The Pros and Cons of Major Agile Roles for Neurodivergents

Abstract

Bottomline Upfront. *Some Agile roles like Product Manager and Product Owner requiring robust foresight, planning, serendipity, and social interaction may be better suited for normal (neurotypes). Conversely, other Agile roles like Release Train Engineer (RTE) and Scrummaster (SM) requiring an intense present focus with minimalistic, repetitive, well-defined rules, and well-scripted telegraphed events and communications may be better suited for neurodivergents.*

Motivation. *The popularity, adoption, and adaptation of lean-agile frameworks like the Scaled Agile Framework (SAFe), Scrum, Scrumban, and Kanban continue to be applied in all market sectors including strategic planning, business operations, and information technology (IT) disciplines. Part and parcel to lean-agile frameworks is moving lightning fast to develop, deliver, and introduce innovatively new products and services with short lead and cycle times to disrupt global markets faster than the competition. The use of artificial intelligence (AI) tools and techniques like Gen AI, GPT, and AI-enabled Agile Lifecycle Management (ALM) systems shortens lead and cycle times of new products and services even further placing greater cognitive constraints on the people using lean-agile frameworks.*

Problem. *The main challenge is that global firms are faced with using a broad spectrum of neurodivergent personnel ranging from high-performing neurotypes with lightning-fast high-capacity brains and neurotransmitters to somewhat lower performing neurodivergents with slower, slightly impaired brains, memories, capabilities, skills, and neurotransmitters. You can stock your global enterprise with only the top 0.1% of high performing neurotypes like Netflix at any cost or you can settle for employing a workforce of neurodivergents widely available to global firms. The questions become: "Does the future involve only high-performing neurotypes? Can global firms succeed using lean-agile frameworks with a neurodivergent workforce? Are all industry sectors created equally in terms of lead and cycle time demands? Are there some lean-agile roles and responsibilities better suited for neurotypes vs. neurodivergents?"*

Approach. *Our approach is to examine the attributes, characteristics, skills, and capabilities of neurotypes vs. neurodivergents. We will examine the requirements, demands, and constraints of lean-agile product vs. process-oriented roles. Then we will do a comparative evaluation of lean-agile product and process roles and responsibilities against neurodivergent capabilities to determine if there is a clear delineation among a broadly neurodivergent workforce. As such, we will determine if some lean-agile roles are better suited for neurotypes vs. neurodivergents.*

Results. *Our results are to recommend a framework for classifying the capabilities of neurodivergent professionals, the demands and constraints of lean-agile roles and responsibilities, and suggestions for neurodivergents when selecting lean-agile roles and responsibilities for optimal performance. That is, help global firms successfully apply lean-agile frameworks for developing and delivering innovatively new products and services with short lead and cycle times (as well as neurodivergents themselves who wish to participate in the application of lean-agile frameworks).*

Conclusion. *Our conclusion is that lean-agile framework roles and responsibilities place highly specialized cognitive constraints on product and process roles that may restrain the selection of neurotypical vs. neurodivergent personnel. That is, if global firms want to be successful, then there are specific cognitive skills, abilities, capabilities, and performance histories needed for lean-agile product and process professionals. Global firms intuitively recognize the needed skills, talents, and abilities of product and process professionals. However, lean-agile frameworks may not clearly or distinctly specify the cognitive abilities of their roles and responsibilities. Furthermore, neurodivergents may not recognize the cognitive constraints of lean-agile frameworks themselves. Therefore, our conclusion is to establish a set of operating conditions in which global enterprises, neurotypes, and neurodivergents may be intuitively successful.*

Footnote. *The field of neuropsychiatry continues to expand and mature, illustrating the neuroplasticity of the human brain. Once considered a byword, neuroplasticity is now a well-respected discipline. That is, the ability of the human brain to grow, adapt, expand, and succeed in highly complex domains. There is growing evidence that Applied Behavioral Analysis (ABA), health management, and pharmaceuticals may be used by neurodivergents to adapt to a higher-performing neurotypical state. This helps neurodivergents to perform roles and responsibilities historically constrained to neurotypes (with limitations). While we may attempt to classify neurotypes and neurodivergents into specific lean-agile roles and responsibilities, we recognize the ability for neurodivergents to grow into advanced lean-agile roles. So, the lessons here should not only be applied to neurodivergents, but higher performing neurotypes when selecting people of all neurodivergent levels for performing lean-agile product and process roles and responsibilities.*

Introduction

Let's begin with a brief recap on lean-agile frameworks. The term "Lean Thinking" was coined in the Western Hemisphere circa 1990 as an interpretation of Japanese manufacturing practices. That is, Japan's manufacturing industry was in its golden age, while America's manufacturing industry and economy had been in sharp decline for two long decades. The U.S. had come to the conclusion that the key to Japan's success was competing on speed, getting early market feedback on innovatively new products and services, and fast continuous improvement. Therefore, the Western conclusion was Western enterprises must also compete on speed with "Lean Thinking." This, of course, was just the beginning, as Japanese manufacturing capabilities were very sophisticated, so the notion of Western "Lean Thinking" was just the beginning or early arithmetic to the adoption of complex Japanese manufacturing principles, practices, tools, terms, phrases, expressions, colloquialisms, and statistics.

U.S. commercial computer programmers also adapted a lightweight form of "Lean Thinking" called "Agile Methods." Commercial computer programmers were lean, mean, and lightweight to begin with, talented computer programmers were the key to commercial software development success, the correct process existed in their heads as tacit knowledge, and the software source code represented the explicit knowledge or documentation. In other words, the key to Western computer programming success was a talented programmer, tacit experience, and well-structured version-controlled source code. It's important to note that public sector organizations had been developing heavy weight big batch document and process driven linear lifecycles for 50 years to deskill the computer programming process. Whereas speed and skill were the key to commercial programmers, 1,000-person multi-decade long, billion-dollar computer programming projects were the norm in the public sector (replete with floor crushing libraries of software documentation). Voluminous documentation was the hallmark trait of this approach.

Basically, the emergence of "Lean Thinking" was all Western computer programmers needed to hear to validate and codify their lean, mean, and lightweight tacit knowledge driven principles, practices, and tools. Lean-agile frameworks were not a rebellion against the heavyweight billion dollar approaches to computer programming institutionalized by public sector enterprises. Well, at least, not necessarily. Lightweight frameworks emerged in droves like Extreme Programming (XP), Scrum, Scrumban, Agile Project and Portfolio Management, and the Scaled Agile Framework (SAFe) rounding out the computer programming "Lean Thinking" movement by herding the cats and aggregating the principles and practices from the emerging ecosystem. For heavy weight framework proponents, XP, Scrum, Scrumban, and SAFe were too undisciplined. For commercial computer programmers, they were far too structured for anything but larger computer programming projects.

The bottom line is that lean-agile frameworks such as XP, Scrum, Scrumban, and SAFe were designed to quickly formulate and introduce innovatively new products and services with the shortest possible lead and cycle times. Certainly, faster than anything an Eastern or Western manufacturing enterprise could ever hope to achieve. That is, heavyweight Western systems were far too burdensome, and even Japan's Toyota Production System (TPS) turned corners like a stuck pig. More importantly, the output of lean-agile frameworks was not a finished market-ready product or service, but a lightweight business experiment or minimum viable product (MVP). The goal was to quickly measure market needs with small business experiments and rinse and repeat quickly and inexpensively to validate or invalidate the viability of innovatively new products and services.

Conversely, it was the norm for public sector enterprises to build a single version of a software product or

service in three decades. And, large commercial firms were three times faster, spending a decade and a billion dollars to introduce an innovatively new product or service. Unfortunately, over 90% of large new product and service introductions failed (i.e., "New Coke," "New Pepsi," etc.). Of course, the lesson here is to measure market needs quickly and inexpensively, and rinse-and-repeat often before diving into the deep end of the pool. The business value of speed, talent, intelligence, quality, cognitive abilities, communications, and tacit knowledge only increased with lean-agile frameworks, while traditional heavyweight frameworks attempted to "deskill" the introduction of new products and services.

