review
backlog
DAILY
acceptance
iteration plan
standup
CONTINUOUS
TDD
refactoring
integration
collaboration
ONE TEAM
Bugs & Rewriting
burndown
velocity
burn
VISIBILITY
Working Software
New & Changed Topics
ACCELERATE DELIVERY

BEST PRACTICES

- Do documentation late in the iteration or in the next iteration
- Depends on DoD: Is documentation part of AC?
- Go from simple to complex: reference, then procedures.
- Save conceptual till hardening iterations.
- Have a documentation scrum team and US for doc tasks unrelated to new feature development.

WHAT WORKS IN AGILE SOFTWARE DOCUMENTATION

- **Minimalism** (that's a whole 'nother slide show)
- **Flexibility** (really, do you need slides for that?)
- **Collaborative reviews** of small amounts of documentation (see next slide)
- **Rewriting** (it's coming too!)

COLLABORATIVE REVIEWS

- Online is best. For example, Acrobat Shared Review
- Advantages include speed, time shifting, ease of use and ease of including additional reviewers
- Fits iterative process. Many small reviews are better than one large one
- Feedback cycle of reviewers commenting and reacting to other reviewers comments. Positive feedback cycle, in theory

REWRITING (THEORIES)

"Good writing is writing you've rewritten"

Professor Frederick Shook, Colorado State University

"Before you can rewrite it, you have to write it"

Me

"The only thing stronger than the sex drive

is the desire to edit other people's work"

Mark Twain

REWRITING (STRATEGIES)

- Use collaborative reviews to rewrite

- Use "Flush text" to flush out the truth. This provides the writing to rewrite

FROM A WRITER'S GUIDE TO SURVIVING AGILE SOFTWARE DEVELOPMENT (WEB SITE)
BY GAVIN AUSTIN & MYSTI BERRY

Write fiction

Learn to feel comfortable writing documentation for products you can't test yet. You’re more likely to meet deadlines by writing fiction than by waiting to write nonfiction for a finished product.

Revise fiction

Learn to revise any fiction before it ships to customers. Insert revision reminders in documents and add revision reminders to your calendar.

WHAT DOESN'T WORK IN AGILE ENVIRONMENTS

- Large doc reviews
- Writing top down from concepts to procedures to reference
- Over-planning

CHALLENGES FOR AGILE TECHNICAL WRITERS

- Distributed teams
- Multiple scrums, capacity, how to estimate
- Determining doc needs, using UI to determine complexity
- Estimating hours, points for each TA or US

THE FUTURE OF AGILE DOCUMENTATION

Photo copyright San Marcos Mercury

Future Opportunities

- Better tool integration?
- More success stories
- Well defined Best Practices

FAQS

Q: Do I need my Gantt chart?

A: No, you can put it away now.

Other questions?

RESOURCES

Manifesto for Agile Software Development	http://agilemanifesto.org/
Agile/Lean Documentation: Strategies for Agile Software Development	http://www.agilemodeling.com/essays/agileDocumentation.htm
Agile Software Development: A gentle introduction	http://www.agile-process.org/
A Writer's Guide to Surviving Agile Software Development	http://www.scrumalliance.org/articles/369-a-writers-guide-to-surviving-agile-software-development
Agile Writer Girish Mahadevan	www/gpod.in

SOFTWARE

- Rally Software (private co., web based)
- Team Foundation Server (Microsoft, Visual Studio based)
- Jazz Agile (IBM, web based)
- XPlanner, (open source, web based)
- TargetProcess (private co., Windows and web)