Scaled Agile Framework® 6.0

28 Public Sector Anti-Patterns and Antidotes

SAFe®
PROVIDED BY SCALED AGILE

“To thrive in the digital age you need business agility.

This new way of working requires a new mindset, values, principles, and practices.”

—**Dean Leffingwell**, Creator of SAFe

What is the Scaled Agile Framework (SAFe)?

- SAFe is a goal-driven framework to rapidly create innovatively new products and services.
- SAFe is the world's leading lean and agile thinking operating framework for enterprise agility.
- SAFe integrates the power of Lean, Agile, Design Thinking, DevSecOps, AI, and Cloud.
- SAFe is a comprehensive operating system that helps enterprises thrive in the digital age.
- SAFe helps to deliver innovative high-quality products and services fast and predictably.
- SAFe helps enterprises, portfolios, large solutions, and product and service teams improve, grow, and respond to change.
- SAFe is used by more than 20,000 international enterprises and more than 1,000,000 practitioners have been trained and certified.
- SAFe is adaptable to product and services ranging from one to hundreds of agile teams.
- SAFe continuously improves on regular basis to ensure enterprises are on the leading edge.
- **SAFe has a large ecosystem of products, services, training, tools, metrics, workers, etc.**

Footnote. *SAFe is often misapplied to manage people like high-precision scope-driven manufacturing systems*.

28 Public Sector Anti-Patterns and Antidotes of SAFe 6.0

1. Scope-Driven Contracts
2. Big Up Front Requirements
3. Buyer-Supplier Inequality
4. Supplier Dominance
5. Horizontal Solution Slicing
6. No Technology Visionaries
7. Outdated Legacy Systems
8. Outdated Development Tech
9. Long Lead Hardware Items
10. Feature Factory Sweatshops
11. Integrated Master Schedules
12. Seeking Full Utilization
13. Manufacturing Statistics
14. Overmeasuring Performance
15. Using Tasks vs. Stories
16. No Architectural Runways
17. No Agile Product Management
18. Buyer-Supplier Role Division
19. Supplier Only Agile Roles
20. Uneven Team Resource Loading
21. Death by Marathon Meetings
22. Lack of Process Tailoring
23. Complex or Expensive ALMs
24. Can't Think "Out of the Box"
25. No "One Team" Culture
26. Treating People Like Automatons
27. Big Bang Testing Events
28. Independent Manual Testing

1

Scope-Driven Contracts

Result

- Too much WIP
- Late, overbudget
- Poor performance

Anti-Pattern

- Specifying the concise scope of a contract in SOW, business requirements documents, design specifications, or agile backlogs (i.e., epics, features, stories, etc.).

Antidote

- Specify the Lean-Agile “process” framework to be used.
- Specify practices, roles, and responsibilities to be used.
- Let the scope of the contract emerge from the framework.

Example

- Allow the product or service scope emerge from road-mapping, lean-agile product management, quarterly planning, iteration planning, etc.

2

Big Up Front Requirements or Modeling Teams

Result

- → Too much WIP
- → Late, overbudget
- → Poor performance

Anti-Pattern

- → Forming a full-time requirements or systems modeling team to create requirements, specifications, system models, wireframes, UX models, etc.

Antidote

- → Apply lean-agile product management best practices.
- → Apply Lean UX thinking principles, practices, and methods.
- → Design just-enough, just-in-time business experiments.

Example

- → Use one-week design sprints with minimalistic Lean UX models in dual-agile development cycles for each major business experiment, quarter, or other planning interval.

3

Buyer-Supplier Inequality

Result

- Suboptimal decisions
- Poor morale, attrition
- Poor performance

Anti-Pattern

- Establishing high power-distance between buyers and suppliers by designating public servants to function as leaders and managers and suppliers as silent developers.

Antidote

- Form a badgeless “one-team” culture and team-of-teams.
- Allow suppliers to be leaders and buyers to be developers.
- Maximize collaboration, participation, and transparency.

Example

- Allow suppliers to have key roles and responsibilities in the lean-agile framework, integrate product and process events, and share decision-making rights and privileges.

4

Supplier Dominance

Result

- Suboptimal decisions
- Very poor solution quality
- Substitution, corner cutting

Anti-Pattern

- Delegate all decision-making rights, power, status, roles, and responsibilities to the supplier or lead integrator (to the exclusion of public sector governance bodies and teams).

Antidote

- Form governance bodies with both buyers and suppliers.
- Ensure everyone is trained in the lean-agile framework.
- Empower joint governance bodies to make key decisions.

Example

- Form solution and product management and engineering teams from both buyers and suppliers, which includes development value streams and teams-of-teams.