Basic Lean-Agile Roles and Responsibilities

There are an infinite variety of roles and responsibilities in both traditional and lean-agile frameworks. The basic roles may be project manager, requirements analyst, architect, designer, developer, tester, operator, maintainer, etc. However, this is a gross oversimplification of roles associated with new product and service development. There are other key or essential roles such as performance, interoperability, reliability, availability, security, usability, operability, extensibility, scalability, testability, auditability, and observability engineering. Some sources identify well over 90 specialized roles necessary for the creation of new products and services. Other popular roles are chief information officer, chief technology officer, enterprise architect, portfolio manager, program manager, lean-agile coach, lean-UX designer, cloud architect, DevOps manager, and newly emerging AI specialists as well.

However, let's limit the scope of this discussion concerning lean-agile frameworks to the discussion of two basic roles and responsibilities for illustrative purposes. Let's focus on Product Owners (POs) and Scrummasters (SMs) for the remainder of this analysis. We're going to do this because these are the two basic roles and responsibilities within the Scrum lean-agile framework which over 90% of global IT new product and service teams utilize. Yes, there's a third basic role within Scrum called the Agile Team (or technical developers), but let's save a discussion of that role for another time. And, of course, POs and SMs exemplify the core components of larger lean-agile frameworks like SAFe as well. Yes, it's important to note there are other lean-agile frameworks like DaD, LeSS, S@S, etc. But POs and SMs are also core roles in these frameworks as well.

While it isn't clear in SAFe, S@S makes it explicit that larger new product and development initiatives iteratively or recursively apply Scrum principles, practices, tools, AND roles as they scale upwards (like a Russian Matryoshka Doll). That is, a single Scrum team has a PO and SM, a team of teams has a PO and SM, multi projects or programs have a PO and SM, portfolios have a PO and SM, and enterprises have a PO and SM. So, you can easily see, POs and SMs are key roles within the lean-agile paradigm. And, again, it's important to note, emphasize, and repeat that there are other key roles and responsibilities like security architect, lean UX designers, DevOps architect, cloud architect, AI architect, etc.

And it's important to note that as one scales up to larger product and service development teams or team-of-teams, there may be Chief Product Owners or Product Managers (PMs) and Chief Scrummasters or Release Train Engineers (RTEs) who oversee the work of multiple POs and SMs. Visionary lean-agile consultancies or boutiques easily recognize that lean-agile roles and responsibilities are not an individual but a small team. This is especially true when new products and services become larger and more complex. That is, there may be a team of PMs, RTEs, POs, SMs, and security, Lean UX, DevOps, Cloud, and AI architects. It's important to note that XP's Pair Programming practice comes in handy (i.e., there should be at least two of each role in larger product and service initiatives). Pair programming has a host of benefits such as better decision-making quality, productivity, efficiency, effectiveness, redundancy, sustainability, quality of life, etc.

Product Owner and Scrummaster

So, let's dive directly into the heart of the matter and dissect core lean-agile roles and responsibilities of the Product Owner (PO) and Scrummaster (SM). Here is a verbatim description of POs and SMs from the 2020 Scrum guide. We felt it is important to clearly establish an authoritative baseline of PO and SM roles and responsibilities from an authoritative source. Global firms have greatly expanded, gold-fleeced, and exaggerated the definitions of POs and SMs over the last 20 years. Therefore, let's go back to the basics or fundamentals so as not to lead our audience astray, thus undermining the value, reliability, and validity of this analysis, business case, and set of findings and recommendations too much (or more than is necessary).

Role/*Artifact*	Definition
Product Owner (PO)	**The PO is accountable for maximizing product value resulting from the work of the Scrum Team.** • The PO is also accountable for effective Product Backlog management, which includes: – Developing and explicitly communicating the Product Goal. – Creating and clearly communicating Product Backlog items. – Ordering Product Backlog items. – Ensuring that the Product Backlog is transparent, visible, and understood. • The Product Owner may do the above work or may delegate the responsibility to others. • The Product Owner may represent the needs of many stakeholders in the Product Backlog.
Product Backlog ***– Artifact –***	**The Product Backlog is an emergent and ordered list of what is needed to improve the product.** • The Product Backlog is the single source of work undertaken by the Scrum Team. • Product Backlog items that can be Done by the Scrum Team within one Sprint.
Sprint ***– Timebox –***	**Sprints are the heartbeat of Scrum, where ideas are turned into value.** • They are fixed length events of one month or less to create consistency. • A new Sprint starts immediately after the conclusion of the previous Sprint. • All the work necessary to achieve the Product Goal happens within Sprints.
Sprint Backlog ***– Artifact –***	**The Sprint Backlog is composed of Sprint Goal (why), item subset, and an actionable plan (how).** • The Sprint Backlog is a plan by and for the Developers. • It is a visible picture of work Developers will accomplish to achieve the Sprint Goal. • Consequently, the Sprint Backlog is updated throughout the Sprint as more is learned. • It should have enough detail that they can inspect their progress in the Daily Scrum.
Scrummaster (SM)	**The Scrum Master is accountable for the Scrum framework and Scrum Team effectiveness.** • Scrum Masters are true leaders who serve the Scrum Team and the larger organization: – Coaching the team members in self-management and cross-functionality. – Helping Scrum Team create high-value Increments that meet the Definition of Done. – Causing the removal of impediments to the Scrum Team's progress. – Ensuring all Scrum events take place and are positive, productive, and timeboxed. • The Scrum Master serves the Product Owner in several ways, including: – Help find techniques for effective Product Goal definition and Product Backlog management. – Help the Scrum Team understand the need for clear and concise Product Backlog items. – Help establish empirical product planning for a complex environment. – Facilitating stakeholder collaboration as requested or needed. • The Scrum Master serves the organization in several ways, including: – Leading, training, and coaching the organization in its Scrum adoption. – Planning and advising Scrum implementations within the organization. – Helping employees and stakeholders apply an empirical approach for complex work. – Removing barriers between stakeholders and Scrum Teams.

As you can easily see, these authoritative Scrum definitions of POs, Backlogs, Sprints, and SMs are a bit vague and open to broad interpretation. This isn't necessarily a bad thing as it allows each team, product, service, project, portfolio, enterprise, industry sector, and global enterprise to tailor and adapt Scrum to its specific context. Hence, lean-agile frameworks are not the overly prescriptive step-by-step nightmares their detractors often claim. Rather, they allow a good balance of guardrails, flexibility, and adaptation. That is, they are firm, but flexible, they bend, and yet they yield consistent performance outcomes across teams, products, services, portfolios, enterprises, and global participants.

More importantly, on the surface, like "Lean Thinking" itself, lean-agile frameworks are simple, minimalistic, easy to learn, master, and adapt, etc. In other words, they have a low barrier to entry which is untrue of most traditional heavier weight frameworks. A small lean-agile framework like Scrum can be learned in a few hours at little to no cost and immediately deployed. A traditional framework may take years and hundreds of thousands of dollars to become officially trained, certified, and experienced. That is, traditional frameworks have a high barrier to entry hindering their application, institutionalization, and evolution which take 40 years to promulgate. Conversely, lean agile frameworks spread quickly across the globe in a couple of years.

Clarifying Lean-Agile Roles, Responsibilities, and Assumptions

Product Owner (PO)

Let's begin by explaining, clarifying, and discussing the roles and responsibilities of Product Owners (POs). As we can easily see from the authoritative Scrum Guide, the PO is responsible for creating a Product (or Service) Backlog. The Backlog consists of ordered items. It is a list of stakeholder needs to "improve" (or create a new) product (or service). The Backlog of items (stakeholder needs) must have a Product Goal, and it must be ordered (prioritized from most to least important). That's it!

