BALANCING THE SPEED OF AGILITY WITH TECHNICAL QUALITY

Johnny D. Morgan, PhD
2017 Project Management Symposium
Balancing the Speed of Agility with Technical Quality

Johnny D. Morgan, PhD
Overview of the Presentation

• Agile software development practices provide an opportunity to reduce project stress. However, they can invoke new stresses in the area of technical quality

• Agenda
  – Describe Typical Progressive Warning Signs of Project Stress
  – Discuss the Application of Agile Practices to Reduce Project Stress
  – Define the Term “Technical Debt”
  – Examine the Manifesto for Agile Software Development
  – Discuss Six Areas For Technical and Programmatic Focus to Maintain Technical Quality
  – Present Conclusions

• Companion Symposium Paper is Available
Why Listen to Me?

• **Disclaimer:** This presentation represents my views, not the views of the multiple companies and customers I have supported

• However, it is based on 37 years of experience working on multiple projects that delivered capabilities to organizations
  – Large world-wide programs for major DoD, IC, and Federal customers
  – Multiple Government customers who want to implement commercial practices to deliver capabilities
  – Multiple small systems, both standalone and integrated capabilities
  – Many different development and delivery models

• **I am not:**
  – An agile proponent
  – An agile opponent

*Agile practices are a set of tools that I have in my engineering and project management toolbox!*
Typical, Progressive Warning Signs of Project Stress

- This pattern is based on anecdotal evidence

**Good Project**

- **Schedule Dates Missed**
  - Potential Causes:
    - Poor Initial Estimate
    - Long range commitment allows people to be pulled for more near-term tasking
    - Worker procrastination
    - Inadequate worker skills and/or performance
    - Work impediments not identified and resolved in a timely manner

- **Cost Overruns Occur**
  - Potential Causes:
    - More time required means more labor hours expended
    - More people added to the project to recover schedule increases communications and collaboration overhead

- **Technical Scope and Quality Suffer**
  - Potential Causes:
    - Technical scope is reduced to get project back within schedule and cost boundaries
    - Quality of remaining work scope is reduced to get the project delivered within schedule and cost boundaries

**Failed Project**
Agile Practices Can Reduce Project Stress

• Reduces Commitment to a Single Release
  – Fewer unknowns, thus it is easier to estimate
  – Further sub-divided to allow better progress assessment of coding effort:
    • Iterations/Sprints (demonstrated outcomes)
    • Features/Stories (daily tote boards)

• Prioritize the Requirements and Features at the Beginning:
  – Deliver most value-added items first, defer lesser value-added items to later
  – Allows the project to adjust to budget adjustments/cost overruns

• Daily Progress Measurement
  – Prevents procrastination
  – Prevents worker from being pulled to support other activities
  – Identified and resolves problems earlier
  – Enhances both individual and team-wide accountability
Where Does Quality Fit In?

Both Models Assume that Quality Is Fixed, Which is Rarely the Case!

Legend

= Variable

= Fixed

Definition of Technical Debt

• Technical Debt was first coined as a metaphor for the trade-off between writing clean code at higher cost and delaying the delivery and writing messy code cheap and fast at the higher cost of maintenance efforts once it’s shipped. (Cunningham, 1992)

• It is similar to financial debt as it supports quick development at the cost of compound interest to be paid later. The longer you wait to garden the design and code, the larger the amount of debt. (Buschmann, 2011a)

“Software projects often cut corners in the rush to meet deadlines, resulting in bad code... Taking short-cuts means that the next time the software is touched, it needs to be fixed before any further work can be done. So the second piece of work is even more expensive to do correctly than the original piece of work, and you can be sure that the deadlines are just as tight as they were the first time.” (Brazier, 2007)
# Managing Technical Debt

<table>
<thead>
<tr>
<th>Strategies for Managing Technical Debt (Buschmann, 2011a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debt Repayment</td>
</tr>
<tr>
<td>Debt Conversion</td>
</tr>
<tr>
<td>Interest Payment</td>
</tr>
<tr>
<td>Debt Retirement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methodologies for Gardening Systems (Buschmann, 2011b,c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refactoring</td>
</tr>
<tr>
<td>Reengineering</td>
</tr>
<tr>
<td>Rewriting</td>
</tr>
</tbody>
</table>
Examining the Agile Manifesto

Manifesto for Agile Software Development:
We are uncovering better ways of developing software by doing it and helping others do it. Through this we have come to value:

<table>
<thead>
<tr>
<th>Individuals and Interactions</th>
<th>Over Processes and Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Software</td>
<td>Over Comprehensive Documentation</td>
</tr>
<tr>
<td>Customer Collaboration</td>
<td>Over Contract Negotiation</td>
</tr>
<tr>
<td>Responding to Change</td>
<td>Over Following a Plan</td>
</tr>
</tbody>
</table>

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

Principles behind the Agile Manifesto: We follow these principles:

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our highest priority is to satisfy the customer through early and continuous delivery of valuable software</td>
<td></td>
</tr>
<tr>
<td>Business people and developers must work together daily throughout the project</td>
<td></td>
</tr>
<tr>
<td>Working software is the primary measure of progress</td>
<td></td>
</tr>
<tr>
<td>Simplicity – the art of maximizing the amount of work not done – is essential</td>
<td></td>
</tr>
<tr>
<td>Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage</td>
<td></td>
</tr>
<tr>
<td>Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.</td>
<td></td>
</tr>
<tr>
<td>Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely</td>
<td></td>
</tr>
<tr>
<td>The best architectures, requirements, and designs emerge from self-organizing teams</td>
<td></td>
</tr>
<tr>
<td>Deliver software frequently, from a couple of weeks to a couple of months, with a preference to a shorter timescale</td>
<td></td>
</tr>
<tr>
<td>The most efficient and effective method of conveying information to and within a development team is face-to-face conversation</td>
<td></td>
</tr>
<tr>
<td>Continuous attention to technical excellence and good design enhances agility</td>
<td></td>
</tr>
<tr>
<td>At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly</td>
<td></td>
</tr>
</tbody>
</table>
Examining the Agile Manifesto (Six Discussion Areas)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this, 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 behind the Agile Manifesto: We follow these 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 software frequently, from a couple of weeks to a couple of months, with a preference to a shorter timescale.
- 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.
- Business people and developers must work together daily throughout the project.
- Continuous attention to technical excellence and good design enhances agility.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- Simplicity – the art of maximizing the amount of work not done – is essential.
Examining the Agile Manifesto (Defining Done)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto:** We follow these principles:

1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**
2. **Business people and developers must work together daily throughout the project.**
3. **Working software is the primary measure of progress.**
4. **Simplicity – the art of maximizing the amount of work not done – is essential.**
5. **Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.**
6. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**
7. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**
8. **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**
9. **Continuous attention to technical excellence and good design enhances agility.**
10. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**
Trading Cost and Quality


Real World Experiences:

- Deploy on Thursday; Fix on Friday
- For each release, we only deployed the software modules we changed, thus we would have to go through 60+ releases to rebuild the system
- In 2011 the performance was great, but in 2015 it takes 30 seconds to respond to a button click
- The only people who really knows how the system operates, just left the program
Recommendations

• At the start of the project, identify a “starting methodology”
  – Try to “get close” to the balancing point, which is different for each project
    • Don’t use all the processes and activities “just to be safe”
    • Don’t use pre-established tailored baselines
  – Future retrospectives should adjust this balancing point

• Write a definition of done for the following:
  – Feature/Story
  – Iteration/Sprint
  – Release

• If you don’t meet the definition of done, you are incurring technical debt which must then be factored into your future iteration/sprint and release backlogs
Examining the Agile Manifesto (Defining the Customer)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto:** We follow these principles:

- **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**
- **Business people and developers must work together daily throughout the project.**
- **Working software is the primary measure of progress.**
- **Simplicity – the art of maximizing the amount of work not done – is essential.**
- **Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.**
- **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**
- **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**
- **The best architectures, requirements, and designs emerge from self-organizing teams.**
- **Deliver software frequently, from a couple of weeks to a couple of months, with a preference to a shorter timescale.**
- **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**
- **Continuous attention to technical excellence and good design enhances agility.**
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**
Highest priority is to satisfy “the customer” through continuous delivery of valuable software

The Customer

- People/organizations that focus on corporate value through business continuity, reputation, competitive position in the market place (5 hour outage costs $150M)
- People/organizations that obtain value from the system (travelers on an airline)
- People who use the system to deliver value (airline traveler agents, gate clerks, baggage handlers, customer service reps)
- People who keep the system operational such that value can be continuously delivered (computer support technicians)
- People who will develop new or updated capabilities that provide more value to the organization (future developers)