5

Horizontal Solution Slicing

Result

- → Suboptimal decisions
- → Poor synchronization
- → Chronic inconsistency

Anti-Pattern

- → Forming external and internal development value streams, teams-of-teams, and agile teams along horizontal layers (i.e., user interface, middleware, backend, etc.).

Antidote

- → Specify vertically sliced MVPs that cross solution layers.
- → Form development value streams around vertical MVPs.
- → Form just-in-time architectural runways and enabler MVPs.

Example

- → Form solution and product management and engineering MVP teams from both buyers and suppliers, which vertically span user interfaces, middleware, backend, etc.

6

Technology Visionaries are Absent

Result

- Too much WIP
- Poor morale, attrition
- Very poor solution quality

Anti-Pattern

- Forming scope, cost, and performance-driven leadership teams to implement change orders to modernize an old outdated legacy system, data center, or technology stack.

Antidote

- Hire and appoint a small team of technology visionaries.
- Seek to form a modern technological solution on purpose.
- Focus on a near-term highly user-centered software MVP.

Example

- Form small mobile apps which can be implemented quickly and inexpensively instead of patching or modernizing an outdated multi-decade old brick-n-mortar data center.

7

Outdated Legacy Systems

Result

- Long, lead-item types
- High WIP, large batches
- Poor quality, performance

Anti-Pattern

- Intense fixation on sunk costs, tossing good money away, and unchecked escalation of information technology budgets and costs to rescue obsolete legacy systems.

Antidote

- Use guardrails, investment horizons, and portfolio budgets.
- Balance portfolio with past, present, and future solutions.
- Replace obsolete legacy systems vs. keep them forever.

Example

- Replace data centers with commercial cloud services, build small modern applications rather than preserve large old ones, and use software vs. hardware when possible.

8

Outdated Development Technology

Result

- Poor morale, attrition
- High WIP, large batches
- Poor quality, performance

Anti-Pattern

- Locked into obsolete technology stacks, development tools and systems, and technological skills that are no longer supported by vendors, consulting firms, and markets.

Antidote

- Use new operating systems and programming languages.
- Seek to replace legacy services with modern tech stacks.
- Proactively replace outdated technologies with new ones.

Example

- Decompose monolithic legacy system architectures into highly modularized microservices, refactor and age off unneeded bloat, and insert newer high-quality technology.

9

Long Lead Brick-n-Mortar Hardware Items

Result

- High WIP, large batches
- Schedule, cost overruns
- Poor quality, performance

Anti-Pattern

- Developing capital-intensive solutions such as custom, on-premises data centers, refreshing legacy system data centers with new hardware, or building hardware solutions.

Antidote

- Use commercial cloud services whenever possible.
- Develop small software MVPs such as mobile apps.
- Avoid capital intensive hardware items whenever possible.

Example

- Design small software MVPs or mobile apps that can be downloaded from mobile app stores, design cloud based microservices, and host these on commercial clouds.

10

Feature Factory Sweatshops

Result

- Poor morale, attrition
- Schedule, cost overruns
- High utilization, wait times

Anti-Pattern

- Filling lean-agile backlogs with mountains of business requirements, UX wireframes, system models, epics, features, and user stories contracted to lowest bidder.

Antidote

- Apply lean-agile frameworks to specify infrequent MVPs.
- Limit WIP and batch size to create a sustainable pace.
- Create an environment of innovation and discovery.

Example

- Establish a development value stream or team-of-teams to explore a product or service roadmap or portfolio of small business experiments to seek and discover user needs.

11

Integrated Master Schedules and EVM

Result

- Schedule, cost overruns
- High WIP, large batches
- High utilization, wait times

Anti-Pattern

- Investing in expensive integrated master scheduling teams to run scenarios and simulations for implementing large requirements backlogs to save outdated legacy systems.

Antidote

- Use lean-agile frameworks to specify infrequent MVPs.
- Create modern software-intensive replacement MVPs.
- Invest in smaller batches of limited business experiments.

Example

- Apply informal project networks, feature and story maps, and planning interval events to build small batches of business experiments and software MVPs with low WIP.

12

Seeking Economies of Scale and Full Utilization

Result

- Poor morale, attrition
- High utilization, wait times
- Poor customer satisfaction

Anti-Pattern

- Applying on-prem software factory concepts from the 1970s to achieve economies of scale and full utilization of limited computer programmers to reduce cost of high WIP.

Antidote

- Reduce utilization and WIP to create sustainable pace.
- Create long lived teams vs. matrixing and multitasking.
- Empower teams to identify small business experiments.

Example

- Incentivize bottoms up innovation riptides where teams co-develop product and service visions, roadmaps, and infrequent low-WIP experimental MVPs and solutions.