At its simplest level, a PO may be interpreted as an order taker like a waiter or a takeout order clerk. Someone calls in an order, a work order ticket is drawn up, scoped, and verified, and a price and timeline is given. It's essentially a first in first out (FIFO) queue, but not necessarily, the items are pretty routine and fast, and the stakeholder (customer) picks them up within the prescribed waiting period. The most advanced skill a PO would need at the basic level would be face validity, emotional intelligence, negotiation skills, and the ability for bidirectional communications to stakeholders and developers. Basically, serve as a mediator between stakeholders and developers. At its core, a PO does not need deep product or service understanding, but only what the stakeholders need. It's the developers who must self-organize to understand how the stakeholder needs relate to the product or service under development.

In its most complex form, a PO is more like a Product Manager (PM) who is responsible for identifying market problems and needs, developing business experiments and prototypes, scoping a new product or service, and developing a business model, pricing structure, and budget. PMs develop visions and roadmaps, translate these into well-defined statements or assertions, define a broad solution architecture, and populate backlogs with these outputs. Scrum quickly evolved from its basic waiter or order taking model to the more complex PM model in practice. Today's PMs must be leaders, well educated, highly talented, respected, have proven track records for creating profitable products and services, and bring a complex technical product to market quickly. At its core, the PM needs to have deep market, enterprise, and product or service knowledge (at least at an architectural level). That is, "I need a custom house with 5 bedrooms, three baths, a swimming pool, and a two-car garage with physical and broadband capability (and is solar panel ready)". The PM may even develop or procure the development of a product or service blueprint (detailed design). The PM is essentially the architect while the developers are skilled construction workers.

Product Backlog

There is NO mention of what a Product Backlog and its items are precisely, how to capture them, and what they represent. And, of course, how do you know when Backlog items are complete (to a stakeholder's satisfaction)? The creators of Scrum envisioned a simple list of English statements (i.e., I need a database, I need a user interface, I need a bug fixed, I need a new subsystem, I want you enhance a subsystem, I want to interface to another product, etc.). Are these requirements, user stories, mathematical assertions, wireframes, or some other way of capturing stakeholder needs? Scrum is silent on this issue. People should err on the side of simplicity (first goals) but inevitably make Backlog items more complex than necessary.

Product Backlog items are often expressed as requirements specifications, wireframes, hardware or other domain specific assertions, objectives and key results or OKRs, user stories with basic or machine-readable Gherkin acceptance criteria, etc. A Product Backlog pertains to a single product or service, but Scrum is silent on how complex this product or service may be. There is a sense that is a perpetual list that constantly changes and grows. There is also a sense the PO must keep it up to date, prune it, reprioritize it add, change, or take items away from it; and extend it into the foreseeable future (next Sprint, quarter, years, 5-year planning period, or entire life of a product or service). It is a perpetual list of high priority stakeholder needs which stakeholders are willing to pay for. We fail to mention that Scrum is completely silent on how to prioritize, valuate, measure, size, and manage Product Backlog performance with quantitative measures.

Sprint Backlog

A Sprint Backlog is little more than the items from the Product Backlog that will be developed in a timeboxed period called a Sprint. The Sprint Backlog may simply consist of the highest priority items at the top of the Product Backlog in pure FIFO fashion. Feasibility may be taken into account (i.e., how many Product Backlog items can be done in a Sprint). Furthermore, the Product Backlog selected for a Sprint Backlog may not be in priority order. It may consist of items two, five, seven, and ten, based on the input of a development team. That is, the difference between a Product Backlog and a Sprint Backlog is a matter of negotiation between the PO and the Developers (what can be feasibly done). Furthermore, a single Product Backlog item may need to be decomposed into smaller items, so a single item may take the entire Sprint, depending on its complexity. Like the Product Backlog, the Sprint Backlog should have a goal. There is a sense of feasibility, capacity or availability of developers, and sustainable work pace. That is, one can underfill a backlog to ensure developers aren't burned out or leave room for uncertainty. Or one can fill it to capacity to optimize utilization or simply overfill it with stretch goals to get the most from people in order to optimize stakeholder satisfaction, PO reputation for getting complex things done, and enterprise revenues or profits. Lean-agile proponents suggest underfilling Sprint Backlogs, while aggressive Global 500 enterprises overfill backlogs and keep global labor working 60 to 80-hour workweeks 365 days a year at the lowest possible wage. This latter approach optimizes profits and stimulates the performance bonuses of executives, PMs, and POs who are not averse to establishing unreasonable OKRs for personal gain.

Sprint

A Sprint is a simple timeboxed period of performance. These were initially 30 days, but some organizations stretched these periods to 90 days, 6 months, or longer. Eventually, the global industry standardized around two business or calendar weeks. The global trend is towards one-week Sprints or less or move away from timeboxes altogether. Lean-agile frameworks are designed to be goal-driven. So, the goal may be obtainable in the timebox. For instance, my goal is to build a database, improve the user interface, increase system

performance by 10%, or repair high priority defects. Scrum suggests that Sprints be development oriented. If you're a bricklayer, then the goal is to lay as many bricks as possible. Bricklayers may not spend any time collecting the requirements for the project, developing a blueprint, or developing models or prototypes to validate stakeholder needs. This is the job of the PM, PO, or other upstream personnel. Sprints are now used in a dual-track development mode, where developers may do some stakeholder discovery and some product or service construction in alternating Sprints. And, of course, there is a trend to apply as much automation as possible, including the use of Gen AI, to automatically translate stakeholder needs into finished products or services much like a 3D printer (i.e., input the blueprint and let the machine make the physical output). In this latter case, you'd get the best of both worlds in parallel, developers formulate discovery blueprints while the Gen AI creates finished output for customer evaluation in quick parallel cycles or Sprints.

Scrummaster (SM)

Originally, Scrummasters (SMs) evolved to be Scrum process engineers. They were to understand, facilitate, and apply the Scrum events (Sprint Planning, Daily Scrums, Sprint Reviews, Sprint Retrospectives, Product Backlog Refinement, etc.). SMs were the living embodiment of the Scrum framework. If they were present, then Scrum was performed, if they were absent, Scrum was not performed. Technically, POs and Developers should apply the Scrum frameworks with or without the SM but often skip Scrum events if the SMs are not present (or even stop doing them altogether). This dependence on SMs as the Scrum process robot or automaton has its benefits, because it allows for quick inexpensive Scrum insertion. However, pure process engineers would say the goal of any process framework is to instill the process discipline (events) directly into the developer's behavior. For instance, a parent may drive a young child to school every day, but the ultimate goal is to teach the child to drive so the child is self-sufficient.

Sorry for the digression, but the authoritative Scrum guide says SMs are professional performance coaches. This is like teaching an athlete to reach optimal performance. SMs are also product and service delivery coaches. That is, help developers achieve product goals. They are trainers. And they help identify and remove impediments which are called "blockers" in the common industry profane language or vernacular. Again, think of an SM as a team or sub-team coach. You may have an offensive coach, a defensive coach, a line coach, a receiver coach, a quarterback coach, a kicking coach, etc. The goal is to help developers reach peak performance. For instance, if a developer is assigned a Product Backlog item to fix a database bug, but doesn't know anything about databases, what should the SM do in this scenario? Well, there are a number of options. The SM may teach the developer how to fix the database, get the developer some just-in-time training, ask a knowledgeable developer for assistance, find a Product Backlog item better suited for the developer, etc. The goal, of course, is to move away from individual specialization, because the SM is there to promote cross-skilling among all developers, so they are capable of taking on any Product Backlog items equally. Finally, SMs are viewed as enterprise change agents to help train, coach, and promulgate Scrum across enterprises (i.e., scale Scrum to promote enterprise agility).

