Dave's Notes—Does the Scaled Agile Framework (SAFe) Apply to Safety-Critical Systems?

What is the Scaled Agile Framework (SAFe) and when to use it?

- SAFe is a multi-dimensional reference model for applying lean and agile concepts for business or enterprise agility, lean portfolio management, program (multi-project) management, and project management (multiple teams building large or complex products and services). Its basic purpose is to help manage the development of moderately complex ecosystems.
- SAFe's sweet spot or goldilocks zone is the application of lean and agile best practices for large solutions (multiple programs) and programs or projects (multiple lean and agile teams building large or complex products, services, or service products).
- SAFe is a unique fusion of lean and agile concepts into an integrated whole that includes value stream mapping and analysis, lean startup, experimentation, design thinking, lean UX, product management, CI, CD, DevOps, Scrum, Kanban, etc.
- SAFe is a highly simplified, streamlined, minimal, and experimental approach for teasing out complex suites of minimum viable products (MVPs) for successfully developing research and development (R&D) oriented new products and services with high degrees of complexity, uncertainty, and risk (it's a mistake to slow R&D with traditional and manufacturing models).
- SAFe is steeped in lean and agile product management, design thinking, and user experience (UX) disciplines for ensuring enterprises, portfolios, value streams, programs, projects, and development teams are sharply focused on solving critical value adding problems for markets, customers, and end users (SAFe is customer and market centric, focused, or facing).
- SAFe's unique lean thinking approach is designed to decimate lead and cycle times, speed up feedback and learning cycles, and continuously improve new products and services to optimize value for buyers (customers) and suppliers (developers).
- SAFe is built upon the four broad values of (1) Built in Quality, (2) Alignment, (3) Transparency, and (4) Program Execution (reducing historically high project failure rates due to unnecessary long-lead item complexity, overscoping, and gold plating).
- SAFe's mantra is consistency, accountability, predictability, and reliable delivery in the merciless face of an extremely volatile, uncertain, complex, and ambiguous (VUCA) world (i.e., reigning in complexity with lean and agile concepts vs. crushing them with traditional process and document heavy waterfall lifecycles, governance, milestone reviews, enterprise architectures, business requirements, schedules, etc.). Traditional nor lean manufacturing paradigms are sufficient in a VUCA world.
- Although SAFe is primarily designed for systems and software development projects, like Scrum and Kanban, SAFe is generalizable to enterprise strategic planning, operations, administration, and manufacturing contexts (although it's ideal for cloud computing and other software defined products and services)—The more flexible the medium is, the more lean and agile one can become (the application of flexible mediums such as software is the embodiment of lean thinking).
- SAFe also embraces test driven development, behavior driven development, acceptance test driven development, continuous integration, continuous delivery, and DevSecOps (which are key enablers for developing mission and safety-critical systems with uncompromising levels of quality, reliability, and dependability at lightning speed, global scale, and minimal cost).
- SAFe provides the tools to directly tap into your market's, customer's, and end-user's most pressing problems, decimate lead and cycle times, validate assumptions, and build just-in-time MVPs with high levels of both UX and uncompromising quality.

Common Myths and Misconceptions about Agile Methods?