13

Manufacturing Statistics and Story Point Physics

Result

- → Poor morale, attrition
- → High utilization, wait times
- → Unsustainable pace, burnout

Anti-Pattern

- → Measuring performance in minutes and hours with a stopwatch like Tayloristic Kanban manufacturing factories and machines to achieve full utilization and optimum WIP.

Antidote

- → Use qualitative measures of performance when necessary.
- → Improve speed and performance with low WIP/batchsize.
- → Encourage and apply goal vs. scope-oriented thinking.

Example

- → Incentivize and coach teams to establish valuable but achievable goals, deliver small timeboxed experiments, and use innovation metrics to gauge end-user outcomes.

14

Overmeasuring Performance in Minutes and Seconds

Result

- → Poor morale, attrition
- → High utilization, wait times
- → Unsustainable pace, burnout

Anti-Pattern

- → Establishing too many metrics, measures, and models to predict outcomes based on lagging indicators in a vain attempt to achieve overloaded schedules and backlogs.

Antidote

- → Use fewer measures rather than more measures.
- → Use qualitative measures of performance when necessary.
- → Focus on value adding outcomes vs. track-and-field stats.

Example

- → Incentivize, empower, and coach teams to establish valuable, achievable, and qualitative goals, deliver small timeboxed experiments, and use softer innovation metrics.

15

Decomposing Stories into Individual Tasks

Result

- Poor morale, attrition
- High WIP, large batches
- High utilization, wait times

Anti-Pattern

- Decomposing stories into tasks that are assigned to individuals for full utilization, maximum micromanagement control, implementing high WIP, and eliminating freeriders.

Antidote

- Implement all stories as teams vs. individuals.
- Apply pair programming principles and practices.
- Balance utilization and valuable performance goals.

Example

- Specify small number of user stories per iteration to achieve the end-user value of small business experiments to be implemented in small agile teams (i.e., less is more).

16

Not Using Architectural Runways or Enablers

Result

- Poor morale, attrition
- Schedule, cost overruns
- Unsustainable pace, burnout

Anti-Pattern

- Filling backlogs with functional requirements on a tight schedule and low budget while discovery, runways, and spikes are performed for free on evenings and weekends.

Antidote

- Plan discovery, enablers, and architectural runways.
- Form full time discovery and architectural runway teams.
- Allow teams to alternate between features and enablers.

Example

- Form enabler teams, epics, features, and stories to build just-in-time shared services, platforms, and tools to improve quality, speed, sustainability, and innovation.

17

No Lean-Agile Product Management

Result

- High WIP, large batches
- High utilization, wait times
- Poor quality, performance

Anti-Pattern

- Substituting legacy business requirements specifications, requirements and modeling teams, and integrated master schedules for lean-agile product management teams.

Antidote

- Create a formal lean-agile product management team.
- Train and certify team in Lean UX principles and practices.
- Specify just-enough, just-in-time business experiments.

Example

- Have a small lean-agile product management team use one-week design sprints to form individual Lean UX business experiments which can be tested in a few sprints.

18

Buyer-Supplier Role Division

Result

- High WIP, large batches
- High utilization, wait times
- Poor quality, performance

Anti-Pattern

- Designating buyers in primary decision-making roles like solution and product management, development value stream and agile team process facilitation, etc.

Antidote

- Share roles and responsibilities with buyers and suppliers.
- Integrate and distribute buyers and suppliers into teams.
- Coach consistent principles and practices across teams.

Example

- Integrate and alternate buyers and suppliers into teams and roles (i.e., one team has a buyer product owner or Scrummaster, while another has supplier leadership).

19

Supplier Side Roles and Responsibilities

Result

- → Suboptimal decisions
- → Poor morale, attrition
- → Poor performance

Anti-Pattern

- → Forming predominantly supplier side lean-agile development value streams and teams-of-teams with supplier side only roles and responsibilities.

Antidote

- → Share roles and responsibilities with buyers and suppliers.
- → Integrate and distribute buyers and suppliers into teams.
- → Coach consistent principles and practices across teams.

Example

- → Integrate and alternate buyers and suppliers into teams and roles (i.e., one team has a buyer product owner or Scrummaster, while another has supplier leadership).

20

Uneven Team Resource Loading

Result

- → Poor morale, attrition
- → High utilization, wait times
- → Bottlenecks, longer queues

Anti-Pattern

- → 80% to 90% of teams perform management and administration, while 10% to 20% of the teams perform value-adding development of product and service features.

Antidote

- → Designate more development than management teams.
- → Distribute feature workload among development teams.
- → Seek to rectify systemic over and under team utilization.

Example

- → Form development value streams where 80% of agile teams are small lean-agile development teams and personnel are dynamically distributed based on workload.