Once again, SMs are often the embodiment of the Scrum framework which isn't necessarily a bad thing. They often become meeting schedulers and facilitators, conflict managers, impediment (blocker) managers, agile application lifecycle management (ALM) tool experts, and user story writers. Teams often become overdependent on SMs to perform all of these functions, including serving in the capacity of PM and PO. In the worst case, they are viewed as ALM administrators, ALM dashboard developers and programmers, and user story discovery managers. Well, maybe this isn't the worst case. Often times, they are the "blocker managers." That is, developers stop working at the slightest obstacle and are dependent on SMs to remove obstacles (i.e., my password is locked so I need the SM to unlock it). SMs must coach developers to remove

ordinary blockers and focus on the highest priority ones (i.e., reporting system outages and facilitating system restoration). In a multi-team scenario, a full-time cross-team dependency manager is sorely needed.

It's also important to note there is a lot of role confusion in Scrum. Enterprises often disempower their employees, so they encroach upon one another's territories. POs are disempowered from doing PM duties, so they assume the role of SM. Enterprises can't afford full-time POs and SMs, so developers assume the role of PO and SM. Enterprises can't afford both a PO and SM, so the SM does the role of the PO and the SM. In the worst case, individuals must be PO, SM, and developer. Well, that isn't the worst case. You may have to be CEO, enterprise architect, PM, PO, SM, developer, tester, operator, benefits administrator, hiring manager, and perform occasional janitorial services. The more overlapping roles the better in today's competitive marketplace. Few enterprises have much room for specialized Scrum roles and responsibilities. Sometimes, SMs are disempowered and limited to inessential tasks (i.e., take meeting minutes, update ALM tickets, or writing user stories). Many POs and SMs are required to work 12 to 18 hours a day, 7 days a week, 365 days a year. Scrum has become a workflow system for non-stop global operations like FedEx.

The benefit of Scrum's ambiguity is its generality to any business domain ranging from strategic planning to business operations. Executive, Business Development, Marketing, Legal, Sales, Contracts, Human Resources, Administrative, and Facilities teams now apply Scrum as a general workflow system. Does widespread Scrum institutionalization and application make enterprises more agile? Well, we think the jury is still out on that issue, although Scrum proponents would suggest so. And, as we've emphasized, Scrum's generality makes it wide open to interpretation. It's like having an empty lot and using it in whatever way you prefer (i.e., business, residence, storage, parking, property, religious worship, athletic field, mining, manufacturing, special events, junk yard, etc.). This also leads to role confusion and overlap among POs, SMs, and developers themselves. No two Scrum teams are the same, yet there is a perception that if you know Scrum, then you must instantly know every team's infinite idiosyncratic adaptations.

Scrum is still a tacit knowledge driven approach, and long-standing Scrum teams adapt, adopt, and develop hundreds of hidden rules. One team's minimalistic approach may enable instant success, while another team's hidden rules may hinder any success at all. Probably the most insidious or hideous Scrum adaptation is the transition from a goal-driven workflow system to a scope-driven workflow system. That is, the PO's and SM's success hinges upon completing all planned user stories and story points (plus all unplanned user stories as well). It's like a waiter serving 25 tables at lunch while running to the grocery store for extra food supplies, unclogging toilets, cooking food, and cleaning dishes in parallel. Yes, you are the waiter, with added roles and responsibilities. The more the better to hit enterprise profit targets. **Note**. *We're a little off-track, but hopefully this establishes a solid foundation and context for what comes next*!

Recapping Basic Lean-Agile Roles, Responsibilities, and Assumptions

Let's briefly recap the basic lean-agile roles, responsibilities, and assumptions, especially as they pertain to Product Owners (POs) and Scrummasters (SMs). At their core, POs are enterprise leaders and middle managers. They represent external stakeholders such as customers, end-users, consumers, and specific market segments and business domains. Most of all, POs represent internal stakeholders like C-Level executives such as CEOs, CFOs, CIOs, CTOs, CSOs, etc. POs must have strong foresight, predictive, and future planning abilities. In other words, POs must read minds, read tealeaves, and predict the future. POs must observe and interact with external and internal stakeholders, POs must anticipate their implicit inexpressible tacit needs and body language oftentimes without direct interaction or communications. Like law enforcement criminal profilers, POs must quickly size up their market and enterprise targets, develop

accurate profiles and personas, and know where their stakeholders will be before they get there. POs must think like their constituents, get inside their heads and subconscious minds, predict their next move, and get in front of them with needed products and services before they even know they need them like soothsayers.

Stakeholder needs must be quickly translated and codified as strategic plans, business model canvases, business models, budgets, visions, goals, objectives, roadmaps, integrated master schedules (IMSs), and stakeholder needs or requirements. Then, all of this needs to be distilled into an ongoing Product Backlog, valuated, sized, prioritized, and communicated to technically oriented Agile teams and developers. They must be influencers, communicators, have strong face validity, command respect, be implicitly trustworthy, develop business cases, get valuable funding, get and hold people's attention, inspire all stakeholders, and bend time, space, and reality with stretch goals and objectives. They can be as slow and convincing as they need to be with compelling visions and business cases, or they can be lightning fast with lean-agile frameworks to pummel markets with gold plated MVPs. I think you get the picture, POs are rainmakers, miracle workers, and shaman who entertain people with convincing ideas. What's the bottom line? POs are wickedly smart, talented, skilled, educated, multidisciplinary hands-on unicorns with impeccable face validity who can single-handedly create gold-plated MVPs if necessary. But their greatest feat is forming small cult followings of highly talented developers who can do their bidding for them. They are patient, cold, calculating, even-tempered, and beat you with long term moves like chess grandmasters. And their emotional intelligence is simultaneously capable of influencing customers, executives, and developers.

Skills	Product Owner	Scrummaster
Explicit	• Represent Stakeholder Needs • Create Valuable Ordered Backlog • Product Backlog Performance • Periodically Refine Product Backlog • Up/Down Bi-Directional Communication	• Represent Agile Team • Create Valuable Agile Team • Optimize Agile Team Performance • Optimize Scrum Performance • Promulgate Scrum Across Enterprise
Implicit	• Broad Foresight and Future Focus • Leadership and Management • Long-Term Planning Skills • Emotional Intelligence • Negotiation Skills	• Narrow Present Focus and Execution • Servant Leadership and Followership • Short-Term Planning Skills • Lean-Agile Intelligence • Technical Skills
Advanced	• Market Knowledge • Business Intelligence • User Experience (UX) • Visions and Roadmapping • Business Models, Budgets, and Prices	• Process Knowledge • Automation Intelligence • Developer Experience (DX) • DevSecOps Automation Pipelines • Quantitative/Qualitative Performance

Let's continue our brief recap the basic lean-agile roles, responsibilities, and assumptions, especially as they pertain to Product Owners (POs) and Scrummasters (SMs). At their core, SMs are operational excellence leaders and functional or technical managers. SMs represent internal stakeholders such as product and quality assurance managers, internally facing organizations, and other process and quality engineers. SMs represent internal stakeholders like lean-agile portfolio managers, lean-agile center of excellence directors, chief process engineers, quality management system directors, lean-agile program management offices, etc. SMs have strong hindsight, compliance, near term planning, performance management, execution abilities, and an intense present or day-to-day focus. SMs are responsible for current product and service initiatives, steering the ship to its final destination on time and on budget, and a deep knowledge of historical successes to mimic and failures to avoid (i.e., when we did it like this it worked well and when we did it like that it

didn't work so well). SMs are narrow specialists, lean-agile minimalists by nature, allergic to bureaucracy and overplanning, time driven to a fault, focused on executing and achieving the current plan with precision, think in terms of establishing planning guardrails to avoid overcommitment, and focus on the positive.