- Agile methods are only for software development projects – Agile is now used for new startups, reengineering corporations (business agility), manufacturing (hardware), business operations (administration), and just about any discipline.
- Agile methods ignore user experience (UX) design – Design thinking and Lean UX are now fully integrated into agile methods to bring a sharp focus on the end-user as well as optimal user experience, usability, and utility.
- Agile methods ignore quality and reliability – Agile methods, especially through the application of Extreme Programming (XP) practices, such as TDD, BDD, ATDD, CI, CD, DevOps, and DevSecOps, result in a historically low Cost of Quality (CoQ)—SAFe and other agile frameworks like Scrum encourage collaboration, focus (limited WIP), and fast feedback cycles to rapidly tease out hidden tacit information for rapidly building high-quality complex solutions (while improving cost, risk, and quality performance by an order of magnitude)—Conversely, traditional methods never consistently succeeded in doing the same that are based on documenting incorrect assumptions leading to low quality solutions (vs. rich, high-context communication, collaboration, and cooperation frameworks like SAFe that dramatically increase quality, speed, and customer satisfaction).
- Agile methods are only for small, low-risk prototyping projects – Agile methods, especially through the use of SAFe, are well suited for larger, more complex solution spaces, suites or ecosystems of loosely coupled and highly cohesive MVPs (like a suite of mobile phone apps), and even complex, global, systems of systems (like over-the-air software apps to IoT devices, vehicles, aircraft, computer systems, etc.)—SAFe and other agile frameworks reduce vs. increase risk of complex systems (and were created to successfully repair and replace traditional frameworks that never worked in complex domains).
- Agile methods are inferior to lean thinking – Agile methods were born from the lean thinking movement, have fully absorbed and enriched lean thinking ideas, and are replacing traditional lean thinking practices such as the Toyota Production System (TPS)—Agile methods are a unique blend of some traditional (push) and lean (pull) ideas to bring just the right amount of business and market-driven focus to create profitable products that satisfy market and mission needs that lean frameworks can't do alone. Agility is the evolution of lean thinking, not its devolution nor first step on the way to traditional lean thinking.
- Agile methods are obsolete now – Agile methods still exist, are growing, and continue to be a global phenomenon that have enriched paradigms such as Design Sprints, Design Thinking, Lean UX, Product Management, Startup Way, Business Agility, etc.—Kanban alone simply doesn't have the tools, frameworks, and richness necessary to compete with agile methods and frameworks. SAFe is a quick way to build solutions for realizing business value whereas Kanban is a slow drip approach.

Common Myths and Misconceptions about the Scaled Agile Framework (SAFe)?

- SAFe is a prescriptive model – SAFe is a normative or descriptive reference model almost like an ISO standard, does not contain prescriptive step-by-step guidance, is adaptable or scalable to almost any context, and is not rigidly inflexible.
- SAFe is only for agile methods – SAFe fully embraces and integrates a lightweight fusion of lean thinking concepts and almost has more lean thinking values, principles, and practices than agile only frameworks such as Scrum, XP, etc.
- SAFe is only for software projects – SAFe is primarily designed for software-intensive systems but is applicable to a wide variety of IT technologies, business or administrative functions, manufacturing contexts, hardware systems, and most other new products and services (training, food products, consumer electronics, financial instruments, strategic planning, planes-trains-and-automobiles, spacecraft, etc.). This isn't to say that software-intensive industries are only a minor niche market.
- SAFe is a Kanban system – While SAFe includes many lean thinking values, principles, and practices such Kanban, SAFe is a fusion of much more such as product management, design thinking, lean UX, CI, CD, DevOps, lean startup, business experiments, etc. (SAFe is not just a glorified Kanban system, nor an architectural runway for building an enterprise Kanban system). Many people use SAFe as a starting point for onboarding enterprises to a niche product development approach.
- SAFe encourages waterfall practices – Like lean and agile methods such as Scrum, XP, or Kanban, SAFe can be used as a feature factory to realize complex integrated master schedules, business requirements, enterprise and system architectures, etc., but this is not its primary application (i.e., SAFe is a lightweight innovation framework for teasing out market, customer, and end-user needs with small business experiments, MVPs, and other informal market probes, prototypes, early models, demonstrations, etc.)—SAFe (unlike Kanban and DevOps alone) recognizes the necessity of leveling and managing workflow with Constant WIP (CONWIP)—That is using Lean-Agile Solution/Product Management, Design Thinking, Lean-UX, Dual Track Development, and other Shared Services teams to create just-in-time enablers, architectural runways, and system-wide backlogs to keep teams moving at a sustainable pace (without overwhelming teams with too much WIP or starving them).
- SAFe is a heavyweight WIP-intensive framework – SAFe was built on the simple notion of a Scrum of Scrums, XP's release planning model, and a smattering of other lean, product management, UX, and DevOps practices (i.e., it's a better Scrum at Scale than S@S)—SAFe ensures a just-in-time level of Constant WIP (CONWIP) is available in the form of lightweight roadmaps and near-term backlogs to keep lean and agile teams productive (but not over or fully utilized).
- SAFe cannot be used for safety critical systems – SAFe is built upon lean thinking concepts used to build safety critical systems such as automobiles, aircraft, electronics, transportation systems, and even run hospitals, medical clinics, and emergency rooms. SAFe is not a manufacturing approach, but it can be used to create innovative safety critical solutions.
- SAFe is obsolete now – SAFe continues to evolve, adapt, and absorb major new innovations faster than competing frameworks including product management, design thinking, UX, lean startup, experimentation, CI, CD, DevOps, Business Agility, etc. While many view SAFe as its initial set of agile project management practices, SAFe is far better than it ever was.
- SAFe success requires organization change first – Change comes last in SAFe, while implementation of SAFe practices comes first as organizations will change when they see SAFe work (organizations will not change without firsthand visual evidence or experience)—That is, SAFe is designed to be put to work first building valuable products and services (SAFe is not a change management model, nor does it require one). This isn't to say that organization psychologists aren't necessary.

