Tra·di·tion·al • Team (*trə-dĭsh'ə-nəl • tēm*). A large, collocated group of fiercely individualistic and highly talented people formed to compete with other groups. Examples include a large group of talented engineers designing a complex, risky, expensive new rocket, spacecraft, automobile, building, bridge, computer system, or software operating system over a very long period of time in competition with other groups. Common terms in the first couple of sentences include large, complex, risky, expensive, long-term, highly talented, individuals, and competition, etc. Each one of these terms involves a high-level of risk, uncertainty, and unpredictability and most traditional teams fail to produce a new product or service under these conditions. Of course, the victims of these risks are markets, customers, end-users, investors, organizations, environment, and the people themselves. Traditional thinking methods, practices, tools, and metrics emerged to help manage these risk factors such as integrated master schedules (IMSs), earned value management (EVM), systems and software engineering process and document-intensive waterfall lifecycles, governance boards, business requirements, enterprise architectures, and a plethora of metrics to quantify time, cost, productivity, and technical performance. Although the original list of risk factors was enough to kill projects dead from the start, the traditional thinking metrics, practices, tools, and metrics were the final nail in the coffin of certain project failure. In other words, the cure was worse than the disease. Teams were often so big that new buildings and skyscrapers were often built to house or collocate the entire group, which was not always possible. Putting fiercely individualistic managers in different buildings, floors, offices, and cubicles with heavyweight traditional practices didn't solve the problem of immense size and complexity (only making it worse).

Dis·trib·ut·ed • Team (*dĭ-strĭb'yo͞o'təd • tēm*). A large group of fiercely individualistic and highly talented people formed to compete with other groups distributed across a city, state, region, nation, or globe. Like traditional collocated teams, distributed teams sought to take a global divide-conquer-approach to building complex new products, services, and systems. There were a variety of reasons for doing so as talented engineers were in short supply, disparate organizations mastered highly specialized core competencies and market segments, global and domestic supply and demand fluctuated rapidly limiting the ability to maintain a single large workforce, and of course low-cost international labor rates provided a competitive advantage. Once again, large domestic and international teams relied upon traditional thinking methods, practices, tools, and metrics to help manage immense global complexity, which quite frankly didn't help at all. If fierce Western individualism served as a barrier to communication and collaboration, now spread your team across 10, 15, or 20 countries in 24 different time zones, languages, cultures, customs, and infrastructures. Costs, lead times, and cycle times were measured in billions of dollars and decades, complexity and risk were inordinately high, processes and documentation were measured in tons, and success rates were inordinately low. For instance, over 25,000 engineers in 15 countries built NASA's International Space Station (ISS) over 40 years at a cost of $150 billion. The initial decade was spent creating a mountain of business requirements at a cost of $14 billion, until its stakeholders forced NASA to abandon traditional thinking practices to begin inserting real hardware into space in short iterations. Due to a dearth of engineers, Strategic Defense Initiative (SDI) or Star Wars failed due to starvation of domestic engineers.

Ag·ile • Dis·trib·ut·ed • Team (*ăj'əl • dĭ-strĭb'yo͞o'təd • tēm*). A small, distributed team of moderately talented complementary knowledge workers using close communication and collaboration to tease out uncertainty and risk by building simple, inexpensive, low risk, and low WIP business experiments in short iterations in an effort to successfully field innovatively new products and services with short lead and cycle times. Notice the terms in this definition—Small, team, moderately talented, complementary skills, close communication and collaboration, simple, inexpensive, low-WIP, business experiment, short iterations, success, and short lead and cycle times. Everything about these terms exudes low risk, risk management, risk aversion, and success. Unlike traditional teams or traditional distributed teams, agile distributed teams do not build overly complex products and services with long lead and cycle times, and they do not employ heavyweight traditional thinking methods, practices, tools, and metrics that introduce even more cost, risk, and time into the equation. Whereas traditional teams exponentially increased cost, risk, and failure by multiplying complexity and WIP, agile distributed teams exponentially decrease cost, risk, and failure by decimating complexity and WIP. Like traditional distributed teams, agile distributed teams often seek to take advantage of global engineering talent, disparate organizations with highly specialized core competencies and market segments, buffering wildly unpredictable supply and demand with elastic virtual enterprises and workforces, and of course low-cost international labor rates for a competitive advantage. However, agile distributed teams are still faced with certain immutable challenges such as barriers to communication, collaboration, synchronization, consistency, and successful execution across time, space, and cultures.

So, the solution or roadmap to the challenge of successfully building innovatively new products and services has been established by these basic definitions. Apply lean thinking principles, agile methods, small groups, intensive teamwork and collaboration, and simple low-WIP, low-cost business experiments to tease out uncertainty and risk in short iterations. Conversely, avoid large teams of uncooperative individuals; avoid building expensive, complex, and over scoped systems; and avoid traditional process and document intensive waterfall systems. Oh yeah, and ditch the integrated master schedules (IMSs), earned value management (EVM), business requirements, enterprise architectures, and traditional metrics. It's as simple as that—Well, no, not really! As easy as this prescription, antidote-to-failure, or get-well-plan may seem on the surface, it's a lot harder to achieve than one can imagine. But, why? As Albert Einstein said, "Any intelligent fool can make things bigger and more complex, but it takes a touch of genius and a lot of courage to move in the opposite direction!" Traditional thinking permeates the new product and service development space as it has for over 100 years, humans naturally want to build complex expensive systems out of delusions of grandeur and personal aggrandizement, and they deeply trust traditional thinking values, principles, and practices. Humans naturally believe they can divine the future in the form of IMSs, EVM, business requirements, enterprise architectures, and they instinctively distrust lean and agile thinking. Therein lies the basic rub and its antidote or solution. Trust and apply lean and agile thinking, distrust traditional thinking, and use small groups, intensive teamwork and collaboration, and simple low-WIP, low-cost business experiments to tease out uncertainty and risk in short iterations. However, this is a lot easier said than done, especially with distributed teams, so let's outline a few basic principles to successfully apply lean and agile thinking to distributed teams.

Lean-Agile Distributed Principles & Practices

1. Lean • Think·ing (*lēn • thĭng'kĭng*) Thin, slim, trim, light, fast; To apply pull-based, just-in-time, and waste-free product and service delivery principles

   ✓ Apply lean thinking principles.
   ✓ Pull, demand-based, just-in-time.
   ✓ Optimize end-to-end flow and speed.

   The single most important principle of lean-agile distributed projects and teams is lean-thinking at all levels, from the client to the provider. This purpose of lean thinking is to quickly deliver innovatively new products and services to markets, customers, and end-users with the shortest possible lead and cycle times. It's a pull vs. push driven philosophy that's fundamentally driven by market, customer, and end-user requirements, demands, and needs. That is, suppliers and providers are "listening" to the market, customers, and end-users (taking orders and requests); packaging high-quality products and services; and quickly delivering them, measuring feedback, and continuously improving (as it's often difficult to get it right the first time). Most of the time, product and service specifications exist as tacit, hidden, and inexpressible customer, market, and end-user needs, therefore it requires many rapid rinse-and-repeat cycles to get-it-right, satisfy those needs, and even delight the socks off customers, markets, and end-users. According to the Lean Enterprise Institute, lean thinking involves identifying value, mapping the value stream, creating flow, establishing pull, and seeking perfection (continuous improvement). A similar set of lean thinking principles is eliminating waste, building quality in, creating knowledge, deferring commitment, delivering fast, respecting people, and optimizing the whole. And, according to Scaled Agile, Inc., their house of lean includes, value, respect for people and culture, flow innovation, continuous improvement, and lean-agile leadership. In practical terms, this means small teams empowered to listen to their customers directly, quickly forming minimum viable products (MVPs) with little waste, rapidly delivering innovations to customers, measuring their feedback, rinsing-and-repeating many times in rapid succession until markets, customers, and end-users are utterly delighted; and, of course, the psychological safety that comes from an enjoyable job, sustainable pace, well-mannered co-workers, and full-empowerment to unleash one's creative energy to drive the solutions in whatever direction markets, customers, and end-users demand. The antithesis of lean thinking is traditional thinking embodied by micromanagement; over scoped product specifications divined by disconnected analysts working in a vacuum; 5-10-and-15-year integrated master schedules (IMSs); large, overallocated, and disempowered teams of automatons (robots); failure to deliver anything at all; per chance, delivering expensive, low-quality obsolete solutions that are not needed at all by markets, customers, and end-users; and, of course, living in fear of overbearing taskmasters cracking the bullwhip to ensure full-utilization of teams and individuals like coachman driving a team of horses to their death to reach an impossible destination. After more than 70 years, lean thinking is entering into its golden age, but old habits die-hard, and there are many proponents of over scoped business requirements, IMSs, heavyweight processes and documents, full utilization, and tyranny.

2. Lean • Lea·der·ship (*lēn • lē'dər-shĭp'*) Guide, steer, point, inspire, influence; To direct, prepare, and practice application of pull-based, just-in-time, and waste-free principles

   ✓ Progressive lean leadership.
   ✓ Trusts lean thinking outcomes.
   ✓ Influences buyers and suppliers.

   The second most important principle of lean-agile distributed projects and teams is lean leadership at all levels. What this means is lean thinking among leaders at all levels of an organization, enterprise, business, or even public sector agency. Lean thinking isn't just another task or work package on an integrated master schedule (IMS) to be performed by a mindless automaton or robot. In other words, lean thinking is NOT a hybrid traditional-lean system where over scoped business requirements, integrated master schedules (IMSs), and enterprise architectures are formed and then delegated to individuals to apply a few lean thinking practices, techniques, and tools to untangle an impossible morass. Lean thinking means untangling the morass first and then having small teams apply some additional lean thinking practices, techniques, and tools to quickly develop innovative solutions. Not because of over scoped business requirements, integrated master schedules (IMSs), and enterprise architectures, but because the goal is to quickly place a minimum viable product (MVP) in the hands of a market, customer, or end-users, collect their feedback, rinse-and-repeat many times, and optimize the value for both buyers and suppliers. Buyers are satisfied because they quickly get a high-quality solution that satisfies their needs at the right price, and suppliers are satisfied, because they optimize revenues, profits, and market share. Therefore, it's important to establish a strong foundation of lean thinking values, principles, practices, and tools at all levels of the enterprise. Lean thinking cannot be delegated to a robot or automaton, it must originate at the very top of the executive food chain, permeate all levels of middle management, and even exist at all horizontal levels of the delivery teams. Lean thinking isn't an individual consultant with a lean six sigma blackbelt in a veritable sea, army, or locust plague of 200,000 traditional thinking people. Organizational change and culture begin at the very top, lean thinking must originate from the executive layers, and it must flow down to all elements of the organization. Lean thinking is not an enterprise objective, slogan, or poster on a wall, it is a fundamental behavior or mindset that is exhibited by all people throughout the enterprise, not just a lean thinking delivery team. If that's so, then lean leaders identify value, map the value stream, create flow, establish pull, and continuously improve. It's simply not enough to ask a team to do sprint planning, hold daily standups, deliver and demonstrate value every two weeks, conduct retrospectives if they have time to impact policies to which they have no power or control, and then move on to the next task in a mile long integrated master schedule (all the while micromanagers and product owners pile on more user stories to their backlog every day to ensure full utilization). Lean leaders must be hired into powerful positions, lead the way to lean thinking, obliterate traditional thinking barriers, reduce utilization, promote experimentation, satisfy customers, and improve morale too.

3. Ag·ile • Frame·work (*ăj'əl • frām'wûrk*) Nimble, adaptable, flexible, malleable, versatile; To apply proven highly-cohesive lean-agile portfolio, program, and project frameworks with known properties

- ✓ Proven lean-agile framework or method.
- ✓ Embodies lean-agile values and principles.
- ✓ Cohesive system of ceremonies and practices.