SMs must observe and interact with product and service managers, other POs and SMs, engineers and developers, subject matter experts, domain specialists, architects and designers, and other hands-on technicians. SMs are downward, inwards, or peer-to-peer focused and oftentimes suboptimize, but they are cognizant of keeping their peers happy like a well-oiled machine. SMs believe in the power of collaboration, teamwork, cooperation, conversations, and tacit communications. They are less interested in strategies, roadmaps, schedules, business requirements, long-lead items, and anything beyond the past, present, next quarterly period of performance. SMs wanna know what pothole they just hit, fill it in quickly, and avoid potholes just beyond the next turn. To SMs, reading minds, tealeaves, and predicting the future involves so much uncertainty that it's futile to dwell on using black magic to get through the next few Sprints. Like POs, SMs must also anticipate their implicit inexpressible tacit needs and body language oftentimes without direct interaction or communications. However, they are less interested in end-users, customers, markets, and C-level executives, and more interested in the subconscious communications of their peers and developers comprising their agile teams. SMs believe happy developers are productive developers, so SMs are devoted to making developers happy, making their lives easier, and preventing them from becoming overburdened. As servant leaders, SMs silently carry the burden of developers on their shoulders like sheep being led to the slaughter. And like good shepherds, SMs don't wanna lose a single developer to the wolves.

Recapping the Capabilities and Attributes of Neurotypes vs. Neurodivergents

Let's briefly recap the capabilities, attributes, and patterns of neurotypes vs. neurodivergents. Again, our goal is neither to extol the abilities of neurotypes nor denigrate or stereotype neurodivergents. We are simply identifying common patterns found in neurotypes and neurodivergents. Some people believe the difference between neurotypes and neurodivergents is a simple matter of extroversion and introversion. We can certainly understand that overly simplistic assumption, since a hallmark attribute of neurodivergents is introversion. We believe there are a wider range of comorbidities and coping strategies exhibited by neurodivergents beyond introversion (i.e., it's an outward manifestation of multiple physical dysfunctions).

However, we do realize that rather than overshare a detailed manifest of neuropsychiatric attributes, it may be simpler to say, "Sorry for being so quiet, I'm an introvert." That's a simple way of saying, "I cannot interact with you on a wide variety of long-term planning needs in real-time, but I specialize in this subdiscipline." "Furthermore, I'll be happy to send you a brief or case study on lean-agile thinking for your consideration." What we get from the table below is that neurotypes are capable of developing long-term product and service strategies and plans, while neurodivergents prefer highly specialized minimalistic lean-agile process roles with a nearer-term execution or present focus which magnifies strengths and masks weaknesses.

It is important to note that neuroscientists have proven the neuroplasticity of the human brain. That is, the human brain can grow, adapt, and rewire itself at any stage or beginning state. This is true whether one is a neurotype or neurodivergent. However, this is particularly good news for neurodivergents. Rigorous Applied Behavioral Analysis (ABA) has proven to help neurodivergents rewire severely impaired brains for near-neurotypical operating status, especially among very young neurodivergents. Even stroke victims who lose large swaths of their brains are capable of achieving near neurotypical status with rigorous ABA conditioning. And there are other promising techniques like biofeedback, visual therapy, diet, mindfulness, vitamins, etc.

Attribute	Neurotypes	Neurodivergents
Brain Health	Normal	Impaired
Cognitive Speed	Faster	Slower
Synaptic Layers, Neurons	More	Fewer
Cortical Wiring	Fast, High-Bandwidth	Slow, Low-Bandwidth
Memory	Larger, Faster	Smaller, Slower
Raw Data Points	More	Fewer
Context Switching	Faster	Slower
Parallel Processing	Larger	Smaller
Predictive Ability	High	Low
Social Awareness	High	Low
Contingency Planning	Robust, Intuitive	Lean, Explicit
Capabilities, Talents, Skills	More	Fewer
Planning Horizon	Long-Term	Short-Term
Raw IQ	Larger	Smaller
Emotional Intelligence	Strong, Balanced	Weak, Imbalanced
Track Record	Impeccable	Erratic
Temperament	Stable	Erratic
Stakeholder Focus	External, Broad	Internal, Immediate
Leadership Focus	Enterprise, Management	Team, Technical
Leadership Style	Delegate, Lead	Servant, Follow
Decision Style	Autocratic	Consensus
Business Focus	Strategic Planning	Operational Excellence
Personality	Extrovert	Introvert
Social Style	Group	One-on-One
Thinking Style	Verbal, External	Non-Verbal, Analysis
Communication Style	Indirect, Suggestive	Direct, Blunt
Communication Type	Voice, Conversation	Written, Electronic
Communication Mode	Synchronous	Asynchronous
Social Reciprocity	Small Talk	Monologue
Career Type	Generalist	Specialist
Goal Orientation	Long-Term	Near-Term
Activities	Broad	Narrow
Routine	Non-Repetitive	Repetitive
Perseverance	Low	High
Functional Role	Product	Process
Anxiety	Low	High
Work Schedule	24x7 Zombie Style	Part-Time, Nine-to-Five

Comparing and Contrasting Lean-Agile Roles for Neurotypes vs. Neurodivergents

Let's quickly attempt to link the major lean-agile roles and responsibilities with the capabilities of neurotypes vs. neurodivergents. Again, as we've argued over and over again, neurotypes are extroverted, customer facing, prefer reams of analytical data about market predictions, and are typically multidisciplinary parallel processing unicorns. Furthermore, neurotypes are uniquely skilled at developing meticulously detailed long-term strategies, roadmaps, and integrated master schedules with task-level dependencies spanning years. The benefits to meticulous long-term market predictions are investors require these business cases before committing to, or allocating, scarce enterprise resources such as capital, human capital, and excess profits. Of course, the weaknesses to meticulous long-term market predictions are that they are highly volatile, unpredictable, uncertain, and risk intensive assumptions. Customer and market requirements exist as hidden inexpressible subconscious needs. They simply do not know what they want until they see it. Developing a decade long multi-billion-dollar product or service is fraught with risk. Over 95% of new initiatives fail. But if someone is going to be a soothsayer, then neurotypicals are particularly convincing at detailed projections. Let neurodivergents come in behind neurotypes to clean up their predictive mess (which is their strength).

Lean-Agile Role	Attribute	Neurotype	Neurodivergent
Product Owner – PO –	Leadership	✓	✗
	Predictability	✓	✗
	Longterm Horizon	✓	✗
	Longterm Planning	✓	✗
	Multidisciplinary	✓	✗
	Quick Learner	✓	✗
	Social Extrovert	✓	✗
	Mindset	✓	✗
Scrummaster – SM –	Operational Excellence	✗	✓
	Execution	✗	✓
	Servant Leadership	✗	✓
	Lean-Agile Thinking	✗	✓
	Goal vs. Scope Driven*	✗	✓
	Minimalistic Mindset	✗	✓
	Sustainable Workpace	✗	✓
	Dev Experience (DX)	✗	✓

** Neurotypes misuse goals, i.e., achieve 150% of my scope with 50% of the resources so I can get promoted this quarter*
** Neurotypical commitment means accomplish all of your user stories and all of my stretch goals and honeydos for my benefit*
** Beware of neurotypical wolves in sheep's clothing espousing scope-driven OKRs to establish global legacy system sweatshops*

As far as neurodivergents are concerned, they are internally, near-term, operational excellence, lean-thinking, (genuinely) goal-driven, minimalistic, and employee morale focused people. They have fewer capabilities than neurotypes, oftentimes narrowly specialized process experts, and are particularly adept and well suited for perseverance, repetitiveness, and highly routine near-term sprint-to-sprint planning. They are particularly adept at (genuine) goal vs. scope orientation in the form of small business experiments, minimal viable products (MVPs), and other market probes to smoke out hidden inexpressible subconscious customer requirements and needs. As empaths, they wanna please developers, employees, and workers instead of customers and executives who work night and day to hit their scope-driven legacy system stretch OKRs.

Both neurotypical POs and neurodivergent SMs are necessary, as they form a symbiotic relationship needed for global enterprise success. Neurodivergents may not be optimal meticulous long-term POs. However, neurodivergents may be very good short-term sprint-to-sprint or release-to-release POs as long as neurotypicals develop long-term roadmaps, epics, and lightweight integrated master schedules. And it's also important to note that there are more neurodivergents than neurotypes, it's crowded at the bottom, and there is much competition and overlap between POs and SMs in short-term execution focused activities.