Can the Scaled Agile Framework (SAFe) Be Applied to Safety-Critical Devices?

- SAFe like lean, Kanban, Scrum, and XP can and has been used for building safety critical devices – Again, SAFe is built upon lean thinking concepts and is a more holistic lean thinking framework than rivals such as S@S, DaD, LeSS, TPS, Kanban itself, Six Sigma, Lean Six Sigma, etc. (SAFe is a more productive just-in-time framework than traditional quality frameworks like ISO 9001, Six Sigma, and Lean Six Sigma, and SAFe is explicitly designed to yield immediate market value).
- SAFe fully embraces TDD, BDD, ATDD, CI, CD, DevOps, and even DevSecOps – These paradigms, frameworks, and tools are built upon the concept of applying concise acceptance criteria or formal mathematical Gherkin axioms, automating them, and then running them millions of times a day to verify and validate products and services (dot coms run hundreds of millions of tests every day that rival and surpass the quality and reliability of any medical device manufacturer on the planet).
- SAFe embraces secure over-the-air or wireless DevSecOps updates – This is much like electric automobile makers such as Tesla which is one of the most reliable safety critical devices ever conceived, and also includes NASA's Space Station, deep space probes, and interplanetary rovers (NASA was still engineering its vehicle control software while Perseverance was on the way to Mars, uploaded the software after it landed, and continues to improve its software for optimal mission performance every day using DevOps values, principles, practices, and technologies from a distance of 110 million miles).
- There are ANSI standards for the application of agile methods to medical devices – There are numerous books and journal articles on successful case studies of applying agile methods to safety critical systems development (which includes SAFe).
- SAFe is being used to implement the U.S. Government's integrated Electronic Healthcare Records (iEHR) system – Agility was used to rescue the Affordable Care Act (ACA) website and SAFe is fully embraced by the Health and Human Services (HHS) Centers for Medicaid and Medicare Service (CMS), especially on the larger Medicare side at the portfolio level.

Common Scaled Agile Framework (SAFe) Challenges and Solutions?

- SAFe experience continues to be a problem – There are a dearth of trained and certified personnel with the experience to administer the SAFe implementation roadmap, setup SAFe projects, operate them appropriately, adapt them to a domain, etc. (oftentimes SAFe SPCs saturate enterprises with too much upfront training and organizational change vs. putting SAFe to work building valuable products and services). There are many wolves in sheep's clothing who often don't practice SAFe.