The third most important principle of lean-agile distributed projects and teams is to apply an agile framework. There are many lean-agile frameworks, methodologies, practices, techniques, tools, and metrics in the global marketplace. There's a veritable smorgasbord of lean-agile ideas, approaches, and piece-part components from which to choose a la cart. However, there are only a few cohesive frameworks that have reached a level of maturity necessary to exhibit the proper lean thinking behaviors. That is, identify value, map the value stream, create flow, establish pull, and continuously improve. These include, but are not limited to Scrum, Scrumban, and even the Scaled Agile Framework (SAFe) for larger teams of teams, programs, solutions (systems of systems), portfolios, and enterprises. So, beyond establishing a solid, strong, and supportive organizational fabric of lean thinking and lean leadership, individual projects and teams must consistently apply proven cohesive agile frameworks. The teams and individuals must be trained, certified, and experienced in the chosen agile framework. However, it's also important for leaders, middle managers, overseers, interfacing teams, and individuals to be trained, certified, and experienced in the chosen agile framework. The teams and individuals must voluntarily desire to follow, comply with, and utilize the chosen agile framework. There's no sense in having a single individual be willing to apply a given agile framework, but everyone else is diametrically opposed to applying the agile framework. This simply will not work, it will be ineffective, and frustrating at best, which, unfortunately happens all of the time. Some agile frameworks, like Scrum and SAFe are NOT a la cart—One simply cannot pick and choose which ceremonies, practices, tools, and metrics to apply to meet individual tastes, beliefs, and lifestyles. If the teams select Scrum as their framework, then the teams should perform sprint planning, daily standups, sprint reviews, retrospectives, and even backlog refinement (if necessary), no ifs, ands, buts, or excuses. If the teams select SAFe as their framework, then the teams should perform program increment (PI) planning, agile release train (ART) syncs, system demos, inspect and adapt (I&A), and even PI readiness planning, no ifs, ands, buts, or excuses. This, of course, does not mean going overboard, like having six-hour daily standup meetings, forming a sprint backlog for 5-10-or-15-years, adding 15 new user stories per day to the sprint backlog because the product owner has an obsessive-compulsive disorder (OCD), which is, unfortunately, often the case, and pushing the team to full utilization with individual user stories and tasks because the product owner and scrummaster are ignorant of lean thinking principles like limiting work in process (WIP). Consistency, brevity, and efficiency are king when it comes to agile frameworks, especially when applying lean-agile thinking in a team of teams context—where teams and individuals will complain, cheat, and game the system—So, care must be applied.

4. Small • Teams (*smôl • tēmz*) Crew, squad, group, troop, collective; To form and apply small two or three person teams of people with complementary technical skills, abilities, and talents

- ✓ Form small, cohesive teams.
- ✓ Keep team size as small as possible.
- ✓ Compose teams of complementary abilities.

A critically important principle of lean-agile distributed projects and teams is to form very small agile delivery teams. While it's intuitive or instinctual to form teams of 5-10-or-15-people or more, the best team size is around two to three people with complementary skills, abilities, experience, and desires. There's an age-old maxim called "Miller's Limit" or the seven-plus-or-minus-two-rule, however, 7 to 9 people on a team is simply far too many (much less 15 or more). Only one person will end up doing all of the work, and the others will end up planning or managing the sole developer. They'll simply break out into a bar room brawl to see to can micromanage the sole developer. This even happens in smaller teams of two or three people, so let's apply lean thinking principles from the start and reduce the management overhead, waste, fat, and work-in-process (WIP) by ten times right out of the gate. Even with smaller teams of two or three people, when the organization is smart enough to keep teams this small, teams will instantly become unbalanced when one of the people assumes the role of overbearing micromanager while the sole developer works faster and faster to keep the micromanager from going postal. Micromanagement is a cancer and disease on agile teams many agile strategists and coaches don't understand the basic difference between servant leadership and micromanagement. There are ways to thwart team imbalance like 360° assessments, anonymous feedback, and team rotation. The notion of pair programming was created around 1990, made it on the short list of Extreme Programming (XP) best practices by the late 1990s, and quickly came under fire by computer scientists in the 2000s who were hell bent on statistically proving that teams were no more effective than individuals. This was a carry over from the 1990s, when computer scientists wrote many PhD dissertations to disprove the effectiveness of Michael Fagan-style software or code inspections. That is, they sought to prove teams were no more effective than individuals at spotting code defects, therefore the economics of software inspections were unjustified. Therefore, when pair programming became all-the-rage in the early 2000s, computer scientists quickly piled on the economics and effectiveness of pair programming in a vain attempt to illustrate that "two heads are not better than one!" Of course, there was a ton of cognitive bias illustrated by the computer scientists, many of their studies were gamed to prove their point, and, of course, they never said programmers were slower than individuals working alone, but they simply couldn't justify the expense of two people working on one user story at a time. In other words, they were steeped in the traditional mindset of full utilization (i.e., have two people work on parallel user stories and get two for the price of one). The reality of course, is that you'll get two half-assed user stories, if at all, rather than one fast, high-quality user story. None-the-less, pair programming is making a comeback!

5. Team·ing • Agree·ments (*tēm'ĭng • ə-grē'mənts*) Pact, treaty, charter, covenant, arrangement; To form, adhere to, and continuously improve a set of teamwork values, principles, and practices

- ✓ Quickly form team working agreements.
- ✓ Focus on lean-agile values and principles.
- ✓ Practical, measurable, and frequently visited.

Another important principle of lean-agile distributed projects and teams is to quickly form teaming agreements. That is, if you have the wisdom to form small agile teams to quickly deliver completed user stories to markets, customers, and end-users, then the team should have a set of basic rules in which to operate. This is sort of like a definition of ready (DoR) or entrance criteria. There is no set formula for what these rules should be, and the teams should be empowered to formulate their own operating rules. However, there should be some basic mandatory screening criteria like apply lean thinking; apply agile an framework; consistently apply agile ceremonies and practices; be fast, efficient, and lean; minimize waste and WIP; maximize openness, transparency, information sharing, honesty, emotional intelligence; no micromanagers; no lopsided teams with only one developer; eat your own dogfood, what's good for the goose is good for the gander, and no hypocrisy; and no OCD psychopaths, bullies, and stalkers who talk a mile a minute but do no work themselves. Teaming agreements should be reviewed during retrospectives, at the beginning of each sprint, and even used as an internal team assessment to determine who's been naughty and nice so that teams, projects, and organizations can institute emotional intelligence training if necessary, in addition to training on lean thinking, agile frameworks, ceremonies, practices, tools, and metrics. Some studies show that 70% or more of employees are extremely dissatisfied with their organizations, programs, projects, and team members. Much of this is due to micromanagers, bullying, and condescending behavior by managers, supervisors, and team leaders. Worse yet, stress heart attacks are more prevalent among worker bees at the bottom of the hierarchy than at the top. Workplace bullying, like micromanagement, is a cancer in modern organizations that must be rooted out, thwarted, and cured. Teaming agreements are only the first step in ending this global organizational pandemic. It's a two-way street, everyone should exhibit high levels of emotional intelligence. Worker bees, programmers, and developers often resort to sarcasm, criticism, and course joking toward their micromanaging supervisors, which only exacerbates the poisonous team atmosphere leading to low morale, team performance, and organizational ineffectiveness. Customers, markets, and end-users are the ones who pay the highest price if no one is listening to them, they receive late inferior products and services, or worse yet, they receive nothing at all from dysfunctional organizational cultures. Everyone on the team works, everyone must exhibit high levels of emotional intelligence, and a positive team atmosphere must be maintained above and beyond our basic human nature. Again, more importantly, lean-agile thinking values, principles, practices, tools, and metrics must be consistently applied; they must be trusted to get the job done; and the teaming agreement must demand continuous improvement.

6. Team•Ro·ta·tion (*tēm • rō-tā'shən*) Turn, change, revolve, circulate, alternate; To frequently change team composition in quarterly cycles to share skills, maintain dynamism, and enhance performance

- ✓ Keep teams together to deliver an MVP.
- ✓ Rotate individuals to maximize value delivery.
- ✓ Keep teams fresh and prevent them from getting stale.

An often-overlooked principle of lean-agile distributed projects and teams is frequent rotation. That is, individuals within teams should be rotated to new teams early and often. This has many organizational benefits, like greater socialization at the program, project, and team level; cross-pollination of skills; performance load-balancing and level-setting; and, of course, keeping individuals, teams, and projects fresh and dynamic. There's always Tuckman's teamwork model to contend with (i.e., forming, storming, norming, performing, and adjourning). That is, it takes a few days, weeks, months, and even years for people to get comfortable with one another, build long-lasting trust, and stabilize team performance. It's also very similar to Virginia Satir's model (i.e., late or old status quo, foreign element or change, resistance, chaos, transforming idea, integration, and new status quo). Or, even the five stages of grief (i.e., denial, anger, bargaining, depression, and acceptance) and five stages of anger (i.e., shock, anger, guilt, depression, and resolution). People realize that change is psychologically traumatic, painful, and undesirable, so they wish to minimize or eliminate change by keeping teams together for a long time (in order to minimize performance disruption). However, this is simply old-school traditional thinking. Yes, there may be some initial dysfunction when a new team is formed, but most teams will quickly recover, overcome hidden cognitive bias, learn something new, develop innovative solutions, and bolster organizational value in the long run. Besides, if a team assignment simply doesn't work out, there is always the next team rotation in which to look forward. Most agile frameworks use quarterly cycles as the basis for creating epics or minimum viable products (MVPs). This is probably the right cadence in which to rotate teams. Again, change is always disruptive, but the benefits of greater innovation certainly outweigh the costs. Micromanagers don't like team rotation, because they don't care who they micromanage, as long as they have a productive team of developers (horses). Of course, micromanagers are quick to rotate out or fire unproductive developers that don't make them look good. Team rotation is a good way to balance workload, performance, and shared responsibilities (i.e., turn everyone into a developer and minimize if not eliminate unproductive micromanagers). Sometimes, teams or individuals naturally self-organize into pairs, duos, or duets. That is, they find someone comfortable, compatible, and complementary, and stick with the favored duo for a very long time. It's okay if people find a soulmate, buddy, or lifelong friend, but this shouldn't prevent programs and projects from rotating individuals among teams. Team rotation is part and parcel to emotional intelligence, psychological resilience, and social and behavioral maturity. People will quickly learn what works and what doesn't, pain points and hot buttons, and develop the social anti-bodies for achieving enduring performance under continuous change.

7. Cross•Train·ing (*krôs • trā'nĭng*) Share, educate, tutor, fertilize, pollinate; To proactively share technical skills within and among technical teams to improve team synergy productivity, performance, and value

- ✓ Disseminate all technical knowledge.
- ✓ Ensure skill redundancy as much as possible.

✓ Openly share data, knowledge, skills, and information.

Another often-overlooked principle of lean-agile distributed projects and teams is cross training. That is, yes, form teams of people with complementary skills. Perhaps one person is a database expert, while another is a Java programming expert. Maybe one person knows how to use Microsoft Office, while another knows how to satiate the voracious appetites of hungry executives. Perhaps one has greater emotional intelligence, while the other one is highly motivated to think out of the box, get out of their comfort zone, and try something new. Maybe one person is good at documentation, while the other one has the gift of gab (telling a good story to clients and managers). Again, maybe one is a good technical developer, while another one has strong face validity with clients. This is part and parcel to forming multi-disciplinary cross-functional teams of individual subject matter experts (SMEs). However, there is a tendency for people to specialize, stay in their lane, and depend on people to play very specific roles. The person with strong face validity is expected to talk to executives and customers, the tool jock is expected to codify team plans, the analyst is expected to write user stories, the designer is expected to develop architecture diagrams, the developer is expected to write Java, the tester is expected to verify and validate the system, etc. Oftentimes, people will get paired with someone who has a similar skill set (i.e., team lead with team lead, planner with planner, analyst with analyst, designer with designer, developer with developer, tester with tester, etc.). The better arrangement is to pair people with unlike skills, so that each teaches one another enough about their discipline to be dangerous. It's said that an individual will be competent if they learn only 20% of a subject matter expert's (SME's) lifelong skills. So, the goal, is not to teach everyone on a team to be 100% competent in one another's skills, but certainly share up to 20% of one another's skills. That is, teach the team lead 20% of planning, analysis, design, coding, and testing skills, and likewise for the other disciplines. Therefore, everyone should have enough competence to perform one another's jobs. If worse comes to worse, and someone suddenly gets run over by a bus, then anyone on the team can perform enough of that person's role to get the job done (at least temporarily until a replacement can be sought with 100% of the needed skills). So, individuals within teams must be willing to share their technical skills with their teammates, they should be generous with sharing vs. selfishness, and individuals should be rotated to other teams in order to achieve further cross-pollination and dissemination of critical skills. Perhaps a few of the individuals will be further inspired to master the skills of one another's discipline, i.e., become a subject matter expert (SME) in more than one discipline. Much of this flies in the face of traditional managers who hire people to be one deep in each category of critical skills and then complain when they lose that person.