And, as we've emphasized, both neurotypicals as well as neurodivergents may use a variety of learning, therapeutic, meditative, and medicinal approaches to adapt their brains beyond highly constrained or limited contexts. That is, it may be possible for neurodivergents to grow, expand, and adapt to lean-agile roles typically reserved for neuro-biologically stronger neurotypes. Many neurodivergents are slower learners than neurotypes and absorb new information at a slower rate of speed. That being said, given enough time, patience, and therapeutic focus, neurodivergents may be able to grow into PO assistants or deputies with longer-term, meticulously detailed principles, practices, measures, and tools. And, as we've alluded to before, neurodivergents are good at untangling the morass created by neurotypical soothsayers. Neurodivergents make good cleaners, fixers, repair people, and are good at hiding bodies along the way.

Early Recommendations and Emerging Strategy

Software leaders have been implicitly and explicitly grappling, wrestling, and untangling the neurodivergence quagmire since the 1960s. Some computer programmers coded 100,000 lines in a few months by themselves. Other computer programmers couldn't code at all. Usually, only 1% of computer programmers coded effectively and efficiently. So, mainframe computer programming managers adopted, adapted, refined, and expanded waterfall lifecycles, which eventually became ISO/IEC 15288, ISO/IEC 12207, ISO 9001, CMMI, PMBoK, and a host of other public and private sector adaptations for monolithic computer programming projects. Their purpose was to slow the process down, make everyone create mountains of documents, and chase off the faster computer programmers who could code 100,000 lines in a few months. Conversely, the strategy of small commercial firms was to hire the fastest 1% of computer programmers to code 100,000 lines in a few months, ignore the waterfall lifecycles and documentation, and also ignore the 99% who couldn't code. Noncoders often became testers, cybersecurity engineers, project managers, etc. It's not unusual for 80% to 90% of IT staff to be comprised of non-coders even today.

What's the bottom line? Mainframe computer programming managers have been trying to solve this puzzle for 70 years. That is, understand why the productivity between a neurodivergent workforce ranges so widely and what to do about it? This gave rise to global software factories. Do we slow the process down with process and document driven waterfall lifecycles, do we let the coding cowboys loose in the Wild West, or is there a middle ground of lean-agile frameworks with just enough discipline and flexibility to blend neurotypes and neurodivergents into a coherent multi-disciplinary team. Some people feel lean-agile frameworks for large computer programming projects are simply too slow. Coding experts feel lean-agile frameworks are too bureaucratic and simply shift the burden of project management onto them. Although this treatise is exploratory in nature and not designed to definitively identify the variables for successfully integrating neurotypes and neurodivergents into fast-paced lean-agile frameworks, there is some light at the end of the tunnel. Much of this is hindsight. Modern IT such as the Web, PCs, tablets, networks, email, collaborative tools, texting, web-based videos, and other instructional aids implicitly level the playing field.

Approach	Description
Primary	• Apply lean-agile frameworks – Limit the WIP. • Untangle the legacy code monoliths – Make them easier. • Move away from brick-n-mortar data centers to Cloud data centers. • Move towards Scrumban – No planning, no sprints, no estimation, etc. • Move towards small business experiments and MVPs. • Use web-based video software documentation for refactoring/maintenance. • Apply easy-to-use AI-enhanced ALMs and Integrated Development Environments.
Secondary	• Assess people's capabilities – Know their strengths and weaknesses. • Use softer mentoring as neurodivergents have grown beyond the tough-love phase. • Apply asynchronous technology, PCs, laptops, tablets, web, email, texting, etc. • Create and use libraries of video/audio tutorials to augment neurodivergence. • Adopt, apply, and use personal AI assistants, companions, and automated tutors. • Use teamwork such as pair programming – Reward teams vs. individual effort. • Promote diet, brain health, exercise, vitamins, supplements, and other health aids.

Footnote. The goal is to apply "Lean Thinking" values, principles, practices, and tools to reduce batch size, limit the WIP, reduce multitasking, use one-piece workflow, automate everything, and use visual aids as much as possible. In other words, utilize neurodivergents as much as possible to create a win-win scenario. *The answer may not be neurotypical POs who predict the future with task-level dependency analysis after all.*

Other Considerations

Global firms experience a wide variety of market challenges, such as competing on short lead and cycle times using lean-agile frameworks to disrupt markets with innovatively new products and services. There are also other challenges and constraints that we haven't mentioned so far. That is, 90% of global IT budgets are wrapped up in multi-decade old legacy code monoliths. A single change, enhancement, modification, or bug fix may take hundreds of people working night and day across the globe at rock bottom prices. To an executive, this activity may seem like a simple OKR so I can get my $50,000 annual bonus. Behind that scope-driven OKR is 300 to 500 people working 24 hours a day in global sweatshops commuting across dangerous and densely populated cities away from the safety of their families, loved ones, and homes, to untangle these legacy code monoliths. It takes a Western IT executive 15 minutes to write a scope driven OKR disguised as a goal driven OKR which will enslave 500 people for the next 90 days at rock bottom prices untangling the code monolith to make a simple code repair (all the while successfully obtaining the exorbitant bonus). The better OKR would be, "Untangle a single subsystem or capability of a mission critical legacy system of systems this quarter and decompose it into modularized microservices so that we can improve developer and user experience by 50% with quick and easy design sprints in future quarters."

Now, that may still require 500 global sweatshop workers in the first quarter, but in subsequent quarters, microservices updates may be performed at a sustainable pace. Other closely interrelated challenges include the use of physical capital-intensive brick-n-mortar on-premises data centers which are extremely hard to build, operate, and maintain (vs. virtual cloud-based data centers that can be composed by single programmer in minutes). So, a legacy OKR may state, "Build a complex brick-n-mortar data center in 90 days" which may require a small army of people to do two years' worth of labor in 90 days. But, a future OKR may be, "Create a virtual cloud-based data center MVP in 90 days which will reduce future operations and maintenance costs of application hosting by 90%." OKRs have no inherent ethics and morals and neither do the Western executives who routinely abuse scope-driven OKRs to enslave global workers around the clock at cut-rate prices to obtain their annual bonuses. Originally, IT enterprises were to have 5 to 7 OKRs in total for the entire enterprise. Today, there are OKR tools and modern enterprises have thousands of OKRs for every conceivable purpose. It's one thing to have an unethical OKR, but thousands of them? Global sweatshop OKRs place an enormous burden on a neurodivergent workforce, the lean-agile frameworks they apply, and the lean-agile roles and responsibilities that must be performed efficiently and effectively!

Summary

So, what have we examined, analyzed, and determined in this treatise? We've determined that global firms apply lean-agile frameworks like XP, Scrum, Scrumban, and SAFe to introduce innovatively new products and services with short lead and cycle times to disrupt global markets. That is, increase revenues and profits while undermining global competitors or increasing market share. We've also alluded to the notion that lean-agile frameworks are goal vs. scope driven although most global firms apply scope-driven OKRs along with lean-agile frameworks. We've even hinted at the notion that complex legacy systems and physical brick-n-mortar data centers (or long lead items) encourage the formation of scope-driven goals and misapplication of the lean-agile frameworks themselves. We've also pointed out that misappropriation of lean-agile frameworks for scope-driven OKRs leads to the formation and enslavement of global 24x7 sweatshops, while Western executives collect quarterly bonuses like Wall Street rainmakers, magicians, soothsayers, and shaman. In a prior case study, we suggested lean-agile frameworks may be well suited for neurodivergents under special circumstances and constraints such as backend teams of larger, slower moving public sector product and service initiatives where technical specialists are highly valued over customer facing neurotypes.