21

Death by Marathon Meetings

Result

- Poor morale, attrition
- Low productivity, velocity
- Minimum business value

Anti-Pattern

- Scheduling all day solution and product management meetings, technical interchanges, Scrum meetings, or daily standups and keeping people as long as possible.

Antidote

- Use short formal meetings as infrequently as possible.
- Encourage people to self-organize to form deliverables.
- Error on the side of meetings under 30 minutes if possible.

Example

- Have teams do lean-agile product management activities without large formal meetings, use short early Scrum meetings when necessary, and half this time if possible.

22

Lack of Process Tailoring, Self Organization, or Streamlining

Result

- Low productivity, velocity
- Minimum business value
- Bottlenecks, longer queues

Anti-Pattern

- Mandating all teams on development value streams to apply the longest possible and most frequent textbook lean-agile framework events, ceremonies, and meetings.

Antidote

- Allow teams to self-organize to achieve process goals.
- Allow teams to tailor agile processes based on context.
- Achieve a balance of formal and informal process needs.

Example

- Teams doing routine or labor-intensive work may be able to combine and streamline events, while teams doing highly creative work may require rigorous brainstorming.

23

Complex or Expensive ALMs

Result

- Excessive training budget
- Entry barrier, learning curve
- Lower return on investments

Anti-Pattern

- Purchasing complex and expensive ALMs that require multimillion dollar investments in full-time programming teams, expensive training, and labor-intensive data entry.

Antidote

- Use the simplest possible ALM tools for large teams.
- Use low-cost commercial cloud services when possible.
- Achieve fine balance of manual and automated reporting.

Example

- Have a development value stream use Confluence and Jira for planning interval artifacts and tracking, and other simple visual collaborative brainstorming tools like Mural.

24

Failure to Think "Out of the Box"

Result

- Poor morale, attrition
- Low productivity, velocity
- Bottlenecks, longer queues

Anti-Pattern

- Mindlessly following prescriptive lean-agile frameworks, recommended best practices, counterproductive rules and sequences, and disallowing deviations or customization.

Antidote

- First, consider the basic principles and practices.
- Focus on the “commander’s intent” vs. letter-of-the-law.
- Zoom-in-and-out and consider the opposite if applicable.

Example

- Vary centralization and decentralization, top-down vs. bottoms up, push vs. pull, low vs. high WIP, achievable vs. stretch goal, and reverse events if they cause train wrecks.

25

No "One Team" Culture and Cooperation

Result

- Low productivity, velocity
- Bottlenecks, longer queues
- Minimum business value

Anti-Pattern

- Competition between buyers and suppliers, product owners and Scrummasters, development value streams, individual agile teams, and the individuals within teams.

Antidote

- Create and incentivize a “one-team” mindset and culture.
- Create shared goals and reward teamwork vs. individuals.
- Teams need to cooperate to achieve value stream goals.

Example

- Incentivize development value streams to form and achieve a “one-team” mindset, work together to achieve planning interval goals, and seek out opportunities to help.

26

Treating People Like Automatons or Interchangeable Cogs in a Wheel

Result

- → Poor morale, attrition
- → Low productivity, velocity
- → Minimum business value

Anti-Pattern

- → Giving developers planning interval objectives, epics, features, and stories without bottoms up self organization and selection (while leadership leads people over a cliff).

Antidote

- → Use built-in process improvement events and ceremonies.
- → Incentivize the identification of systemic bottlenecks.
- → Fix bottlenecks instead of hiding your head in the sand.

Example

- → If a large monolithic legacy system has unreasonable bottlenecks and issues, then incentivize and empower teams to identify and repair them vs. ignoring bottlenecks.

27

Late Big Bang Quarterly Testing Events

Result

- Poor morale, attrition
- Low productivity, velocity
- Bottlenecks, longer queues

Anti-Pattern

- Scheduling labor intensive big bang testing events in infrequent intervals (i.e., three, six, nine, or twelve months or more), which often occur during holidays or Summers.

Antidote

- Apply development testing (i.e., one piece workflow).
- Apply test and behavior driven development practices.
- Automate as much testing as possible (as you go along).

Example

- Acceptance criteria for user stories must include writing automated tests in advance, check into version control, and running and passing each test before moving on.

28

Independent Manual Testing Teams

Result

- → Poor morale, attrition
- → Low productivity, velocity
- → Bottlenecks, longer queues

Anti-Pattern

- → Forming large manual testing teams, writing manual test procedures, and independently running regression tests in long infrequent intervals while developers go on vacation.

Antidote

- → Apply development testing (i.e., one piece workflow).
- → Apply test and behavior driven development practices.
- → Automate as much testing as possible (as you go along).

Example

- → Develop and apply a cloud based continuous integration, continuous delivery, and DevOps pipeline, write and contribute development tests, and run them continuously.