8. Lean•Meet·ings (*lēn • mē′tĭngz*) Summit, session, assembly, gathering, conference; To routinely, regularly, and voluntarily hold a few, sparse, short, essential and time-based meetings at prescribed intervals early in the day and week

✓ Hold bare minimum number of meetings necessary.
✓ Keep meetings short and strictly time boxed.
✓ Schedule as early in the day as possible.

An absolutely essential principle of lean-agile distributed projects and teams is lean meetings. That is, institute only the bare minimum number of fast, efficient, and critical lean-agile ceremonies and meetings. This is a bit of an oxymoron in most of the developing world that relies on all-day meetings, which often spill over into weekends. Sometimes, critical business meetings begin Friday at 5:00 pm in the Western hemisphere and carry over into all-day Saturday meetings. That is, Monday through Thursday are reserved for personal and developer time (i.e., sleeping late, going to the doctor, going shopping, taking the kids to school, cleaning the pool, etc.), so Fridays and Saturdays are reserved for serious group innovation time. Maybe the theory is that these days are not used for client meetings, so this is a good time for good old fashioned business development, research and development, and other innovation and training activities. Needless to say, late week all-day meetings are NOT lean meetings. A lean meeting is a 30-minute sprint planning ceremony, a 15-minute daily standup, a 30-minute product demonstration or review, a 30-minute retrospective, a 30-minute backlog refinement session, etc. 10-minute meetings are also very popular. If you're following lean thinking principles, limiting work in process (WIP), lowering utilization, focusing on a minimum viable product (MVP), and using one-piece workflow, then this is plenty of time to quickly develop a solution, get it in front of customers, collect feedback, and quickly replan for the next cycle (i.e., two-week iteration). However, if you have a mile-long integrated master schedule (IMS), business requirements document, or enterprise architecture, then, of course, this is not going to be enough time. More importantly, all of the basic meetings should be sparsely scheduled, held in the morning when people are the most attentive, not skipped, strictly timeboxed, and not repurposed or highjacked by hungry executives with insatiable appetites. Lean meetings are forcing functions for synchronization, cadence, teamwork, collaboration, communication, and innovation. To skip, abuse, repurpose, or ruin lean meetings is to ignore these fundamental lean-agile thinking values, goals, and objectives. Lean meetings should not be rescheduled willy-nilly because you got up late and have to walk the dog, you can't login to your computer fast enough in the morning, or you have something more important to do like brown nose with an executive on a regular basis to move up the food chain. Get your priorities straight, commit yourself to your team, honor your teaming agreement, eat your own dog food, and show some respect for your teammates who got out of bed on a rainy Monday morning to attend the 9:00 a.m. daily standup. If you've got some other priority that prevents you from honoring lean meetings first thing in the morning, then you're probably not a good candidate for THIS team. Don't abuse your team privilege for personal aggrandizement and expect a raise and promotion because you think you deserve it.

9. Meet·ing•Et·i·quette (*mē′tĭng • ĕt′ĭ-kĕt′*) Code, rules, manners, customs, conventions; To form, adhere to, and continuously improve a set of immutable lean-agile meeting values, principles, and practices

✓ Establish strict meeting rules and guidelines.
✓ Keep meetings open, collaborative, and egalitarian.
✓ Teach leaders and managers to obey meeting etiquette.

An egregiously ignored principle of lean-agile distributed projects and teams is meeting etiquette. This is a little bit like

teaming agreements, except that it is a basic set of immutable rules and laws for holding lean meetings. Many people have set out to write books on how to run productive meetings, including the more infamous “Robert’s Rules of Order!” First of all, schedule all meetings well in advance, preferably recurring meetings for iterative delivery, and don’t schedule or cancel a meeting five minutes before it is to occur (that’s simply moronic, inconsiderate, and selfish). Some people may plan their entire week or day around a single meeting, move personal appointments around, or hold off on important activities, only to have a narcissist cancel a meeting five minutes before it is to occur. Be on time to meetings, don’t come in five or ten minutes late or leave five or ten minutes early. For important planning meetings that may involve many people, schedule the meeting 10 to 15 minutes earlier so that participants may log in, verify their audio and video connections, and be ready to go when the meeting is scheduled to begin, not afterwards. For large group meetings like training events, materials are concisely orchestrated down to the minute, so its simply unacceptable to tax an important meeting by 15-30-or-45-minutes getting logged in. If its your first meeting of the day, don’t get out of bed and turn on your computer one minute before an important meeting, when you know its going to take your computer 10 to 15 minutes to warm up, login, stabilize, and connect to teleconferencing application. All meetings should have a clear agenda, each agenda item should be strictly timeboxed, and the entire meeting should be strictly timeboxed as well. It’s okay to end a meeting early, which should always be the goal, but its unacceptable to go longer than expected. It’s pretty routine for narcissists to continue talking for 10-15-or-30-minutes after a meeting has officially ended. Someone will say, “We have one minute left” and the narcissist will continue talking for an additional 30 minutes which is annoying. Hold all meetings, especially important decision-making meetings as early in the day as possible, preferably about 8:00 or 9:00 am in the morning. Never hold an important decision-making meeting after noon or at the end of the day or a Friday afternoon, because you’re done with your household chores and that’s when your personal schedule clears up. Holding important meetings at 4:00 pm sends a powerful message that the subject is not important in the grand scheme of things and neither are the participants. Space the meetings apart, don’t schedule back-to-back-to-back meetings, and don’t fill every 10-minute block of time with a meeting because you’re an extrovert who likes to think out loud or a traditionalist who wants to ensure full utilization. It’s okay to schedule working sessions, but even these should be kept to about one hour maximum, when productivity wanes. Stop scheduling late meetings as a weapon, which is quite common.

10. In·no·va·tion • Time (*ĭn′ə-vā′shən • tīm*) Buffer, margin, shield, cushion, guardrail; To plan, allocate, build-in, and adhere to extra time in an individual’s and team’s workday for creativity, invention, novelty, thought, uncertainty, and unpredictability

- ✓ Decrease task size for uncertainty.
- ✓ Proactively build in excess capacity.
- ✓ Allow time to be productive and creative.

One of the most critically important principles of lean-agile distributed projects and teams is innovation time. This represents the essence of lean thinking and lean leadership. Knowledge work is inherently unpredictable and uncertain. Specifications or requirements for innovatively new products and services exist as tacit, hidden, and inexpressible customer, market, and end-user needs. This is not something a business analyst, enterprise architect, or integrated master scheduler can divine, predict, or synthesize in a vacuum and program everyone’s time down to the minute and second for the next 5-10-or-15-years. There’s no sense in stuffing Scrum and SAFe backlogs full of epics, capabilities, features, user stories, and tasks every day in a vane attempt to boil the ocean. Instead, innovators, knowledge workers, and lean-agile thinkers listen to the voice of the customer (i.e., talk to real live people), formulate a few key business hypotheses, schedule a few sparse business experiments, and build a few small minimum viable products (MVPs) at a sustainable pace. When the MVPs are ready—representing simple models or facsimiles of market, customer, and end user needs—MVPs are shown to a few lead users; impressions, feedback, and data are gathered; and more, small, and sparsely spaced MVPs are created over and over again (with a very narrow scope). Once a true, tangible, or explicit market, customer, or end-user need is uncovered, revealed, or elucidated, then more detailed production-level products and services can be fashioned beyond a few simple MVPs. Even final product and service solutions should be kept as simple as possible, because technology simply evolves too fast and becomes obsolete; customer, market, and end-users are fragile, fickle, and volatile; and their needs will shift very quickly on a whim. Think of a new house, automobile, laptop, or smartphone. It doesn’t take very long, sometimes days and weeks before an even newer and innovative model appears on the market, so there’s no sense in building a gold fleeced, over scoped product or service over 5-10-or-15-years, when the appetite for an innovation is about 90 days. The more uncertain, volatile, risky, and innovative the problem domain is, the more innovation time is required to address it. In traditional thinking, the goal was always to fully allocate an individual’s time and sometimes over allocate it, just-in-case individuals are highly motivated and can achieve more than expected. Some lean-agile frameworks suggest an 80% allocation of a team’s and individual’s time (how utterly generous). Well, innovators now realize that in highly uncertain markets, the allocation of an individual’s time should be no more than 40% to 50%. This is especially true for network engineers operating brittle brick-and-mortar data center technologies from the last century. This is a hard pill to swallow for traditional thinkers who are dead set on pushing people to their breaking point every day with 120% allocation, but the payoff for allocating innovation time is tremendous.

11. Com·plete • Trans·par·en·cy (*kəm-plēt′ • trăns-pâr′ən-sē*) Open, clear, honest, visible, translucent; To share all data, information, and communications both upwards and downwards throughout a portfolio, program, and project team

- ✓ Openly share all project and team data.
- ✓ Ensure all data is available to leadership.
- ✓ Ensure leadership data is available to teams.

An essential principle of lean-agile distributed projects and teams is complete transparency. Traditional leaders have always insisted upon having full transparency into daily activities. That’s why status meetings, status reports, metrics, tools, project plans, schedules, and technical documentation are so important to them. Unfortunately, the cost of producing all of this data

usurps the ability to create innovative products and services. That is, 80% of an individual's effort is producing status reports and participating in status meetings to keep traditional leaders informed. This is especially true when there are mountains of business requirements, enterprise architectures, and integrated master schedules (IMSs). When there are literally thousands of elements in each of these traditional instruments, it makes it very difficult if not impossible for leaders to know what's going on at any given time. As Newton's Third Law states, "Every action has an equal and opposite reaction!" That is, we've bred a super race of left-brained traditional project managers who can sort through thousands of project elements each day to find a needle in a haystack! Not to worry, lean thinking to the rescue! When one downsizes batch sizes, project scopes, team sizes, minimum viable products (MVPs), etc., then there are no longer 15,000 elements in each business requirements specification, enterprise architecture, and IMS with total of 45,000 elements to consider. Instead, there may be 7 to 15 elements or less on a team's Scrum or Kanban board at any given point in time (assuming lean thinking, lean leadership, and WIP limits exist). Downsizing an individual team's scope from 4,500 elements to 15 is a huge difference, so we can now shift the skill for our leaders from extremely left-brained super computers to highly creative servant leaders with oodles of emotional intelligence, motivational spirit, and soft skills. A leader merely has to view a team's Kanban board in 5 minutes to comprehend its status, attend a 15-minute daily standup, or glance at a team's dashboard. It's sort of like being in a taxi and glancing at the dashboard to see how fast the automobile is traveling. However, complete transparency is much more than enabling lean leaders to instantly see and perceive a program's, project's, or team's status, but for leaders to share strategic organizational, portfolio, program, and project data with the teams themselves (and for teams to share their status with other teams too). Agile frameworks emphasize complete transparency so much, it is often the first-time traditional workers actually see another team's data. Even lean-agile team leaders insist upon knowing what their team members are doing, but rarely communicate leadership data down to the teams (such a shame). Complete transparency means 360° visibility into all program, project, and team data (not just upwards). Complete visibility helps everyone make strategic decisions, which agile teams are empowered to do, while lack of 360° visibility prevents agile teams from achieving their basic outcomes, purpose, and full potential.