In this analysis, we've further delineated the constraints or sliced the pie to examine the major roles of lean-agile frameworks like Scrum's and SAFe's Product Owners (POs) and Scrummasters (SMs). There has always been some ambiguity, controversy, polarization, and even outright rejection of the PO role and lean-agile frameworks themselves. That is, is the PO a simple order taker like a waiter who works sprint-to-sprint which doesn't place too many cognitive constraints on lower-performing neurodivergents? In fact, we have seen neurodivergents succeed as sprint-to-sprint POs with flying colors. However, from the very beginning, global enterprises have stretched, reconstituted, and adapted the PO role to be more of an enterprise Product Manager (PM). PM jobs are wickedly complex requiring extremely capable neurotypes to analyze global markets in excruciating detail, develop detailed business strategies, and develop meticulously detailed integrated master schedules with task level dependencies spanning years. The subtle transition of the PO from simple order taker to enterprise soothsaying PM places enormous cognitive constraints on individuals where neurotypes perform best. Some say neurotypical PM unicorns will survive a dystopian AI Winter.

This transition is a little bit like boiling a frog. If you place a frog in boiling water, it will jump out and save itself. That's a little like throwing a neurodivergent into an enterprise PM role without much thought. However, if you put the frog in neutral water and slowly raise the temperature to a boiling point, the frog may not sense the danger quickly enough and will eventually get boiled and die. That's essentially what we've done with the PO role. We've raised the temperature slowly in the last 20 years, so now jumping into a PO role is like jumping into a pan of boiling water. We've illustrated that the PO role now requires meticulous planning, a long-term outlook and mindset, and the ability to capture and parallel process thousands of data points about customers, markets, enterprises, and stakeholders. Basically, neurotypes can instinctively solve complex problems with thousands of variables and equations very quickly. Today's PO role has grown beyond the constraints of a historical SM, while the basic SM is still an internal team-facing function with simple constraints. This simplicity may make the SM roles better suited for neurodivergents than today's complex PM-like PO roles and responsibilities. However, as we've seen from the Scrum guide, the SM is responsible for promulgating Scrum across the entire enterprise causing cognitive overload.

Like the PO role, enterprise coaches also place enormous constraints upon individuals. Our ultimate goal is to help find the optimal lean-agile roles and responsibilities in which they may be minimally successful. Conversely, we wish to identify the lean-agile roles that require advanced cognitive abilities. Furthermore, we've offered promising hope for the future of neurodivergents. This includes ABA techniques for behavioral modeling, biofeedback, visual therapy, deep brain stimulation, diet, mindfulness, vitamins, etc. Lean-agile proponents are weaning global enterprises off of decade long PM initiatives and larger enterprises in favor of emergent sprint-to-sprint Lean UX businesses and experiments where neurodivergents may once again return to basic PO roles and responsibilities. We've also hinted that the key to enabling short term business experiments is to decompose legacy system monoliths, develop virtual cloud-based data centers, and create highly efficient and effective DevSecOps pipelines. Of course, we can always do what Netflix does and gobble up the globe's advanced neurotypes, but this is a zero-sum gain. The real future is successfully employing the globe's neurodivergent workforce. And we don't mean using unethical OKRs to enslave global workers in 24x7 sweatshops at rock bottom prices to obtain selfishly exorbitant bonuses every 90 days.

In reality, today's global enterprises employ widely ranging neurodivergents placing enormous cognitive load on individuals. That is, integrating high performance neurotypes and lower performing neurodivergents into the same team can be quite challenging for both people. Neurotypes will never adapt to the slower moving neurodivergents and neurodivergents will never adapt to neurotypes who report task level dependencies 16 releases into the future in daily Scrums and other key lean-agile events like quarterly planning, PO and SM synchs, system demos, retrospectives, and other key lean-agile events. The cognitive burden is quite high.

Limitations

As we mentioned in a prior case study, the American Psychological Association (APA) is a bit divided and ambiguous on the precise causes, definition, and symptomology of neurodivergence. This alone hinders the proper identification of treatment protocols for neurodivergents, leaving the victims of neurodivergence to their own devices. A closely related phenomenon is Solution Focused Behavioral Therapy (SFBT) which often leads neurodivergents to quick fixes once they clearly identify their major mental health issues. A popular approach for managing neurodivergence is mindfulness, which certainly has its benefits. A more aggressive approach to managing neurodivergence is Applied Behavioral Therapy (ABA) which involves rigid behavioral conditioning protocols not unlike strict military training for young children. The human brain is remarkably neuroplastic, and highly structured and intense conditioning helps shape, mold, and form new human behaviors. As the old saying goes, "Once a Marine, always a Marine, " which suggests rigid military behavioral shaping and conditioning establishes well-structured lifelong routines, heuristics, and rules for daily living. A U.S. Navy Admiral once admonished his audience in a keynote speech with the phrase, "The first thing you must do when you get up in the morning is make your bed." And, of course, the Bible suggests rigid discipline with the controversial admonishment, "Spare the Rod, Spoil the Child."

Others insist that any sort of involuntary behavioral shaping is simply psychological and physical abuse. Of course, the most important point is that the field of neurodivergence is still young, many of these treatment protocols are still in their infancy and are stopgap measures, which SFBT completely confesses to and admits (i.e., sometimes the stopgap measure is the "miracle cure"). Many neuroscientists are still unconvinced that mindfulness and behavioral therapies have any lasting, consistent, or positive effect on leveraging neuroplasticity for shaping positive human behaviors. That is, they believe the human brain is a simple chemical (hormonal) state machine and thoughts and actions are subconsciously formed in microseconds before humans can choose a rational course of action. While we are not debating the role of the human subconscious mind, we believe it can also be managed, shaped, and conditioned so that we are far less (subconsciously) impulsive. Some brain states, conditions, or anomalies may need more aggressive treatment such as surgery, powerful pharmaceuticals, and other vitamins, minerals, and supplements. This leads us to the next point about the human diet. The Western diet is glucose rich which may exacerbate neurodivergent comorbidities, so a possible solution may be glucose free diets. Blood sugar levels and hypoglycemia may also mimic or enhance neurodivergent behaviors, along with other hormonal disorders. Common viruses, bacteria, and parasites also mimic neurodivergent behaviors such as toxoplasmosis. We've failed to mention epigenetics causing cross-generational physical and behavioral anomalies.

Given the wide variety of sources of neurodivergent comorbidities, symptoms, and suboptimizing coping strategies, it's no wonder the APA has been totally perplexed by the field of neurodivergence for more than a century. No two are alike and neurodivergents just have to keep chipping away until they find their goldilocks zone, sweet spot, or center of percussion. Likewise, no two agile teams and adaptations are precisely the same, while pundits argue that agile teams are exactly the same "IF" you really know what you're doing (i.e., you have legitimate training, qualifications, and experience). We have "NEVER" found two agile teams or enterprises that are exactly the same, heuristics vary from team to team, some may have few heuristics while others have hundreds. So, the challenge is "NOT" a lack of legitimate training, qualifications, and experience, but rather the (neurotypical) cognitive speed, capability, and talent to recognize a wide variety of agile heuristics and quickly, intuitively, and instinctively grasp the differences and execute a new set of heuristics as though you've been doing it all along. We like to experience a new agile team's specific heuristics firsthand by wading into the pool with the agile teams, while neurotypes attempt to predict the agile heuristics of teams sight unseen with a few quick conversations, chitchats, and predictions. Predicting the future and rapidly adapting to new environmental conditions is the hallmark trait of neurotypes.

