What did I do wrong? An episode on Agile Anti-Patterns

Supannika Mobasser, Brook Cavell,
Dan Ingold, Andrew Melnick,
Joanne Succari, and Eric Yuan

The Aerospace Corporation

Systems Engineering Forum
Digital Engineering In Practice
14 June 2022

Outline

- Introduction, Problem Statement, and Working Definition
- Examples of Anti-Patterns
- Agile Anti-Patterns on
  - *New Agile Projects*
  - *Product Backlog Management and Requirements Management*
  - *Architecture and Design*
  - *Testing*
  - *Software Development Team*
  - *Project Management*

Introduction

Problem Statement: In the course of adopting new systems, processes, and modalities, people sometimes do not utilize a process correctly. In doing so, they engage in a **practice or pattern** that they think will work, but if it is not applied correctly, the **practice or pattern** actually works against them in delivering product value, and may impede development progress.

A Pattern as defined by Merriam-Webster

- a form or model proposed for imitation: exemplar
- a reliable sample of traits, acts, tendencies, or other observable characteristics of a person, group, or institution
- a discernible coherent system based on the intended interrelationship of component parts

Colloquialism for this briefing is termed as an “Anti-Pattern”. We conducted a panel to discuss anti-patterns and this presentation was the product of that exercise.

Ref: https://www.merriam-webster.com/dictionary/pattern

Working Definition

For this exercise, we established consistent definitions of Anti-Patterns to help us characterize the boundaries to address the problem.

Anti-Pattern:

> *“ An anti-pattern is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.[Laplante 2005, Brown et al 2000]. The term, coined in 1995 by Andrew Koenig,[Koenig 1995] was inspired by a book, Design Patterns, which highlights a number of design patterns in software development that its authors considered to be highly reliable and effective.*
>
> *The term was popularized three years later by the book AntiPatterns, which extended its use beyond the field of software design to refer informally to any commonly reinvented but bad solution to a problem”*

Ref: Laplante, Phillip A.; Neill, Colin J. (2005). Antipatterns: Identification, Refactoring and Management. Auerbach Publications.
Ref: Brown, William J.; Malveau, Raphael C.; McCormick, Hays W.; Thomas, Scott W. (2000). Hudson, Theresa Hudson, ed. Anti-Patterns in Project Management. John Wiley & Sons.
Ref: Koenig, Andrew (March–April 1995). "Patterns and Antipatterns". Journal of Object-Oriented Programming. 8 (1): 46–48
Ref: Anti-pattern, https://en.wikipedia.org/wiki/Anti-pattern

Examples of Organizational and Project Management Anti-Patterns

Organizational

- **Analysis paralysis:** A project stalled in the analysis phase, unable to achieve support for any of the potential plans of approach
- **Groupthink:** A collective state where group members begin to (often unknowingly) think alike and reject differing viewpoints
- **Micromanagement:** Ineffectiveness from excessive observation, supervision, or other hands-on involvement from management
- **Mushroom management:** Keeping employees "in the dark and fed manure" (also "left to stew and finally canned")
- **Seagull management:** Management in which managers only interact with employees when a problem arises, when they "fly in, make a lot of noise, dump on everyone, do not solve the problem, then fly out“
- **Vendor lock-in:** Making a system excessively dependent on an externally supplied component

Project management

- **Cart before the horse:** Focusing too many resources on a stage of a project out of its sequence
- **Death march:** A project whose staff, while expecting it to fail, are compelled to continue, often with much overwork, by management which is in denial
- **Ninety-ninety rule:** Tendency to underestimate the amount of time to complete a project when it is "nearly done"
- **Overengineering:** Spending resources making a project more robust and complex than is needed
- **Scope creep:** Uncontrolled changes or continuous growth in a project’s scope, or adding new features to the project after the original requirements have been drafted and accepted (also known as requirement creep and feature creep)
- **Brooks’s law:** Adding more resources to a project to increase velocity, when the project is already slowed down by coordination overhead.

Anti-Patterns are generally caused by not knowing any better way in solving a particular type of problem.

Ref: https://en.wikipedia.org/wiki/Anti-pattern#Examples

Outline

- Introduction, Problem Statement, and Working Definition
- Examples of Anti-Patterns
- Agile Anti-Patterns on
  - *New Agile Projects*
  - *Product Backlog Management and Requirements Management*
  - *Architecture and Design*
  - *Testing*
  - *Software Development Team*
  - *Project Management*

Anti-Patterns on New Agile Projects (1/3)

- **Agile Theater: "Do Agile" vs "Being Agile"**
  - *Follow Agile ceremonies. But just a stage-play, no change behind the scenes.*
  - *Going through motions without understanding what intended outcomes should be*
    - Design review for each Sprint
    - Death in CDRLs
    - Fluid Sprints – no time box or extended Sprint length
    - Not planned well
    - Expectations that stories can move in and out or that Sprint stays open if work is not completed
  - ***Recommendation***
    - Be Agile, often need to change mindset or organizational culture
    - Follow Agile manifesto values
    - Understand the objectives and benefits before tailoring the Agile process
    - Study Agile lessons learned from other programs
    - Look for solutions that are documented, repeatable, and proven to be effective