12. It·er·a·tive • De·liv·er·y (*ĭt'ə-rā'tĭv • dĭ-lĭv'ə-rē*) Frequent, regular, repeated, continual, recurrent; To expect all teams to deliver working, valuable products and services at frequent regular intervals such as every two-weeks

- ✓ Apply iterative and increment ceremonies.
- ✓ Regularly deliver small valuable batches.
- ✓ Gather measurable feedback and replan.

Another important principle of lean-agile distributed projects and teams is iterative delivery. Agile teams should deliver something of value about every two weeks. This may be one or more small business experiments, a small minimum viable product (MVP), or a solution that satisfies a two-week iteration or sprint goal. All iteration or sprint goals should be written in the language of the business, market, customer, or end-user (i.e., "single sign-on prompt to consolidate disparate system credentials," "one-click checkout to speed product or service ordering process", "simple metasearch engine to consolidate the results of multiple disparate online used bookstores," etc.). While empathy, experience, or story maps may be used to explore the scope of a larger epic, MVP, or feature, iterations often focus on a subset of large epic or feature in the form of a small group of user stories that may require several iterations or sprints to complete. Regardless of whether an MVP is completed in one or several iterations, actual market, customer, or end-user feedback must be sought on the results of the iteration or sprint. Traditional managers assume all user stories in an iteration or sprint must be finalized using earned value management (EVM). However, this is not the case, and the outcome of an iteration or sprint is measured by the completion of its goal or objective. Perhaps, a small agile team estimates that it requires 15 user stories to achieve the goal but realizes experiments are realized in only 9 user stories. The purpose of lean thinking is to minimize waste, fat, or work-in-process (WIP) to rapidly deliver a high-quality, innovatively new product or service with the shortest possible lead and cycle time (vs. 100% of the planned scope as traditional managers errantly believe). Therefore, the extra 6 user stories are merely waste, fat, or WIP and the team succeeded in satisfying the iteration or sprint goal (while the traditional manager believes the agile team failed because they only deliver 60% of the user stories, whereas they actually eliminated 40% of the unneeded waste, fat, or WIP). Tacit, hidden, and inexpressible market, customer, and end-user needs are difficult to predict, so it's simply asinine to expect agile teams to estimate with EVM-like precision. None-the-less, small agile teams should aim to rapidly complete a small business experiment, epic, MVP, or iteration or sprint goal's worth of user stories about every two weeks. The goal is not full utilization, so its unnecessary for a traditional project manager turned product owner to cram a backlog full of user stories, allocate everyone's time down to the minute, and ensure everyone is earning their hourly pay individually. Again, with knowledge workers, the greater the uncertainty, the lower the necessary allocation of time and greater amount of teamwork involved in co-solutioning the business experiments. If the agile team keeps the experiment as small as possible, then they should have no problem delivering a small solution about every two weeks (vs. 5-10-or-15-year long analysis paralysis)!

13. Small • Batch·es (*smôl • bǎch'əs*) Set, lot, pack, bunch, group, collection; To expect all teams to deliver working, valuable products and services of a very small to moderate size at frequent regular intervals

- ✓ Reduce batches to smallest possible size.
- ✓ Ensure batches are minimum viable product.
- ✓ Use one-piece workflow to serialize production.

Another essential principle of lean-agile distributed projects and teams is small batches. That is, programs, projects, and teams should plan a series of small business experiments, epics, minimum viable products (MVPs), or features. Once again, true product and service specifications exist as tacit, hidden, and inexpressible market, customer, and end-user needs that are difficult to predict. So, it's completely unnecessary to hire a small army of business analysts, enterprise architects, and integrated master schedulers to create tens of thousands of requirements, design elements, and work packages spanning 5-

10-or-15-years. Systems and software engineers have known for decades that over 95% of a complex system's features are never used at all by markets, customers, or end-users, so it's ridiculous to build gold fleeced, over scoped, and bloated systems at a cost of decades and billions of dollars. This fails to mention that innovatively new technologies evolve every 90 days and selecting a technology stack for an innovatively new product or service is like hitting a moving target. Many global manufacturers simply cannot keep up with the pace of change. New product and service specifications are quickly obsolete before detailed blueprints can be developed and a manufacturer can be identified to produce a prototype for evaluation of complex new products and services with long lead times. Manufacturers themselves are subject to highly competitive market forces, so if they cannot keep up the pace of technological change (i.e., their facilities cannot produce competitive wares), they simply cannot exist long enough to produce a rapid prototype of an over scoped design. Therefore, lean-agile thinking to the rescue! That is, create innovative new products in services in extremely small batches that can be produced in hours, days, and weeks (vs. 5-10-or-15-years). Better yet, use agile technologies like virtual cloud experiments that can literally be produced in minutes. Again, the goal is not to create a new product or service and then find a gullible market, customer, or end user to buy it against their better judgement—That's traditional thinking at its finest. Instead, listen to the market, customer, or end user; rapidly formulate a small batch of business experiments; quickly design and implement them; and then measure their overall satisfaction with the result (in hours, days, and weeks). Using "design sprints," a team of innovators can formulate a product or service specification, rapidly prototype it, and present it to customers or end-users in only one week. There are a ton of lean-agile thinking principles here (i.e., extremely small batches, "teams" of innovators, rapid prototyping, and quickly gathering measurable feedback from live users). There's no time for the dust to settle, plan oodles of waste, fat, and WIP, or let technologies become obsolete. There's simply no room for 5-10-or-15-year business requirements, enterprise architectures, or integrated master schedules (IMSs) in the modern world of creating innovatively new products and services.

14. Lim·it·ed • WIP (*lĭm'ĭ-tĭd • wĭp*) Waste, excess, surplus, overflow, inventory; To expect all teams to obsessively minimize the unnecessary materials and work necessary to quickly deliver value-adding features

- ✓ Viciously limit work-in-process (WIP).
- ✓ Use single-tasking and simple processes.
- ✓ Dramatically reduce utilization for uncertainty.

Another critical principle of lean-agile distributed projects and teams is limited WIP. In manufacturing, work-in-process (WIP) is excess inventory used to produce large batches of goods and services. If you're making an automobile, then you many need raw materials such as steel, wiring, paint, plastic, rubber, glass, etc. You may also need components such as doors, bumpers, seats, bodies, windshields, tires, engines, suspensions, exhaust systems, etc. A 100 years ago, automobile manufacturers accumulated tons of raw materials and component parts to make a few automobiles, most of which was waste (never used at all). Raw materials and labor were cheap, and the world was chock full of natural resources for the taking. The equivalent of inventory for modern products and services includes mountains of business requirements, enterprise architectures, integrated master schedules (IMS), design diagrams, test plans and test cases, user documentation, life cycle documentation, and complex, slow, ineffective, and expensive multi-million-dollar tools. Product and service lifecycles from the 1970s demanded such excess inventory in the form of traditional linear waterfall models like MIL-STD-1521b, DoD-STD-2167a, MIL-STD-498, J-STD-016, ISO/IEC 15288, ISO/IEC 12207, CMMI, ISO 9001, Zachman Framework, DoDAF, or some insane combination of these. The more excess inventory the better according to these paradigms as though it was the 1920s all over again (because they directly sprang from Taylorism originating in 1900). With a dearth of talented product developers, programmers, and engineers dating to the dawn of the electronic computer, it simply became more cost effective to produce business requirements, enterprise architectures, and integrated master schedules (IMSs). The only problem with these models is they slowed product and service delivery to a crawl, often stopping it completely. Remember, full utilization of developers of innovative products and services beyond 50% actually stops productivity completely. The sad news is that proponents of traditional thinking truly believed they were improving productivity with excess inventory. Once again, lean-agile thinking to the rescue—Dump the business requirements, enterprise architectures, and integrated master schedules (IMS); focus on a few small business experiments; and build utterly simple minimum viable products (MVPs) to iteratively tease our tacit, hidden, and inexpressible market, customer, and end-user needs a little bit at time. There is no room, time, budget, or patience for excess inventory in lean-agile thinking; strip the process and product down to its bare minimum essential scope; and dispense with the over scoped, gold fleeced, and bloated product and service specifications requiring 5-10-or-15-year cycle times and billions of dollars to complete. Be careful now, because many product owners and scrummasters are former traditional project managers and will jam product backlogs full of user stories to ensure full 100% utilization, instantly killing productivity dead.

15. Keep • It • Sim·ple (*kēp • 'ət • sĭm'pəl*) Easy, plain, basic, elementary, uncomplicated; To expect all teams to keep construction and delivery of working products and services as small, simple, narrow, and easy as possible

- ✓ Keep backlogs as simple as possible.
- ✓ Focus on MINIMUM viable product (MVP).
- ✓ Keep batchsize and complexity extremely small.

A closely related principle of lean-agile distributed projects and teams is keep-it-simple. This may be a little more palatable for small agile teams to easily comprehend instead of complex manufacturing mathematical concepts like small batches and limited WIP. Keep-it-simple is rather intuitive for lean-agile thinkers but is extremely difficult for product managers and scrummasters to comprehend who were traditional project managers in a former life. Some people are naturally gifted, deeply analytical, and were born to boil the ocean with large batches and excess inventory in the form of abundant traditional, lean, and agile practices. Today, there are hundreds of traditional, lean, and agile methodologies, literally thousands of techniques, and traditionalists want to apply them all, simultaneously. The more the merrier, especially if "YOU" have to implement them.

What's good for the goose is certainly not good for the gander, traditional product managers want to pop into a daily standup for a few minutes and fill your product backlog full of new user stories like Santa filling your stocking on Christmas Eve! When they're not filling your backlog with user stories in daily standup meetings, they're meeting with customers, collecting even more action items, and will be happy to drop these into your backlogs before the next daily standup, or collect these in personal notebooks with which to fill your backlog at the earliest opportunity. Boil the ocean at every opportunity is the mantra of traditional thinking product owners and it's difficult to convince them otherwise. Naturally left-brained product owners have a sense of manifest destiny, which is to micromanage your product backlog to full utilization several times over every day. Hence the principle, keep-it-simple! Put one small business experiment in the backlog each week or two, ditch the power-distance hierarchy, collaborate "together" as a team, and use one-piece workflow principles to translate the epic, feature, or user story into a workable solution. But, wait, there's more, it's not done, its still WIP. It must be shown to a live market, customer, or end-user representative; measurable feedback must be collected; and it may need to go through a few more revisions before its ready for primetime. That is, it may take two or three two-week iterations to get it just right—That is, find the "Goldilocks Zone"—Not too hot, nor too cold! Again, there's no sense in filling the product backlog chock full of 70 user stories in a six-hour daily standup meeting, taking a fierce Western individualistic "divide-and-conquer" attitude, and assigning or pushing each of these onto an individual team member to work in parallel. Worse yet, expecting one or more fully solutioned user stories by individuals every two or three weeks. As Albert Einstein said, "Any intelligent fool can make things bigger and more complex, but it takes a touch of genius and a lot of courage to move in the opposite direction!" Unfortunately, there are more naturally left-brained traditionalists than lean thinkers in the field of new product and service development.

16. Fre·quent • Ret·ro·spec·tives (*frē'kwənt • rĕt'rə-spĕk'tĭvz*) Analysis, reflection, remembrance, recollection, investigation; To expect all teams to conduct frequent root cause analysis and process improvement activities at regular intervals

- ✓ Schedule regular retrospectives.
- ✓ Empower teams to make improvements.
- ✓ Build confidence necessary to make them work.