- SAFe implementation roadmap – SAFe comes with an implementation roadmap to help actively certified SAFe coaches and leadership teams to properly implement, operate, improve, scale out SAFe across values streams and enterprises.
- SAFe barrier to entry – The SAFe implementation roadmap can seem a bit onerous at first, and it takes some expertise and confidence to move through it briskly without getting lost in the weeds (it's an awesome tool that must be used with care).
- SAFe training – SAFe recommends that programs, projects, or agile release trains (ARTs) be trained before starting, which may seem a bit onerous at first, but is essential to a strong SAFe startup at the team, ART, Solution Train, portfolio, etc.
- SAFe overtraining – Some SAFe coaches recommend training everyone in a small enterprise or portfolio for optimal buy-in, whereas SAFe recommends targeting small to medium sized projects first (eat the elephant one bite at a time).
- SAFe undertraining – While the ART developers are often targeted for training, all business functions associated with an ART should be trained and even retrained to enforce SAFe concepts (including APMO staff, customer, administration, supporting functions, shared services, etc.). One must be careful when deepening vs. fracturing a team's management approach.
- SAFe adaptation – Again, SAFe is primarily designed for software-intensive systems, but people are smart and should be trusted to adapt it to their context, even if it is a business strategy or growth function, business operations, portfolio management, new product or service development, etc. (i.e., SAFe is extensible beyond software-intensive systems alone).
- SAFe is a team sport – SAFe should be implemented by a small team of people (two or three) for a small program, project, or ART (SAFe should not be implemented by an individual, because it's too much work, and requires validation of implementation ideas by a second party so one doesn't skip important elements or go down a rabbit hole which is quite common). Two heads are better than one when it comes to pair programming, Scrummastering, ART management, etc.
- SAFe is a continuous improvement paradigm – SAFe has many built in opportunities for continuous improvement, has built in assessments, and many valuable qualitative and quantitative measurement tools for gaining rapid situational awareness.
- SAFe is falling out of favor – Many people view SAFe as an outdated, outmoded, and irrelevant concept in lieu of lean startup, startup way, design sprints, product management, design thinking, UX, DevOps, Kanban, etc. (while SAFe offers pre-packaged fully-integrated scaling implementation guidance these other standalone paradigms simply don't offer).
- SAFe is supplanted or highjacked by Kanban – There is more demand for SAFe than Kanban, so many Kanban trainers use SAFe training to get Kanban clients, criticize SAFe, replace its training slides, and then replace SAFe with Kanban which is a mistake (SAFe is a new product and service innovation model vs. a Kanban manufacturing model—Kanban can't compete with SAFe as a standalone paradigm)—It's a mistake to bait-n-switch manufacturing for innovation application areas.
- SAFe is irrelevant to Cloud Computing, Design Thinking, UX, DevOps, or non-Software projects – All of these paradigms are knowledge intensive contexts or paradigms (like new products and services), so SAFe is fully applicable to all of these contexts, situations, environments, and applications (SAFe's goldilocks zone is high-risk, highly-uncertain bleeding edge IT).
- SAFe is only for extremely large portfolios, programs, or projects – SAFe can be applied to as few as three Scrum or Scrumban teams (25 to 75 people) – In fact, the smaller the better as its easier to rollout SAFe on a small to medium-size program, project, or ART than a large one (it's probably not ideal to oversize an ART, Solution Train, portfolio, or enterprise all at once—Eat the elephant one bite at a time).
- SAFe is only for development teams – Successful proliferation, implementation, and operation of SAFe (yielding its benefits) starts at the top – SAFe is a leadership concern and SAFe must be embraced from the very top including the business unit, portfolio, solution, or agile program management office (APMO) team (SAFe cannot operate successfully in a vacuum by small development teams while the APMO uses a traditional, hybrid, ad hoc, or orthogonal management paradigm).
- SAFe is best for hybrid traditional-agile projects – SAFe is best for SAFe projects, it is not wise to hybridize it into an iteration-driven feature factory or sweatshop to implement bloated enterprise architectures, integrated master schedules, business requirements, or other full-utilization concepts, paradigms, models, or frameworks (SAFe should NOT be used for full or maximum utilization—Squeezing blood out of a turnip as is commonly attempted).

Common Scaled Agile Framework (SAFe) Anti-Patterns?