How are these “value perspectives” represented in your agile team?”
Examining the Agile Manifesto (Architecture)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto: We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage
3. Deliver software frequently, from a couple of weeks to a couple of months, with a preference to a shorter the timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
11. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage
12. Deliver software frequently, from a couple of weeks to a couple of months, with a preference to a shorter the timescale
13. The best architectures, requirements, and designs emerge from self-organizing teams
14. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly
Agility versus Architecture

Agile Perspective:
- Architectures emerge from self-organizing teams
- Be resolutely adaptive: respond only when changes occur
- Perceive architecture as Big Up-Front Design (BUFD) with You Ain’t Gonna Need it (YAGNI) features
- Focused on functional requirements

Architecture Perspective:
- Architectures naturally seek the maximum level of complexity (Law of Entropy)
- Anticipate change, plan for them, and isolate their impact
- Small successive refactoring through each sprint is insufficient
- Focused on both functional, and non-functional requirements (Reliability, Availability, Maintainability, Performance, Information Security, etc.)

How Can We Integrate These Different Perspectives?
Defining Architecture

Architecture: Encompasses the set of significant decisions about the structure and behavior of the system. These decisions prove to be the hardest to undo, change and refactor (Abrahamsson et al., 2010)

Try to Interleave architectural stories and functional stories
Reasons Why Architectures Matter

1. They enable designers to document assumptions and understand the structure of their creation. Architects use architectural patterns that leverage reusable solutions to commonly recurring problems.

2. Architectures provide the ability to establish design baselines and perform trade studies or "what-if" analyses.

3. Architectures help clarify details at the micro and macro level:
   - End users might be interested in details that affect implementation, efficiency, and security.
   - Owners may be interested in the business requirements and environment contexts of a system that change over time, such as legal, social, financial, competitive, and technology concerns.

4. Architectures allow reuse of components between projects.

5. Architectures provide insight into the maintainability of the product (Valerdi, 2014).
Examining the Agile Manifesto (Communications)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto:** We follow these principles:

- **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**
- **Business people and developers must work together daily throughout the project.**
- **Working software is the primary measure of progress.**
- **Simplicity – the art of maximizing the amount of work not done – is essential.**
- **Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.**
- **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**
- **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**
- **The best architectures, requirements, and designs emerge from self-organizing teams.**
- **Deliver software frequently, from a couple of weeks to a couple of months, with a preference to a shorter timescale.**
- **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**
- **Continuous attention to technical excellence and good design enhances agility.**
- **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**
Synchronous Communications
Enriched with Non-Verbal Elements
Low Latency Feedback

Communications = \( N \times (N-1) \)
Pathways = 2
Knowledge
Experience
Perceptions

Transfer Results
To Explicit Formats

Group Communications
Good: Diversity of Information
Builds Trust
Promotes Learning

Bad: Overbearing People
Mediation Required

Post Meeting
- Thoughts/Reflection
- Research

Memory
Personality
Culture/Language

Knowledge
Experience
Perceptions

Memory
Personality
Culture/Language

Knowledge
Experience
Perceptions
Differentiating Learners and the Knowledge They Seek


<table>
<thead>
<tr>
<th>Cognition Dimension</th>
<th>Knowledge Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Factual</td>
</tr>
<tr>
<td>Experts</td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>[Documentation]</td>
</tr>
<tr>
<td>Evaluate</td>
<td></td>
</tr>
<tr>
<td>Practitioners</td>
<td></td>
</tr>
<tr>
<td>Analyze</td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td></td>
</tr>
<tr>
<td>Novices</td>
<td></td>
</tr>
<tr>
<td>Understand</td>
<td></td>
</tr>
<tr>
<td>Remember</td>
<td></td>
</tr>
</tbody>
</table>

Even Experts are Novices when they initially join an existing project!
Exchanging the Agile Manifesto (Documentation)

**Manifesto for Agile Software Development:**

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto:** We follow these principles:

<table>
<thead>
<tr>
<th>Principle</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our highest priority is to satisfy the customer through early and</td>
<td>Continuous delivery of valuable software</td>
</tr>
<tr>
<td>continuous delivery of valuable software</td>
<td></td>
</tr>
<tr>
<td>Welcome changing requirements, even late in development. Agile processes</td>
<td>Build projects around motivated individuals. Give them the environment</td>
</tr>
<tr>
<td>harness change for the customer’s competitive advantage</td>
<td>and support they need, and trust them to get the job done.</td>
</tr>
<tr>
<td>Deliver software frequently, from a couple of weeks to a couple of</td>
<td>The most efficient and effective method of conveying information to</td>
</tr>
<tr>
<td>months, with a preference to a shorter the timescale</td>
<td>and within a development team is face-to-face conversation</td>
</tr>
<tr>
<td>Continuous attention to technical excellence and good design enhances</td>
<td>Agile processes promote sustainable development. The sponsors, developers,</td>
</tr>
<tr>
<td>agility</td>
<td>and users should be able to maintain a constant pace indefinitely</td>
</tr>
<tr>
<td>At regular intervals, the team reflects on how to become more effective,</td>
<td>Continuous attention to technical excellence and good design enhances</td>
</tr>
<tr>
<td>then tunes and adjusts its behavior accordingly</td>
<td>agility</td>
</tr>
<tr>
<td>Simplicity – the art of maximizing the amount of work not done – is</td>
<td>The best architectures, requirements, and designs emerge from</td>
</tr>
<tr>
<td>essential</td>
<td>self-organizing teams</td>
</tr>
<tr>
<td>The best architectures, requirements, and designs emerge from</td>
<td></td>
</tr>
<tr>
<td>self-organizing teams</td>
<td></td>
</tr>
</tbody>
</table>
Thoughts on Documentation

• The code is the truth, but not the whole truth.
• Large architectural decisions can not be discerned from the code itself
  – The results of these decisions are scattered throughout the code
  – Their meaning and presence are in the heads of the code’s creators and not easily evident by staring at the code (Booch, 2011)
• In agile software development, the code changes often, and thus keeping the lowest level documentation in-line with the code is difficult (almost futile)
• “The architect’s and designer’s job is not to pass along ‘the design’ but to pass along ‘the theories’ driving the design....
  – This latter goal is more useful and more appropriate....
  – Knowledge of the theory is tacit in the owning, so passing along the theory requires passing both explicit and tacit knowledge” (Cockburn, 2007)

*Document those items that helps the next programmer build an adequate theory of the system/program*
Recommendations on Documentation Artifacts

- Change the mindset from “document” to “artifact”
  - A collection of knowledge, irrespective of the media that contains it

- Collect and maintain the following:
  - Knowledge that defines to users, operators, and maintainers how to operate and maintain the system
  - Knowledge that describes how to rebuild and redeploy the system should a problem or disaster occur.
  - Knowledge that allows future personnel to modify the system over its life cycle (The theory of the system)
Examining the Agile Manifesto (People)

Manifesto for Agile Software Development:

We are uncovering better ways of developing software by doing it and helping others do it. Through this 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 behind the Agile Manifesto:** We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Business people and developers must work together daily throughout the project
- Working software is the primary measure of progress
- Simplicity – the art of maximizing the amount of work not done – is essential
- Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- 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
- 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
*Reprinted from Takpuie & Tanner. (2014), Investigating the characteristics needed by scrum team members to successfully transfer tacit knowledge during agile projects, Journal of Systems Evaluation, 19(1)
Conclusions

• Projects can experience a progressive set of warning signs indicating project stress.

• Agile methods can address these warning signs:
  – Only commit to a release and break that into visible iterations/sprints and then to feature/stores.
  – Prioritize requirements at the beginning and work the highest value-added items first.
  – Daily standups induce productivity and accountability.

• Because agile methods assign time-boxes for delivery, an inherent risk is the introduction and accumulation of technical debt.
Conclusions

• To ensure technical quality is not reduced, the following recommendations are provided:
  – Provide a definition of done for features/stories, iterations/sprints, and releases.
  – Expand the definition of “customer” to ensure that non-functional requirements and long-term project life cycle concerns are addressed.
  – Incorporate architecture concerns into the project such that non-functional and future functional requirements can be anticipated.
  – Communication mechanisms must incorporate both tacit and explicit knowledge transfer mechanisms.
    • Innovative teams effectively implement both mechanisms.
  – When developing documentation, focus on those artifacts that provide others with the architectural (theory) constructs of the system.
    • The software code does not represent the theory of the system but future developers need this awareness as they continue to modify the system.
  – Agile methods place significant reliance on the skills, qualities, and talents of the individuals that comprise the team. A model is provided to assist in identifying and selecting people for inclusion in agile projects.
Contact Information

Johnny D. Morgan, PhD
INCOSE ESEP, AWS CSA
PMI PMP, SAFe Agilist
General Dynamics Information Technology
johnny.morgan@gdit.com
References