CDRL – Contract Data Requirement List
Ref: Agile Anti-Patterns, https://www.devteams.at/2017/09/15/agile-anti-patterns.html

Anti-Patterns on New Agile Projects (2/3)

- **Scrum-but** : “we’re doing Scrum, but we…” [do something that is completely the opposite of what it says to do in Scrum]
  - *Extensive up-front design*
  - *Large user story to fit in a Sprint*
  - *Silo teams*
  - *Part-time team members*

- **Agile-on-the-fly:** start applying Agile quickly without thinking or careful planning
  - *Blindly follow practices of other programs*

- ***Recommendation***
  - *If teams are new to Agile, it’s recommended that they adopt it properly first*
  - *Need training, mindset change, full team (including management) buy-in, on-going coaching*

Anti-Patterns on New Agile Projects (3/3)

- **Assuming Agile planning is entirely ad hoc and as-you-go**
  - *Agile planning is intended to be flexible, but not chaotic*
  - *One program planned "Releases" but had only the vaguest notion of what those Releases would contain*
    - Resulting in lack of structure and priority in their Sprint planning
  - ***Recommendation***
    - Planning should be oriented to achieve a Minimum Viable Product or Minimum Operational Capability (focus on delivering features or capabilities rather than functional components) then enhance that capability incrementally in subsequent Sprints and Releases
    - Never be in a state where the product isn't working. Plan every iteration to enhance and deliver additional mission capabilities.
- **Not building in quality**
  - *Not enough testing, especially regression testing, preferably automated regression testing*
  - ***Recommendation***
    - Start with the end in mind, use acceptance criteria, definition of done
    - Embed testers as part of the development team

Anti-Patterns on the Product Backlog Management or Requirements Management (1/3)

- **Inefficient use of Time/effort**
  - *Not spending time during backlog management - impacts Sprint Planning efficiency*
  - *Too many items or items too old – clutter and difficult to prioritize*
  - *Review/estimate everything too early - unnecessary effort by team*
  - *No acceptance criteria on user story or unclear requirements before the Sprint starts*
  - *Too much information or acceptance criteria – leave room for discussion between product owner and the team for new perspectives and negotiation on scope; leaves team less engaged if everything is spelled out*
  - ***Recommendation***
    - Organize backlog grooming sessions to ensure that items are ready for next Sprint
    - Review/estimate the top priority items that are likely to be addressed in the next 1-2 Sprints

Anti-Patterns on the Product Backlog Management or Requirements Management (2/3)

- **Prioritization issues**
  - *Prioritization by proxy – someone else (external to Product Owner) dictates the priorities; no accountability of Product Owner*
  - *No backlog prioritization*
  - *Full prioritization upfront*
  - ***Recommendation***
    - Appoint a Product Owner with authority
    - Priorities should expect to adjust as you observe working software and assess how much is enough

Anti-Patterns on the Product Backlog Management or Requirements Management (3/3)

- **Failing to organize and prioritize the backlog by feature**
  - *Item is horizontal component versus end-to-end feature*
    - May lead to completed components, but not operational or not delivering mission value
- **Composition of a backlog item**
  - *Items not decomposed from Themes or Epics*
- **Assign a non-functional requirement to be developed in one Sprint**
  - *E.g., scalability requirement can not be achieved in one Sprint*

- ***Recommendation***
  - *Factor in or bake in non-functional requirements from day one*
  - *Start from requirements decomposition; Organize requirements in a capability-driven structure*
  - *Filter by an epic and see what the features have been defined and help to convey scope of a particular release*

Anti-Patterns on Architecture and Design

- **Architecture and Design are somewhat orthogonal to development methodologies**
  - *Agile doesn't work well with stovepipes or monolithic hardware/software systems*
  - ***Recommendation***
    - Modularity, layered architecture, abstracted dependencies
    - Small, self-contained, testable feature decomposition
    - Start with Just Enough Architecture
- **Several space and ground software are developed by engineers from other domains without software engineering background**
  - *Trade off between domain expert developers and professional software developers*
  - ***Recommendation***
    - Be cognizant of technical debt
    - Start by defining some principles, tenets and architectural decisions upfront
- **Lack of Government-owned software architecture**
  - *Risks of interoperability, sustainability*
  - ***Recommendation***
    - Define interface specifications

Anti-Patterns on Testing

- **Accepting manual testing as suitable and effective**
  - *Challenges with a fast-paced development*
  - ***Recommendation***
    - Design for test
    - Use Test-Driven Development at the unit level
    - Use Behavior-Driven Development at the integration level
    - Continue to enhance automated functional tests throughout the lifecycle
- **Accepting automated unit testing covers everything**
  - *Feasible, but take longer time, cost more, and difficult to maintain*
  - ***Recommendation***
    - Need a good balance between automated and manual testing
- **Not integrating Test with Development**
  - *Silo teams throw products over the wall*
  - ***Recommendation***
    - Test as you go, continual testing, automated regression testing