- Transformation as individual vs. team – Need a small group of SPCs (at least two) to implement a small ART (and 3 or 4 SPCs for a Solution Train).
- Excluding SAFe coaches from APMO or RTE Team – Using SAFe coaches as trainers, team coaches, or excess baggage (best to include them on APMO, RTE team, and general ART level assessment, improvement, and oversight).
- Failure to Apply SAFe assessments – Ignoring SAFe's built-in assessments, rolling your own, and/or not doing them at all (using them gives you immediate data and feedback).
- Failure to Use SAFe for SAFe's sake – Using SAFe as a lead in or architectural runway to implement an alternative framework, model, paradigm, or bait-and-switch approach (i.e., LeSS, DaD, S@S, Enterprise Kanban, etc.).
- Failure to have a small dedicated ALM team – Not having an ALM team, having an individual, or having the ALM team be an adversary to the ART, coaches, and staff (ALM teams, when they do exist are territorial, combative, and noncooperative).
- Failure to use basic SAFe metrics – Most teams avoid program predictability (scoring of PI objectives) by business owners, customers, solution managers, or product managers (and instead contract, cost, and count user stories and story points).
- Using SAFe for Vanity Measures – Focusing on story points, velocity, agile earned value management (EVM), output measures, and measuring development speed (vs. outcome measures)—One must use both output and outcome measures with SAFe (with a focus on the latter)—Don't just focus on training metrics, velocity, utilization, cost and schedule performance, etc. (but on market, customer, and end-user value)—95% of business experiments (MVPs) yield no value.