One of the most valuable principles of lean-agile distributed projects and teams is frequent retrospectives. While many programs, projects, and teams focus on customer feedback, iterative delivery, teamwork, or lean-agile frameworks, frequent retrospectives are truly the secret sauce. That is, reflecting on team performance in frequent, near-term intervals (preferably every two weeks). In traditional thinking, these were known as project post-mortems that were conducted every 5-10-or-15-years—Literally nothing was gained. However, if teams conduct a retrospective every iteration, sprint, or couple of weeks, then there is a lot to be gained. For one thing, only two weeks may be lost or sacrificed at best. That is, the team formulates a two-week plan, identifies a few user stories to satisfy a sprint or iteration goal, develops and delivers a solution to an end-user, gathers feedback, and reflects on whether end-users liked it or not. Not just reflecting, but translating possible improvements into constructive changes (i.e., smaller batches, better experience mapping, faster performance, innovative designs, malleable product or service mediums, better security, etc.). Although teams may brainstorm many possible improvements, the Pareto Principle applies, and 20% of the suggestions represent 80% of the possible gains. Traditional managers turn every process improvement into a new user story, business requirement, design element, or work package on an integrated master schedule (IMS). Not all of the process improvements will have equal weight, value, or impact; and doing so increases technical debt, cost, complexity, lead and cycle time, low morale, dissatisfied customers, and technological obsolescence. However, in today's marketplace where small, frequent business experiments reign, small teams conduct dozens if not hundreds of experiments "per day!" Therefore, it may be wise to conduct daily retrospectives, because waiting two weeks is like "dog years" or is the virtual equivalent of a 5-10-or-15-year IMS. Yes, the best method is to show a market, customer, or end-user the result of a small business experiment quickly and retrospect on the spot (i.e., what went well, what went badly, and what can go better the next time). Customers are not shy, and they'll be happy to redesign your innovatively new product or service for you on-the-spot! This is far better than "ignoring the voice-of-the-customer," going back to the office, and then asking new product and service developers their opinion of what went well, what went badly, and what can go better the next time! The essence of voice-of-the-customer is the voice-of-the-customer, so let them retrospect on your innovatively new product or service for you. Doing so increases product or service innovations by orders of magnitude in just a few minutes at little cost. Ironically, retrospectives are one of the most skipped, forgotten, or omitted ceremonies in lean-agiledom. People simply don't have time for them, do them badly, do not see the value of this powerful tool, have big egos, and are simply afraid of criticism.

17. A·non·y·mous • Feed·back (*ə-nŏn'ə-məs • fēd'băk'*) Comment, proposal, observation, suggestion, recommendation; To collect individually unattributable suggestions for process improvement during frequent retrospectives

- ✓ Collect anonymous feedback early and often.
- ✓ Make brainstorming non-attributable if possible.
- ✓ Maximize psychological safety through anonymity.

A powerful principle of lean-agile distributed projects and teams is the notion of anonymous feedback. Oftentimes, agile team members or individuals are a little bit shy about making suggestions for improvement. Worse yet, most project managers, team leaders, agile strategists, agile coaches, product owners, scrummasters, and other team members are simply immune to suggestions for improvement. An organizational structure, be it a rigid traditional hierarchy or self-organizing agile team, is an implied power-distance enforcement mechanism. That is, people at the top of the formal or informal hierarchy have all of the power and status, while people at the bottom have little to no power and status at all. It's not unusual for an agile team of 10 to 15 people to have 14 leaders and analysts managing a single software developer directly adding market, customer, or end-user value. In this scenario, only the 14 non-developers have all of the power and status, while the single developer has none. Therefore, retrospectives are often skipped by the leadership core who don't want any constructive change from powerless

developers, or simply ignore it. People at the bottom of the hierarchy become cynical about conducting retrospectives, say they are too busy “coding” for such a useless activity, and the leaders who don’t want any feedback anyway to cave-in and cancel all team retrospectives. Continuous improvement is one of the most powerful tools in the lean-agile arsenal, and it’s the secret sauce or “flux capacitor” that will enable your team to jump to light speed on a moment’s notice. Therefore, skipping retrospectives because leaders do not want input from lowly developers and cynical developers are tired of having their input ignored is like getting stuck in first gear, when your team could jump to light speed and become a high-performing team. One way to thwart this lack of team performance and unwillingness to conduct frequent retrospectives is to solicit anonymous feedback. This way, all comments are non-attributable and are not directly associated with implied power distance hierarchy. There is a plethora of online tools to enable teams to collect anonymous feedback. For instance, Poll Anywhere allows people to text in as many responses to a set of retrospective questions as possible in a few minutes or seconds. It’s fun, it gamifies the retrospective process, it engages people without attribution, and everyone can see dozens if not hundreds of retrospective responses appearing on the screen at one time, helping people to think of more ideas. Best yet, Poll Anywhere collates the results in real time, identifies patterns, and puts them in a fun easy-to-use graphical chart without a scrummaster having to fat finger in all of the responses manually. It makes doing retrospectives, fun, easy, powerful, and less apt to be skipped, unless your traditional or lean leadership simply does not want to supercharge their teams due to inflated egos or ignorance of retrospectives. This maximizes knowledge worker input and turns them high-performing teams if results are acted upon.

18. Sim·ple • Met·rics (*sĭm'pəl • mĕt'rĭks*) Unit, number, amount, measure, quantity; To collect and utilize only the bare-minimum metrics and measurements necessary to capture the essence of project and team performance

- ✓ Apply bare minimum lean-agile metrics.
- ✓ Gamify lean-agile metrics as much as possible.
- ✓ Use metrics to achieve optimal team performance.

A perplexing principle of lean-agile distributed projects and teams is simple metrics. Unfortunately, the notion of using metrics has become burdensome, laborious, and hazardous. Using metrics has always been highly suspect since Frederick Taylor measured blue collar work crews with a stopwatch in 1900, earned value management (EVM) was pioneered in the 1960s, and dozens of them emerged in the 1970s Precambrian explosion of metrics. Almost everyone was dissatisfied with metrics, from mathematicians who questioned their validity, project managers who were resistant to becoming mathematicians, and knowledge workers who refused to be measured like blue collar workers. Although volumes of textbooks sprang out of the 1970s and 1980s, few teams ever used any performance measures at all, save an integrated master scheduler or two using an IMS to divine a highly uncertain and unpredictable future. It was pretty typical for any metrics proponent to instantly be burned at the stake from all quarters. Humans simply do not want to be measured. Many industries, business functions, and other highly competitive teams gamified the use of metrics throughout the decades. For instance, soft drink makers gave free prizes for large numbers of bottle caps, airlines gave free tickets for airline miles, and salespeople used play money to see who could gather more sales in a given quarter. To gamify, meant to make it fun and some people seemed to respond to games better than individual performance measures like sales quotas, units produced, and other similar other productivity measures. Even agile proponents stepped up to gamify the performance of agile teams in terms of sprint goals achieved, relative story points, business value per user story, velocity, burndown, etc. The point is, no pun intended, is that it was meant to be a “game or fun,” hence the phrase, “business value game!” That is, assign relative business value to user stories, rank them by business value, size, and complexity, and determine the best mix of user stories necessary to satisfy markets, customers, and end-users. It was a little bit like playing Monopoly! Unfortunately, traditional managers turned product owners immediately stepped in to convert the qualitative “gamified” agile metrics into quantitative ones. Story points were equated to hours or days, every user story was assigned one point, and every individual on a team was assigned 8 user stories per day. Therefore, a small agile team of 5 developers could possibly produce 400 story points worth of product scope per two-week iteration. If a team reduced their velocity below this theoretical limit, then the product owners bullied the individuals and teams on a daily basis to reach the theoretical limit of 400 story points per iteration. So, what was initially a game, turned into white collar slavery. The point of this diatribe is to keep it simple, gamify the agile process with simple metrics, and supercharge your team’s productivity with plenty of extra time to account for unpredictability, uncertainty, and innovation (i.e., limit WIP)!

19. Vi·su·al • Dash·boards (*vĭzh'o͞o-əl • dăsh'bôrdz'*) Graphs, charts, instruments, control panel, visualizations; To configure a small set of aggregated lean-agile metric, model, and measurement dashboards to visualize overall performance

- ✓ Use performance management dashboards.
- ✓ Establish consistent dashboards across teams.
- ✓ Apply consistent measures as much as possible.

A closely related principle of lean-agile distributed projects and teams is visual dashboards. That is, if you’re gonna bite the bullet and use simple metrics, then the programs, projects, and teams should visualize the metrics data or results as much as possible. Humans are naturally right-brained creatures and, images are far easier to comprehend in an instant, because a picture is worth a thousand words. Therefore, it’s best to use visual dials, graphs, histograms, pie charts, radar charts, and other visualizations to communicate the performance of programs, projects, and teams so everyone can instantly grasp overall performance. Or, vice versa, identify anomalies, outliers, and other inexplicable spikes for further investigation. However, there’s a fine line between looking for variations, which humans are prone to do since we’re genetically wired for highly sensitive visual acuity (i.e., spotting differences), and overreacting to special causes (which is yet another one of our more unpleasant instincts). Humans instantly want to promote their favorite fair-haired, blue-eyed golden child if they see even a tiny spike in productivity. Conversely, humans will immediate punish the guilty dark swarthy looking foreigner with a funny accent, often with termination, for even the slightest downward trend in productivity. Therefore, W. Edwards Deming, the

infamous quality guru from the 1920s vehemently warned global managers AGAINST overreacting to special causes, which they kindly ignored for more than a century. Again, humans are genetically wired to violently discriminate against the slightest visual or auditable variation in data (i.e., it's a simple survival skill). In addition to simplifying and gamifying metrics, the visualizations should be as simple and fun as possible too. Most automobiles come with a dashboard that tells the driver how fast the automobile is traveling, how much fuel it has, the temperature of the engine, environment, and cabin, and a host of other critical indicators. An automobile would be completely useless without a visual dashboard. So, why would you want to drive an agile team without one (i.e., you simply wouldn't). So, let's stop driving our agile programs, projects, and teams blindly in the dark and give them some simple visual dashboards to help them stay on track (much like a GPS). Again, the biggest consumer of these visualizations is the agile team itself, much like the driver of the car. The purpose of visualizations is for the team to self-organize, regulate its own speed, performance, and morale, and continuously improve itself to the highest possible performance. Visual dashboards are not just for upper (micro) managers to use for instant punishment when a dial, graph, or needle points in the wrong direction because of special causes as we've done for 100 years. Again, the goal of visual dashboards is not to ensure that agile teams reach 100% utilization and saturation, but rather that they have enough extra capacity to properly manage uncertainty, unpredictability, and innovation (which is quite the turnaround from EVM).

20. Vir·tu·al • In·fra·struc·ture (*vûr'choo̅-əl • ĭn'frə-strŭk'chər*) Network, platform, applications, administration, communications; To apply a low-cost commercial distributed information technology platform for use by management and technical teams

- ✓ Acquire use of commercial IT services.
- ✓ Provide consistent IT infrastructure to teams.
- ✓ Select reliable, low-cost IT infrastructure services.

An immutable principle of lean-agile distributed projects and teams is a virtual infrastructure. That is, programs, projects, teams, and, of course, individuals should be proactively provided with a state-of-the-art virtual infrastructure. This may manifest itself as a virtual information technology (IT) infrastructure hosted on a commercial cloud service provider such as Amazon Web Services (AWS) or Microsoft Azure. The virtual infrastructure should provide access control services, i.e., single sign on identity management system (IDM), high performance network access, and a reliable and persistent platform for hosting administrative applications, productivity tools, application lifecycle management (ALM) tools, databases, content management systems, etc. Again, most of this can be instantly procured on a low-cost, per-user elastic basis from a cloud provider such as AWS and Azure. It's completely unnecessary for programs, projects, and teams to depend upon 20-year-old traditional brick-and-mortar data centers that are yet to be procured, are stuck in the stone age, or are subject to system outages and degraded performance multiple times per week (especially during peak workloads). Most of the workforce logs into the network for their first daily standup meeting at 9:00 am, and its simply inexcusable for the data center to crash when a few thousand agilists try to login at one time just as they're getting out of the shower. With virtual cloud services providers such as AWS or Azure, system availability, reliability, availability, performance, capacity, and fault tolerance are almost guaranteed. Furthermore, they are subject to elastic demand and can scale up and down with spikes in user demand (i.e., extra servers are automatically provisioned at peak access times and automatically released when these short-term spikes in demand diminish). The point is that the purpose of your virtual agile team is to provide innovatively new products and services to markets, customers, and end-users, and they deserve a state-of-the-art virtual cloud services infrastructure that scales with their needs (vs. a 20-year-old brick-and-mortar data center that crashes at peak operating times because humans are simply unable to predict the provisioning capabilities necessary to sustain fluctuating user demand). Again, with commercial cloud services such as AWS or Azure, elastic load balancers allow new virtual data center components to be automatically provisioned for access control, more processing power, access to administrative and technical applications, relational and big data storage, backups and long-term storage, and an infinite variety of needs that finite humans cannot even think, dream, or imagine in a traditional data center. Of course, it's a luxury in the Western hemisphere to have broadband Internet service at access point locations that other countries may not have. Furthermore, developers may reside in a country without reliable broadband service, commercial cloud services, or power grids. So, great care should be applied to these teams.