Anti-Patterns on the Software Development Team

- **Product Backlog Grooming done privately**
  - *Top-down management, less team involvement*
  - ***Recommendation***
    - Full team must be involved to ensure a shared understanding of the “why” and “what” since anyone on the team should be able to pick up a story and work on it
- **Fill Sprint Backlog with 100% of the team capacity**
  - *Lead to “I’m busy” attitude and no time to help others*
  - ***Recommendation***
    - Leave room for collaboration and team support
- **Pushing team to do extra work to finish the committed Sprint backlog items**
  - *Violate “sustainable pace”; overtime has a detrimental impact on productivity*
  - ***Recommendation***
    - Adjust team velocity in the next Sprint
- **Not allowing team to work on Technical Debt**
  - *Incremental impact on system performance, team productivity, testing quality*
  - ***Recommendation***
    - Set aside HIP (Hardening, Innovation and Planning) Sprint or buffer Sprint

Anti-Pattern on Project Management (1/2)

- **Planning failures**
  - *Pre-planning iterations as though using traditional planning (IMS)*
  - *Failing to plan releases in terms of expected features/capabilities*
  - *Creating a too-detailed IMS, and evaluating progress (and EV) against it*
  - ***Recommendation***
    - May continue to use high-level IMS
    - Use more of Product Roadmap and Architectural Runway
      - *Consider dependencies and constraints when developing Release Plan*

- **Team organization failures**
  - *Teams organized by functional decomposition, rather than by Feature*
  - ***Recommendation***
    - Encourage multi-disciplinary teams organized by Feature/Epic
    - Focus on flat organization structure to speed up decision-making process

EV – Earned Value; IMS – Integrated Master Schedule

Anti-Pattern on Project Management (2/2)

- **Tasks too large that it sits in "In Progress" the whole Sprint**
  - *"Yesterday I worked on X and today I plan to work on X"*
    - Team doesn't really know what you are doing or whether progress is being made
    - It's not apparent if you are blocked or stuck on a problem that possibly someone else can help with
    - Impacts dependent subtasks and likely the larger story
  - ***Recommendation***
    - Scrum master should pay attention to progress, shared vision, and potential impediments
- **Ineffective retrospective**
  - *No action or follow-up taken to remedy issues*
    - Repetition of issues, deteriorating morale
  - *Team less likely to raise concerns if they feel nothing will change*
  - ***Recommendation***
    - Team should identify actionable, measurable, and controllable items
    - Put action items in the backlog
    - Review past action items with the team

Conclusions

- No one-size-fits-all solutions
- Understand the practice/patterns you are adopting
- Study lessons learned from other programs
- Perform continuous improvement

References

- Definition of Pattern from Merriam-Webster: https://www.merriam-webster.com/dictionary/pattern
- Definition from Agile Scrum Alliance *https://www.agilealliance.org/glossary/antipattern*
- Antipattern catalog: http://wiki.c2.com/?AntiPatternsCatalog
- Wiki's antipattern page: https://en.wikipedia.org/wiki/Anti-pattern#Examples
- InfoQ list of Agile specific Antipatterns https://www.infoq.com/news/2016/03/agileindia-7sins-scrum
- The Agile Antipattern Project: http://www.agileantipatterns.com/ , collection of Agile specific antipatterns including a "Go Fish" option for various random antipatterns
- Scrum-but and Agile anti-patterns *https://www.extremeuncertainty.com/scrum-but-and-agile-anti-patterns/*
- Sprint Planning Anti-Patterns *https://age-of-product.com/scrum-sprint-planning-anti-patterns/*
- Product Owner Anti-Patterns You Should Be Aware Of *https://www.knowledgehut.com/blog/agile/product-owner-anti-patterns-should-be-aware-of*
- A Koenig (March–April 1995). "Patterns and Antipatterns". Journal of Object-Oriented Programming. 8 (1): 46–48
- D J Gullo, Real World Agility: Practical Guidance for Agile Practitioners, Addison-Wesley Professional; 1 edition (July 25, 2016)
- D Leffingwell, A talk regarding SAFe 4.5 by Dean Leffingwell from O'Reilly: Leading SAFe Scaled Agile Framework 4.5 Live Lessons
- M C Layton and S J Ostermiller, Agile Project Management For Dummies, For Dummies; 2 edition (September 5, 2017)
- M E Moreira, The Agile Enterprise: Building and Running Agile Organizations, Apress; 1st ed. edition (March 16, 2017)
- S W Ambler and M Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press; 1 edition (May 31, 2012)
- P A Laplante; Neill, Colin J. (2005). Antipatterns: Identification, Refactoring and Management. Auerbach Publications.
- W J Brown; Malveau, Raphael C.; McCormick, Hays W.; Thomas, Scott W. (2000). Hudson, Theresa Hudson, ed. Anti-Patterns in Project Management. John Wiley & Sons.