For neurotypes, if you've seen one agile team you've seen them all, while neurodivergents tend to count the unique number of angels on the head of each agile team. Robin Dunbar suggested the human brain evolved as a conversation or gossip machine which neurotypes exemplify and neurodivergents don't. So, like neurogivergents themselves, heuristics vary widely from agile team to team. Therefore, it's very difficult to stereotype agile teams and neurodivergents. If agile teams and neurodivergents are highly idiosyncratic, then it may be difficult to develop a consistent set of principles, practices, and techniques in which neurodivergents may succeed. That is, is it reliable and valid to suggest only neurotypes can be successful POs and only neurodivergents can be successful SMs? We may discuss neurodivergents who are frontal lobe locked and may be better at forecasting in a future case study. To round out this discussion of limitations, we must mention another lean-agile framework such as Kanban which is a pull-driven system devoid of POs, future forecasting, prediction, long term roadmaps and plans, and other trappings of traditional waterfall and product management disciplines. If it is true that a lean-agile framework like Kanban can help agile teams be successful with its intense focus on day-to-day operations without any prediction, forecasting, or planning, which is highly doubtful, then perhaps Scrumban or Kanban may be even better suited lean-agile frameworks for neurodivergents. However, we must caution that even pull-driven Kanban systems will fail or starve without a continuous flow of orders from strategic plans, roadmaps, integrated master schedules, and product backlogs. So, once again, it's highly doubtful that any lean-agile framework can truly succeed without the hallmark predictive abilities of neurotypes with robust out-of-the-box cognitive abilities.

Further Reading

- Brechner, E. (2015). Agile project management with kanban. Redmond, WA: Microsoft Press.
- Hastings, R., & Meyer, E. (2020). No rules rules: Netflix and the culture of reinvention. New York, NY: Penguin Press.
- Leffingwell, D. (2023). Scaled agile framework (SAFe) 6.0. Boulder, CO: Scaled Agile, Inc.
- Lovaas, O. I. (1981). Teaching developmentally disabled children: The me book. Austin, TX: Pro Ed.
- Lovaas, O. I. (1997). The autistic child: Language development through behavior modification. New York, NY: Irvington Publishers.
- Lovaas, O. I. (2003). Teaching individuals with developmental delays: Basic intervention techniques. Austin, TX: Pro Ed.
- Maurice, C. (1993). Let me hear your voice: A family's triumph over autism. New York, NY: Random House Publishing.
- Maurice, C. (1996). Behavioral intervention for young children with autism: A manual for parents and professionals. Austin, TX: Pro Ed.
- Pichler, R. (2010). Agile product management with scrum: Creating products that customers love. Upper Saddle River, NJ. Addison-Wesley.
- Reddy, A. (2016). Scrumban revolution: Getting the most out of agile, scrum, and lean kanban. New York, NY: Addison-Wesley.
- Rupp, C. G. (2020). Scaling scrum across modern enterprises: Implement scrum and lean-agile techniques across complex products, portfolios, and programs in large organizations. Birmingham, UK: Packt Publishing.
- Schrage, M. (2014). The innovator's hypothesis: How cheap experiments are worth more than good ideas. Boston, MA: MIT Press.
- Schwaber, K. (2004). Agile project management with scrum. Redmond, WA: Microsoft Press.
- Schwaber, K., & Beedle, M. (2001). Agile software development with scrum. Upper Saddle River, NJ: Prentice-Hall.
- Schwaber, K., & Sutherland, J. (2020). The scrum guide: The definitive guide to scrum (the rules of the game). Mountain View, CA: Creative Commons.
- Schwartz, J. M., & Begley, S. (2002). The mind and the brain: Neuroplasticity and the power of mental force. New York, NY: Harper Collins Publishers.
- Wilmshurst, D. (2023). SAFe coaches handbook: Proven tips and techniques for launching and running SAFe teams, ARTs, and portfolios in an agile enterprise. Birmingham, UK: Packt Publishing.

Actual Case Studies

Team A. *Team A was a large multi-decade long legacy system modernization effort using lean-agile frameworks for managing dozens of agile teams. A neurodivergent was asked to be the RTE (Chief Scrummaster) of all of the teams. The neurodivergent intuitively knew managing dozens of teams was a 24x7 job, which it was, and was well beyond her cognitive abilities, so she politely declined. Instead, she was assigned to be the SM of two of the software development teams. She felt comfortable enough with Scrum to accept the job, knowing that each Scrum team was unique. The nice thing about it, is that the team of teams had instituted basic Scrum events years in advance, and jumping into the SM role would be fairly seamless. An initial obstacle would be using a new ALM system, which she was shown how to use very quickly. The larger obstacle was that she would have to facilitate quarterly planning for the two teams (i.e., serve as a joint or surrogate product manager/owner). Fortunately, a senior product manager joined her first quarterly planning event and helped her prepare quarterly plans. The other nice thing was the simple Confluence based approach to quarterly planning. Each team only had to fill in a single table. While it looked effortless on the surface, a lot of near-term foresight was necessary to populate it. She helped the two teams prepare quarterly plans, sprint plans, and facilitate basic Scrum events for five quarters. Although she was both the PO and the SM, it became quite routine and required a fraction of her limited cognition. The only real cognitive burden was the week of quarterly planning when it was time to herd the cats, formulate a quarter's worth of effort, and get approval from senior leadership. This was a prime example of near-term focus, dependence on others to do long-term planning, and routine minimalistic Scrum events. Her teams were some of the most productive, and she coached the teams to be self-organizing.*

Team B. *Team B was also a large multi-decade long legacy system modernization effort using lean-agile frameworks. A neurodivergent was asked to be SM of the backend infrastructure teams. She was told the biggest challenge would be mastering the use of multiple ALM systems for quarterly and sprint planning and execution. The neurodivergent was a bit slow on the uptake and struggled mastering new ALM systems. After some fits and starts, multiple projects, and multiple ALM systems, learning and using new ALM systems became easier. She was about 10 to 15 years behind her peers in terms of Scrum and ALMs, but she was finally gaining some momentum. The thought of being an SM of three teams sight unseen and juggling multiple new ALM systems didn't seem very daunting. She was happy to be away from the leadership team where the neurotypes sliced and diced each other on a daily basis. They'd been together for years and had already formed, normed, stormed, and performed, but the neurodivergent hadn't. So, jumping into an SM role with backend teams in a data center far away from the neurotypes was a great relief. She took charge of the three teams immediately; informally assessed the climate, maturity, readiness, and tolerance of Scrum; and got to work as quickly as possible. The backend teams had been using Scrum for a few years but used only the bare minimum events possible. They were not Scrumbutts but had transitioned through the Shu-Ha-Ri Scrum adoption cycle very quickly. After some initial fits and starts, she welcomed the minimalistic Scrum practices. She focused on mastering the new ALM systems. There were four in total. Like her earlier engagement, she was responsible for preparing quarterly and sprint plans. The POs were more interested in being developers. She maintained virtual cognitive interfaces between the disconnected ALM systems that perplexed parallel processing neurotypes which didn't involve very much foresight.*

Team C. *Team C was also a large multi-decade long legacy system modernization effort. A neurodivergent was asked to be the PO of a single middleware team. Although she'd served as PO multiple times before, including joint PO and SM on her prior two engagements, something told her she would be in over her head as a PO. She'd been asked to be a PO once before, but somehow knew being a PO would test her cognitive abilities quite a bit. She'd felt it would be a matter of domain expertise, knowledge, and information mastery, all of which overloaded her cognitive abilities. Her previous PO and SM roles had involved little to no domain expertise, quarterly and sprint planning, and little long-term planning abilities. She reluctantly took the new PO job, refamiliarized herself with the literature of POs, and scripted daily routines. None of the materials she consumed captured the essence of being a PO. Previous POs had forgotten to tell her about unique adaptations and heuristics. Therefore, she stepped on big landmines in spite of her meticulous preparations (i.e., "we don't do things like that, we to things like this instead, why didn't you know that?"). Senior leaders wanted her to formulate long-term strategic plans, roadmaps, and integrated master schedules. They wanted her to refine product backlogs six quarters into the future. They wanted her to predict the future several years in advance and tell senior leadership where the potholes in the road would be. Was that really in the Scrum Guide? Was that really the job of the PO to identify potholes six releases out? Was she really to ignore Sprint execution? She'd been asked to do multiyear task-level dependency analysis before but shied away from such nonsense. These subtle requests just kept on sneaking up on her in spite of fleeing from these types of roles. None of the Scrum materials prepared her for this expectation. Her Scrum team was the lowest performing group due to lack of daily coaching.*