21. End·point • De·vic·es (*ĕnd'point' • dĭ-vīs'əs*) Tablet, laptop, smartphone, workstation, personal computer; To provide new, state-of-the-art devices to teams and individuals to access the client's virtual infrastructure and collaborate together

- ✓ Provide endpoint devices to all teams.
- ✓ Provide modern laptops and smartphones.
- ✓ Use commercial VPNs and authentication services.

Yet another immutable principle of lean-agile distributed projects and teams is endpoint devices. All team members should be provided with state-of-the-art endpoint devices such as laptops, smartphones, and tablets for basic access to the virtual infrastructure. This may also include docking stations with full keyboards, risers and desk stands, and multiple monitors and monitor desk mounts and desk organizers too. This array of information technologies should also include wireless blue tooth headsets, earbuds, video cameras, speakers, ethernet cables, etc. The point is, pun intended, is that fundamental purpose of agile teams is to quickly produce innovatively new products and services, which they're going to use for 8 to 16 hours a day, up to seven days a week, so they should have high-performance state-of-the-art endpoint devices at the client's or host corporation's expense. Long gone are the days when only mainframe high-priests accessed computer systems in high-security data centers, employees had to fight for a terminal in a computer lab with limited resources, or only executives, managers, and other critical administrators were afforded the luxury of state-of-the-art endpoint devices. In today's knowledge economy, the most productive worker is the lowest-level member of an agile team, they are closest to the customer, and they have the technical skills to design, implement, and test the solution to an epic, feature, user story, business experiments, or minimum viable product (MVP). These devices should be available on DAY-ONE, they should be shipped using premium

delivery services to team members in advance of their first day so they can hit the ground running, and they should be easily configurable to access the network and be ready to go with little effort. Special information technology (IT) support teams and help desks should be instantly available around-the-clock to assist the members of agile teams in getting started, configuring their endpoint devices, trouble shooting them, and resolving all discrepancies immediately (including drop shipping new endpoint devices if the first set are inoperable). There should be no wasted time, and endpoint devices should be provisioned and replaced within a sprint or iteration without affecting velocity or sacrificing a sprint goal or business objective. Ergonomics should also be taken into account, so that agile team members are not hunched over a small laptop for 8 to 16 hours a day resulting in an occupational injury (i.e., Occupational Safety and Health Administration or OSHA violation). Many people do not have the luxury of a dedicated home office nor a desk in which to work, so an assessment should be made of home office accommodations and provided to the agile team members at little to no cost, or even temporarily leased (especially if they are long-term agile team members). Multiple large monitors and keyboards should be provided for optimum productivity, because it's next to impossible to do anything but read a few emails on an infinitesimally small laptop screen or keyboard.

22. Mes·sag·ing • Plat·form (*měs'ĭj'ĭng • plăt'fôrm'*) Note, memo, letter, communiqué, intimation; To provide low-cost commercial distributed text, chat, and instant messaging services for communication and collaboration

- ✓ Acquire low-cost commercial messaging services.
- ✓ Use internal tools already built-in to access points.
- ✓ Integration with IT platform, video, and voice a plus.

An important principle of lean-agile distributed projects and teams is a messaging platform. This is a given with smartphones, but people have to configure it properly, know how to use it, and use it well. Microsoft Office comes with Skype for instant messaging and teleconferencing. On some jobs, especially in the innovation sector where social skills may be a bit lacking, the first application configured and mandated is a messaging platform like Skype. Although Skype does archive or log conversations as a Microsoft Outlook email file, there are other persistent messaging platforms like Slack, Google Hangouts, Microsoft Teams, Matter Most, etc. Many agile teams prefer Slack to instant messaging services like Skype. The benefit of Skype is that it is bundled with Microsoft Office, proprietary communications are within an organization's Intranet, it has access to the organization's active directory service, and it is great for quick, private unscheduled communications. That is also a downside, because people will Skype you while you're concentrating, working on a tight deadline, participating in a meeting, or interrupt the team with an unplanned epic, capability, feature, user story, or task that somehow escapes sprint or iteration planning. There's an old adage in Scrum, that a sprint should not be interrupted once its begun, but a hungry executive (shark) will always swoop in with an instant Skype message to personally task you with an out-of-scope epic, capability, feature, user story, or task 24 hours a day. Skype is an executive shark's best friend (so much for Sprint planning)! Slack offers a more persistent messaging platform, where "watchers" can have their smartphones, desktops, laptops, tablets, and other endpoint devices "buzz" them when a new message has been posted to the chatroom. There can be multiple channels or chatrooms for group messaging, specialized chatrooms for subject matter experts (SMEs), or even private channels for direct person-to-person asynchronous communications. Again, these more persistent messaging platforms like Slack are a busy multi-tasking executive's best friend, because they can be working on many different projects or tasks at once, but instantly notified if there is something requiring their attention (which they seem to love). While interruption is not an agile team's nor developer's best friend, interruption drives an executive's daily schedule. More importantly, messaging platforms form a collective hive or memory, where the thoughts of every developer can be instantly captured, and subject matter experts (SMEs) can be instantly notified to learn something new, pick everyone's brain, or launch a personal criticism in five minutes or less to justify charging eight hours of salary for 30 seconds of criticism. Again, messaging platforms are a multitasking extrovert's best friend and help bring introverts into the conversations too (occasionally). Messaging services like Slack are used for upwardly mobile executives to instantly pick worker bee brains and are not an introvert's best friend.

23. Tel·e·con·fer·ence • Ser·vic·es (*tĕl'ĭ-kŏn'fər-əns • sûr'vĭs'əs*) Video, telephone, discussion, meeting, conference; To provide low-cost commercial distributed video conferencing services for communication and collaboration

- ✓ Acquire low-cost commercial meeting services.
- ✓ Ensure they're tightly integrated to access points.
- ✓ Ability to support largescale collaboration is a plus.

An essential principle of lean-agile distributed projects and teams is teleconference services. Examples include Skype, Zoom, WebEx, Microsoft Teams, Google Meet, Goto Meeting, Google Hangouts, etc. that can be used for team program, project, and team meetings. These services can be used for webinars, training, vendor and partner meetings, and meetings with other firms, public sector agencies, etc. Microsoft Windows comes with a built-in, tightly integrated teleconferencing service called Skype, which is quite nice. Skype also doubles as an instant messaging service as well. This is great if everyone is on the same network or has access to Skype web services beyond the local intranet. However, when doing highly collaborative training events, large group training events, or hosting or attending public webinars to share critical business intelligence, then other services may become necessary. This latter group may include Zoom, WebEx, Microsoft Teams, Google Meet, Goto Meeting, Google Hangouts, etc. Skype and Zoom are certainly the top two teleconference services, but WebEx continues to have a large following with Microsoft Teams rapidly gaining in popularity. The bottom line is that no one teleconference service is sufficient, programs and projects should make multiple teleconferencing services available to agile teams, and, of course, the agile teams themselves need the skills to interact with multiple tools. Of course, many corporations and their corporate networks were not designed for heavy workloads, were very traditional in nature, and were acclimated to traditional face-to-face meetings with good-old-fashioned whiteboards. However, in today's fully remote, networked world, this has placed big constraints obsolete corporate networks that were not designed to handle heavy workloads. In some cases, they'll

use video, but no audio to reduce bandwidth constraints. In other cases, they'll use audio, but no video. And, in many cases, they simply cannot handle a single event with hundreds of interactive participants very well, video and audio become choppy, and network latency, reliability, and availability become a major issue. Security also becomes an issue, as some corporations are very sensitive about their intellectual property and do not permit video nor audio transmissions even over highly secure encrypted lines. They may allow a generic training course or business intelligence webinar to come into the corporate network, but any proprietary transmissions emanating from corporate networks are often strictly prohibited. This presents a multidimensional challenge for early 21st century firms. One, corporate networks must be designed for heavy audio, video, and remote workforce workloads (which most are not). Second, information security must be managed carefully to prevent loss of intellectual property, proprietary data, or even export controlled information. Third, corporations must procure tightly encrypted virtual private network (VPN) services and trust them to maintain information security for 100% virtual workforces.

24. Vir·tu·al • Min·utes (*vûr'choo̅-əl • mĭn'ĭts*) Log, note, record, account, transaction; To provide low-cost commercial distributed easy-to-use application to capture instant meeting minutes for communication and collaboration

- ✓ Acquire low-cost commercial meeting minute tools.
- ✓ Best if integrated with commercial office services.
- ✓ Ability for all teams to access minutes a plus.

A subtle principle of lean-agile distributed projects and teams is virtual minutes. That is, a virtual note or meeting minute recording service for chronicling the events of key meetings, conversations, and other interactions. Microsoft Office comes with OneNote for this purpose, which can not only be used for recording individual or private notes but linked together in a common repository so everyone can instantly see all notes recorded on a program, project, or team. They're relatively flexible and simple-to-use but do require some skill and motivation to employ. For one thing, recording meeting minutes is more of an art than a science. Some people are good at capturing the essence of a meeting, others record every sentence like a court reporter, and yet others scratch out a few unintelligible nouns rendering the notes useless. Few people, save a left-brained traditional project manager will ever go back and read the meeting minutes, and only to serve as more fodder for crushing Scrum backlogs to death. That is, a team may have half a dozen meetings per day, five days per week. So, just to be sure no user story is left behind, everyone is fully utilized, or to help guarantee a promotion, a traditional project manager turned product owner will convert every bullet from meeting minutes into new user stories as often as possible. It's a bit unnerving if you're a lean thinker working on a single business experiment or sprint goal and come in on Monday morning to find 100 new user stories in your backlog midway through a sprint. Furthermore, few people can walk and chew gum at the same time (i.e., if your brain is fully engaged in a meeting, it's a bit difficult to take meeting minutes at the same time). The brain simply isn't wired to think and record at the same time. If you're on a teleconferencing service, hearing someone typing is a bit disruptive, people assume you are being rude and not paying attention, or are simply multitasking because the conversation is too unimportant. Skype is a bit annoying, because anyone taking notes on their computer will show up in the conversation box as typing a message. That is, Skype assumes every computer keyboard interaction is an instant message. So, speakers get a bit distracted and annoyed if someone appears to be typing an instant message while the main speaker is talking. Executives are meeting hounds, so they'll instantly assign a high-paid knowledge worker to disengage their brain and take meeting minutes for them. A better alternative to a manual virtual minutes system is an automatic note taking app like Otter, Sonix, Descript, Trint, Maestra, SpeedScriber, Ebby, isLucid, etc. There are so many to choose from, and perhaps Microsoft will bundle one in with Microsoft Office to speed their adoption to free up expensive knowledge workers while capturing high-fidelity meeting minutes. Fidelity is such an important aspect of meeting minutes, because upwardly mobile executives will simply ignore or mistranscribe the utterances of their direct competitors (i.e., cognitively blind to anything but the sound of their own voice).

25. Pro·duc·tiv·i·ty • Tools (*prō'dŭk-tĭv'ĭ-tē • to̅olz*) Email, browser, word processor, spreadsheet, presentation; To provide new, state-of-the-art office application suites on access points for communication and collaboration purposes

- ✓ Acquire low-cost commercial office suites.
- ✓ Provide commercial office suites to all teams.
- ✓ Include visualization and graphical processing tools.