- SAFe is vulnerable to Goodhart's Law – Any measure immediately ceases to become useful once it is selected as a target metric (SAFe projects will suboptimize on a single vanity metric such as story points, velocity, or even outcome measures)—While its more common for SAFe projects to suboptimize on vanity metrics (story points) to the exclusion of outcome measures (profit), suboptimizing on outcome measures alone generally leads to gaming the system to achieve the outcome measure (even if it means breaking the law)—Outcome measures are more susceptible to legal malfeasance (i.e., Twitter, Verizon, or Enron faking customer accounts to bolster stock market valuation and garner outside investors and bank loans).
- Treating ARTs as a Feature Factory – SAFe is not meant to be a sweatshop for implementing enterprise architectures, integrated master schedules, or five-year-old business requirements documents at breakneck speed by cheap IT workers.
- Treating user stories as requirements – User stories are an agreement to have a conversation (90% of the user story is in the tacit conversation)—SAFe is NOT a documentation driven framework (it is a framework that facilitates interpersonal communications, collaboration, cooperation, and teamwork at scale)—Business value is in the conversation or the act of sharing valuable, real-time, just-in-time information and data (vs. obsolete, incorrect, or ambiguous documents).
- Failing to invest in architectural runways or enablers – Having teams only build epics, features, or user stories, but not invest in discovery, middleware, or other enabling IT fabrics, functions, and services (i.e., ask teams to take enablers out of hide).
- Scaling up too large – Forming a 200, 300, 500, or even 1,000 person ART or Solution Train – First of all SAFe should not have an ART greater than 125 people (includes shared services and supporting functions not just developers), 50-75 people is ideal.
- Not having a consistent ALM tool established that supports SAFe – Scrum is the fundamental atomic unit in SAFe, so it's okay if the ALM only supports Scrum, but it's much better if the ALM is designed for SAFe or has essential SAFe widgets, gadgets, etc. (this includes solution or program boards, which can be added onto basic ALM tools, but not very easily).
- Not having a solution or product management team – Forming solution and product backlogs prior to SAFe program increment planning and having STEs and RTEs facilitate these ceremonies, including early PI pre-planning by Scrum teams is a must (even the latest Scrum guide says a Scrummaster's major goal is to facilitate the creation of a Product Backlog BY the Product Owner)—Therefore, STEs and RTEs should be facilitating the creation of solution and product management backlogs (and they're failing if they push it on Scrum teams to make bricks without straw which they do by default).
- Having too much WIP or overutilized feature factories – Filling everyone's backlog to 150% capacity (especially if Scrum teams have to create their own solution and product management backlogs).
- Subjecting ARTs and Scrum teams to drive by tasking – Adding new epics, features, user stories, and tasks to teams after PI and Sprint planning and treating them like personal programming and admin teams (with burning hail or meteor storms).
- Using story points as days or hours – Story points were never meant to be staff days or staff hours, used for contracts, budgets, or cost management, nor used for integrated master schedules, earned value management (EVM), timekeeping, and tracking individual utilization (if you need fine grained measurement, then just use staff hours, days, weeks, months, etc.).
- Inconsistent Scrum practices – Having some teams do SAFe PI and Scrum planning ceremonies, while other teams fake it.
- Overbearing product owners – Product owners with too much power who refuse to accept user stories, grow their scope beyond control, overload Scrum teams instead of reducing WIP, and strive for full utilization of Scrum teams (that's simply asinine)—English is an ambiguous language, and user stories and acceptance criteria are too easy to be abused by overbearing product owners—Product owners are not project managers, nor are they Scrummasters, to which some aspire.
- Failing to use the IP sprint for innovation – Using IP sprint as just another development sprint or jamming more work in the IP sprint (or using IP sprints for major labor-intensive system releases).
- SAFerfalling or Scrummerfalling – Saving all of your development for the last sprint in a PI or after two PIs and having big bang testing events instead of DevOps, daily deployments, and small inexpensive business experiments.
- Not using DevOps – Failing to standup a basic CI, CD, and DevOps pipeline because SAFe coaches don't understand it or don't think it's necessary (and whining about how development teams are simply not ready for DevOps practices and tools).
- Lack of teamwork within and across ARTs – Failure of teams to support one another, rivalry and competition, and even conflict and blocking other teams on the ART for personal aggrandizement (an ART is a team-of-teams, they should pitch in to help one another out, and they should probably have user stories specifically for cross-team cooperation)—However, this should be planned, coordinated, and formally requested (managers should simply not walk into a team's space and request unplanned, out-of-scope work).
- Hybridizing SAFe – Having a traditional IMS driven PMO but expecting Scrum teams to use SAFe as fully utilized feature factories working 60 hours a week at maximum capacity, utilization, or overallocation.
- Overscoping ARTs – Adding too much WIP to ARTs, fully utilizing ARTs, or underbidding contracts and having too few people to deliver the scope of the work in the contract (which is quite common)—If you've underbid your contract by two-thirds, then it's time to reduce WIP vs. sacrifice architectural runways, enablers, or promise 150% of the contract's scope.
- No leadership buy in – Have a program manager, APMO, or admin staff that is not trained in SAFe nor supports it (SAFe doesn't work if bloated APMOs violate SAFe's values, principles, and practices with traditional management nonsense like integrated master schedules, earned value management, enterprise architectures, business requirements, etc.).

Myklebust, T., & Stalhane, T. (2018). The agile safety case. Cham, Switzerland: Springer.

Rico, D. F. (2022). Generic Safety Critical Systems Standards and Activities. http://davidfrico.com/agile-safety.jpg

Hanssen, G. K., Stalhane, T., & Myklebust, T. (2018). SafeScrum: Agile Development of Safety-Critical Software. Cham, Switzerland: Springer.

Van Schooenderwoert, N., & Shoemaker, B. (2018). Agile Methods for Safety-Critical Systems: A Primer Using Medical Device Examples. Charleston, SC: CreateSpace.

Van Schooenderwoert, N., & Shoemaker, B. (2021). Agile Methods for Safety-Critical Systems: Case Studies of Medical Product Companies. Charleston, SC: CreateSpace.

Email. dave1@davidfrico.com • **Background**. http://davidfrico.com/daves-background.pdf • **YouTube**. http://www.youtube.com/c/DavidRico/videos • **Slideshare**. http://www.slideshare.net/davidfrico/presentations • **Business Card**. http://davidfrico.com/bcard.jpg • **Agile Book**. http://davidfrico.com/agile-book.htm • **Website**. http://davidfrico.com

Generic Safety Critical Systems Standards & Activities