One of the most basic principles of lean-agile distributed projects and teams is productivity tools. This is generally in the form of an office productivity suite such Microsoft Office. This can be a web-based version or a thick-client version on laptop, tablet, desktop computer. A thick-client version is best for optimal team productivity, performance, and outcomes. This suite should have highly cohesive ecosystems of email, messaging, video conferencing, word processor, spreadsheet, presentation, and note taking apps. These should be the latest versions and they should be updated and patched with regularity by the original vendor. This way, the agile teams have access to all of the latest features, updates, security patches, and quality and performance enhancements. This sounds so basic in the Western hemisphere, but this is not always a given, and basic state-of-the-art productivity tools are not always available to agile teams. A little-known requirement may also be basic training. While less than 95% of the features and functions in modern productivity suites are never used by the average end-user, there are a still a few agile team members that may not even know how to use the essential 5% of features like formatting text, changing fonts, creating tables, creating a basic presentation, or even creating a more sophisticated spreadsheet. Don't take these basic functions for granted and provide some training to your teams if necessary. Many agile team members are product owners or scrummasters who may have been traditional project managers in a former life, who delegated hands on data entry into productivity tools to their subordinates every day. An agile team has a flat, egalitarian structure, everyone works, everyone earns their keep, and this may be the first time an individual team member actually has their hands on a keyboard. Sometimes, this is because client and host corporations simply did not provide all employees with state-of-the-art endpoint devices. Again, many agile team members do not reside in the Western hemisphere where state-of-the-art

productivity tools are a basic luxury, so it's important to ensure all international team members not only have state-of-the-art endpoint devices, but state-of-the-art productivity tools as well. Likewise, basic training notwithstanding. Some teams may actually struggle with the finer art of productivity tool craftsmanship and struggle to create basic documents, spreadsheets, presentations, etc. It's important to provide templates for all team members of programs and projects to share, common repositories of sophisticated examples that lesser capable individuals can reuse and repurpose, and even have more advanced power users provide regular lightning talks, lean coffees, brown bags, and webinars on the finer art of creating sophisticated documents, spreadsheets, and presentations. In some cases, there are dedicated teams for rapidly branding materials in a professional manner to ease the burden on new team members and ensure a consistent look-and-feel.

26. Ag·ile • Tools (*ăj'əl • toolz*) Planning, tracking, workflow, reporting, project management applications; To provide low-cost commercial distributed lean and agile application lifecycle management tools and systems

- ✓ Acquire low-cost commercial agile tools.
- ✓ Select visually intensive, easy-to-use tools.
- ✓ Keep plans as simple and visual as possible.

Another essential principle of lean-agile distributed projects and teams is agile tools. These are agile application lifecycle management (ALM) tools such as Rally, VersionOne, Azure DevOps (ADO), Jira, IBM's Engineering Lifecycle Management (ELM) or Jazz, etc. This may also include basic Kanban tools such as Trello, Kanbanize, Monday.com, etc. It's important to align the tool with the type of agile framework the agile programs, projects, and teams are applying. For instance, use a tool optimized for Scrum if that's what the teams are using, select a tool optimized for the Scaled Agile Framework (SAFe) if that's what the teams are using, or apply a tool optimized for Kanban if that's what the teams need to apply, etc. Most agile ALM tools like Rally, VersionOne, ADO, Jira, or ELM/Jazz support Scrum, Kanban, and SAFe. However, that's where the similarities end, because some are more visual and simpler to use than others. ADO, Jira, and ELM/Jazz are like completely-knocked-down-kits or Lego building blocks (i.e., they must be assembled one widget, gadget, or gliffy at a time, and require heavy-duty scripting, programming, and configuration skills). So, be prepared to hire a full-time tool administrator, ask your developers to double as tool jocks, or manage long analytically intensive lists of bottomless Scrum backlogs without dashboards, visualizations, or other sophisticated features. Not many tool jocks know how to use the advanced features of these tools and there's a lot of vaporware and smoke-and-mirrors in advertising. Per chance the tool jock does know how to assemble more sophisticated configurations, they probably may not share their hard-earned knowledge with the rest of the agile team. Developers are particularly adept at withholding technical skills from the rest of the teams, and agile ALM tool administrators are no different. In some cases, enterprise agile ALM tool members may actually lock entire agile teams out of the tools, refuse access control, or even block individual users to maintain their exclusive base of enterprise power (i.e., job security). What's the point of this diatribe? Keep-it-simple, beware of the advanced skills required of completely-knocked-down-kits like ADO, Jira, and ELM/Jazz, and lean towards more sophisticated highly cohesive tool suites like Rally, VersionOne, or even emerging tools. There are literally dozens of agile ALM tools, so the sky is the limit. Again, although all tools will support bottomless list-driven Scrum backlogs, insist that agile teams use visual Scrum or Kanban boards to limit their WIP, visualize their workflow, and ensure instant maximum transparency to an agile team's current status. It doesn't help if a teammate, coach, or manager has to stare at a bottomless backlog with several hundred or thousand user stories, or even a Scrum or Kanban board with far too much WIP which left-brained analysts are prone to do. Again, the goal is to manifest lean thinking, visualization, transparency, and innovation, not physics equations, complex database queries, nor job security.

27. Vis·u·al • In·ten·sive·ness (*ˈvĭzh'oo-ə-l • ĭn-tĕn'sĭv'nĭs*) Graphical, pictorial, illustrative, pictographic, photographic; To provide low-cost commercial distributed visualization vs. list-driven lean and agile application lifecycle management tools and systems

- ✓ Focus on visual information radiators.
- ✓ Liberally use Scrum and Kanban boards.
- ✓ Create light, low-WIP and fast moving plans.

A closely related principle of lean-agile distributed projects and teams is visual intensiveness. This is part-and-parcel to lean thinking, transparency, keep-it-simple, and visual dashboards, but from a unique perspective. For one thing, a picture is worth a thousand words, it's much easier to convey information with illustrations, and communication and transparency are enabled through visual intensiveness (pictures). Again, many product managers, scrummasters, and even agile team members are former traditional project managers, so the default mode of capturing and managing work is to begin forming bottomless list-driven Scrum backlogs. In most commercial application suites, 95% of their features are never used at all, and 95% of user stories in bottomless Scrum backlogs are never used at all either. Agile application lifecycle management (ALM) tools are more than happy to use their elastic databases to stretch unto infinity with a daily helping of more and more user stories, action items, defects, honey-dos, nice-to-haves, and any other serendipitous idea. Traditional managers are left-brained analytical thinkers, wish to substitute Scrum backlogs for integrated master schedules (IMSs), talk to a dozen executives each day to brownnose and move up the food chain, and create a few dozen more user stories per day just to look good. That is, commit developers to as many user stories as possible to help them get promoted fast and then shame developers into working long hours to get them done. They often take a "divide-and-conquer" approach, mindset, or "assign" user stories to individuals, and hope to get promoted even faster by parallelizing all work and cracking the bullwhip! Of course, there is no lean thinking, transparency, or WIP limits in the mind of a "boil-the-ocean," "full-utilization-oriented" traditional project manager. And even developers themselves like the anonymity hiding behind the smoke-and-mirrors of bottomless Scrum backlogs because they know that only 5% of the backlog has any meaning at all, and it was probably done months ago. Instead, manifest your backlogs as simple visual Scrum or Kanban boards, bring your user stories out of the closet and into the light, and let everyone see what's hasn't been started, what's in progress, and what's been completed. Of course,

traditional project managers, left brained analysts, and developers will still obfuscate Scrum and Kanban boards with too much detail and decompose user stories into a few dozen or even hundred tasks so that a physicist cannot understand them. Yes, it is possible to put so much detail on a one-page Kanban board so than even an executive has no idea what's going on for months. It's even better if a Kanban exists as a graphical image with short abstract titles and no details to hide the fact that nothing is being done at all (for months). So, the trick is to get out of the bottomless backlogs, get onto visual Scrum or Kanban boards, keep-it-simple, limit the WIP, and pull a few user stories through quickly (shorten lead and cycle times).

28. Con·tent • Man·age·ment (*kŏn'tĕnt' • măn'ĭj-mənt*) Report, record, document, material, artifact; To provide low-cost commercial distributed lean and agile wiki tool and system for communication and collaboration

- ✓ Acquire low-cost commercial wiki services.
- ✓ Use wikis as lean-agile documentation tools.
- ✓ Link to application lifecycle management tools.

A critically important principle of lean-agile distributed projects and teams is content management. This is basically a wiki service tightly integrated with your agile application lifecycle (ALM) tool. While there are dozens of basic wiki services available for agile programs, projects, and teams, Confluence is certainly one of the most popular examples. It's basically an online website for creating and openly sharing textual, graphical, and numerical data. Think about Wikipedia for example, which is one of the largest collections of open knowledge and information in world history; gets 20 billion hits or visits per day, and replaced traditional encyclopedias and even some traditional libraries. For agile teams, a wiki web service like Confluence serves the same purpose, a central representation of all knowledge related to a group of programs, projects, and teams. In other words, it replaces most traditional forms of project, process, and technical documentation, reports, and other transient information. In some cases, it becomes a central forum for semi-structured collaboration, planning, reporting, and persistent information dissemination. An individual or team can simply place their latest planning, execution, and delivery data and results on a wiki, ask other teams to visit their wiki, or other teams and managers can visit wiki pages to gain insights into a particular team's state and status. Think of a wiki like Confluence as one giant virtual notebook. In some instances, Confluence is tightly integrated with agile ALM tools like Rally, VersionOne, Azure DevOps (ADO), Jira, IBM's Engineering Lifecycle Management (ELM) or Jazz, etc. So, backlogs and reports can be directly posted to the wiki through a system of application programming interfaces (APIs), so that agile team members can see the latest project status directly in the wiki instead of having to navigate through complex non-intuitive tools and views. Many tools are individually programmable and configurable, so a given team may create custom dashboards and tools that give them the best insight into the team's status at any given time. However, these custom visualizations, reports, dashboards, and tools are often only available to the teams themselves. So, if managers want visibility into a team's status, they must have the technical skills to navigate and data mine complex agile ALM tools themselves (or ask teams to walk them through their backlogs, Scrum boards, or Kanban boards via Skype or Zoom indirectly). However, these presentations are only transient vs. persistent and audiences may not have an opportunity to drill down into the details. In some cases, they may not have licenses, access privileges, or skills to do so. However, with a wiki service like Confluence, agile teams can link their reports and backlogs directly to the wiki and teams can casually view recent data with great ease nor the necessity for advanced agile ALM tool skills, query abilities, or access privileges. This isn't to say there is no privacy in with Confluence, and teams still have the ability to control individual access.

29. Com·mon • Re·pos·i·to·ries (*kŏm'ən • rĭ-pŏz'ĭ-tôr'ēz*) Record, archive, storehouse, depository, warehouse; To provide low-cost commercial distributed persistent electronic file storage space for communication and collaboration

- ✓ Acquire low-cost commercial repository services.
- ✓ Best if used to share large files across teams.
- ✓ Ability to support persistent storage a plus.

A related principle of lean-agile distributed projects and teams is common repositories. Agile application lifecycle management (ALM) tools like Rally, VersionOne, Azure DevOps (ADO), Jira, IBM's Engineering Lifecycle Management (ELM) or Jazz, etc. serve as basic content management systems for programs, projects, and teams. Even wiki services such a Confluence, Mediawiki, Tikiwiki, Dokuwiki, Slab, Slite, etc. also serve as basic content management systems as well. However, SharePoint, Google Workspace, Facebook Workplace, etc. may be better alternatives for collecting and distributing large number of big files to programs, projects, and teams. These are good places to store persistent data like corporate policies, standards, guidelines, business documents, training, and other highly cohesive and esoteric collections of closely interrelated files like those emanating from centers of excellence (CoEs), communities of practice (CoPs), portfolio, program, and management offices (PMOs), governance boards, and other special interest groups. These are alternatives to file sharing services that are native to large enterprise operating systems like Microsoft Windows Server. Whereas Windows Server was an active directory structure for not-so-public private files, common repositories like SharePoint are meant for not-so-private public files for larger enterprise-wide dissemination. Like Windows server, most common repository services like SharePoint come with access control services to limit the distribution and dissemination of data. Unfortunately, locking valuable corporate intellectual property behind closely held access controls is a bit antithetical to lean-agile thinking values, principles, and practices. In traditional thinking, information is distributed through carefully controlled stage gates (i.e., white belt, yellow belt, green belt, brown belt, black belt, master black belt, etc.). That is, corporate employees have to earn their way up the access control food chain. Even then, training and certification courses may only be available to a privileged few insiders, cliques, and corporate gangs serving the ego of a narcissistic junior executive. In lean-agile thinking, enterprises dispense with the cloak-n-dagger stage gates and freely disseminate all policies, guidelines, training, tools, and user manuals through wiki-like self-service dojos for the taking. Locking corporate secrets in tightly access controlled SharePoint sites is the quickest way to thwarting expensive, decade long multi-billion dollar change initiatives, while releasing corporate intellectual property in self-

service dojos is the quickest way to institutionalize change. Individual team members who embrace lean-agile thinking values, principles, and practices of openness, transparency, collaboration, and information sharing will be the first ones to post large, expensive, and highly innovative content to open SharePoint sites. Whereas dark narcissists will not be caught dead posting team intellectual property to SharePoint sites. Fierce Western individualism vs. lean-agile thinking—You make the call!

30. Vir·tu·al • White·boards (*vûr'choo̅-əl • wīt,'bôrdz'*) Flipchart, blackboard, chalkboard, posterboard, writing space; To provide low-cost commercial distributed easy-to-use brainstorming collaborative space for freeform communication and collaboration

- ✓ Acquire low-cost commercial virtual whiteboards.
- ✓ Best if simple, visual, intuitive, and freeform input.
- ✓ Ability to support anonymous brainstorming a plus.

An overlooked principle of lean-agile distributed projects and teams is virtual whiteboards. It's exactly like it sounds (i.e., a freeform virtual whiteboard with some bells and whistles of course). It's a space where people on a program, project, or team can (anonymously) gather to brainstorm simultaneously. That's a pretty loaded statement all by itself. With a single conference room, it's difficult to get anything greater than one-team in front of a whiteboard. Second, of all, few people are allowed to draw on the whiteboards—usually only a C-level executive—so the rest of the attendees are relegated to non-interactive participants. Perhaps an executive will solicit the opinions of the meeting participants but ignore statements of competitors to personally secure all of the commissions for a new product or service launch. In the event that a facilitator is allowed to gather inputs from all of the participants, the facilitator is a limited resource (bottleneck) and simply can't gather critical feedback from all of the participants. Even if the facilitator uses round robin techniques, a competitor may instantly shoot down or criticize an idea forcing the facilitator to disregard or retract a competing remark. There are other techniques to overcome competition, like having participants place post-it notes on a whiteboard and use dot voting to rank the ideas. However, they'll often ask whose idea is on a post it note and relegate it to the trashcan if its from a competitor or low-ranking team member. Agile facilitators are excellent at gathering ideas and then instantly down selecting them to only one, which usually comes from themselves. Traditional project managers are good at stabilizing a dynamic system and then skipping retrospectives and other continuous improvement activities for fear of destabilizing the system once again. In other words, executives are experts at filtering out creative ideas that will destabilize a low-performing system (which is actually the goal). No worry, virtual whiteboards to the rescue, whereas large conference rooms are hard to find. With virtual whiteboard apps, hundreds of participants can login simultaneously, remain anonymous, post ideas to the whiteboard at the same time, and vote on the best ones. They are ideal for highly freeform brainstorming sessions where the goal is to transfer information directly from a brain's neural network into the virtual whiteboard at the speed-of-light from many simultaneous participants without fear of attribution or retribution (regardless of rank or status). The first thing an executive will do will be to start deleting ideas, so be sure to turn off superuser privileges to capture everyone's ideas. Traditional managers hate virtual whiteboards because they are not highly structured integrated master schedules (IMSs) or list-driven application lifecycle management (ALM) tools where solo left-brained analysts function best. Although virtual whiteboards are one of the most agile tools in the arsenal, they are often one of the most overlooked by traditional project managers turned product owner or scrummaster.

31. Com·mon • Desk·top (*kŏm'ən • dĕsk'tŏp'*) Table, bench, counter, surface, workspace; To provide low-cost commercial distributed persistent virtual computer desktop for open and transparent communication and collaboration

- ✓ Acquire low-cost commercial virtual desktop tool.
- ✓ Best if used by teams for daily working sessions.
- ✓ Ability to support persistent information a plus.

A forgotten principle of lean-agile distributed projects and teams is a common desktop. Any personal computer, laptop, tablet, or smartphone comes with the notion of a desktop. This is a place of temporary storage, analysis, brainstorming, and the place of pure creative thought stuff, which becomes a person's second brain or mirror image of one's brain (where their most valuable intellectual property resides). Perhaps an individual is faced with a problem, goal, or objective, browses the Internet, and downloads a few files from their desktop. These artifacts represent creativity in its rawest form. Knowledge workers will often work on secret projects, while only revealing a small fraction of their ideas to their greater program, project, or team like the tip of the iceberg. Furthermore, desktops serve as poison pills, where knowledge workers can break glass in case of emergency and delete files from their desktops if they get disgruntled and leave, get promoted, or get fired. Knowledge workers will participate in many brainstorming meetings or working sessions with their teams, hold a temporary project hostage on their desktop for weeks, and may never reveal it to the rest of their team. Furthermore, they'll only demonstrate it to an executive or show parts of it on teleconferencing services, never to commit it to a content management system like Confluence, SharePoint, an agile application lifecycle management (ALM) tool, or other version control system. Extreme Programming (XP) recognized this weakness of personal desktops and insisted pair programming teams commit all working code to version control tools, and have a continuous integration service automatically test, build, and deploy it too. This also served other values of the XP paradigm such as open workspaces, information sharing, common code ownership, etc. That is, a pair programming team's software was not the personal property of an individual knowledge worker but was the community property of a program, project, or team. Although tools like Confluence, SharePoint, and ALM systems seek to serve as community repositories, the basic default storage system of Microsoft Windows, PCs, and laptops is the personal desktop where 99% of the community property resides. Law enforcement teams sought to overcome the limitations of keeping one's intellectual property to themselves and formed cockpits, community desktops, and other crime-board like workspaces where all data referring to an incident, case, or investigation could be persistently posted and shared. We need to go back to that model, Microsoft Windows should have a team desktop, and team members should be prevented from holding intellectual property hostage on their personal desktops for self-aggrandizement and perpetuation of fierce Western individualism.

Narcissists will find a way to keep team intellectual property hostage that is a direct result of team collaboration. Some organizations archive desktops, but teams still cannot openly access this valuable participative team produced data.

32. E·mo·tion·al • In·tel·li·gence (*ĭ-mō'shə-nəl • ĭn-tĕl'ə-jəns*) Social, friendly, communal, congenial, empathy; To create a team, culture, and environment of well-managed, constructive, and generative social interactions

- ✓ Teach and expect emotional intelligence.
- ✓ Create a culture of emotional intelligence.
- ✓ Assess and improve emotional intelligence.

The single most important principle of lean-agile distributed projects and teams is emotional intelligence. It is the personal skill, talent, ability, maturity, and wisdom to carefully manage your base animalistic emotions to deflect stress, anger, and conflict. It involves listening to and empathizing with others before speaking. More importantly, it involves communicating calmly, rationally, intellectually, and professionally at all times to arrive at mutually beneficial outcomes. It simply means to exhibit a calm, cool, and collected demeanor, as well as exemplary soft, people, and social skills. The five major dimensions of emotional intelligence are self-awareness (knowing your strengths and weaknesses), self-regulation (remaining calm), social skills (carefully managing relationships), empathy (theory of mind to place other's feelings first), and motivation (leverage people's interests for the benefit of the whole). Other examples include thinking about feelings, taking pause, controlling your emotions, accepting criticism, exhibiting authenticity, empathetically listening, generously and graciously praising others, asking thoughtful questions, apologizing when people are put out, quickly forgiving people who offend you, being reliable, helping people become self-sufficient, and immunity to emotional high jacking. Agile programs, projects, and teams are emotionally intense experiences. The goal of agile teams is to create innovatively new products and services on a short deadline and shoestring budget with people you've never met. Like the financial industry that thinks and plans in quarterly cycles, agile teams must create innovatively new minimum viable products (MVPs) in only 90 days. That means you have to form, storm, norm, perform, and adjourn (deliver) in only 3 months. Your team will rapidly exhibit the full range of emotions (i.e., shock, denial, fear, anger, panic, guilt, depression, and acceptance if you're lucky). All-the-while, your small agile team has to plan, analyze, design, create, test, deliver, and improve an MVP in a short period of time without self-destructing so you can get paid, not get fired, and maintain your reputation as a team player (i.e., someone who embraces the lean-agile mindset). Don't forget that your teammate is probably a former traditional project manager who will add a dozen new stories to the Scrum backlog every day, show up 15 minutes late for every other daily standup, brownnose with executives like a movie star, and embody a divide-and-conquer approach so you'll do all the work. What's the bottom line? It's gonna take the combined emotional intelligence of Mahat Magandi, Nelson Mandela, Martin Luther King, and Thich Nhat Hanh to survive your first 90 days with a traditional narcissistic project manager who measures your performance with a stack of business requirements, integrated master schedules (IMSs), earned value management (EVM), Net Promoter Scores (NPS), and Friday night lean coffees while it is only 1:00 pm in their time zone. This is where multi-cultural sensitivity resides.

Agile Distributed Team Summary

So, what have we learned on this short treatise on how to run successful distributed lean-agile programs, projects, and teams? Well, we've learned to apply lean thinking principles, agile methods, small groups, intensive teamwork and collaboration, and simple low-WIP, low-cost business experiments to tease out uncertainty and risk in short iterations. Conversely, we've learned to avoid large teams of uncooperative individuals; avoid building expensive, complex, and over scoped systems; and avoid traditional process and document intensive waterfall systems. And we've learned that integrated master schedules (IMSs), earned value management (EVM), business requirements, enterprise architectures, and traditional metrics provide little overall value. We've also learned that the single most important principle of lean-agile distributed projects and teams is to apply lean-thinking at all levels, from the client to the provider. This purpose of lean thinking is to quickly deliver innovatively new products and services to markets, customers, and end-users with the shortest possible lead and cycle times. It's a pull vs. push driven philosophy that's fundamentally driven by market, customer, and end-user requirements, demands, and needs. More importantly, we've learned that product and service specifications exist as tacit, hidden, and inexpressible customer, market, and end-user needs, therefore it requires many rapid lean-agile rinse-and-repeat cycles to get-it-right, satisfy those needs, and delight the socks off customers, markets, and end-users. We've also learned that the second most important principle of lean-agile distributed projects and teams is lean leadership at all levels. Lean thinking cannot be delegated to a robot or automaton, it must originate at the very top of the executive food chain, permeate all levels of management, and exist at all levels of the delivery teams (especially team leads).

32 Principles and Practices for Successful Lean-Agile Distributed Programs, Projects, and Teams

1. **Lean Thinking**—To apply pull-based, just-in-time, and waste-free product and service delivery principles
2. **Lean Leadership**—To direct, prepare, and practice application of pull-based, just-in-time, and waste-free principles
3. **Agile Framework**—To apply proven highly-cohesive lean-agile portfolio, program, and project frameworks
4. **Small Teams**—To form and apply small teams of people with complementary technical skills, abilities, and talents
5. **Teaming Agreements**—To form, adhere to, and continuously improve a set of teamwork values, principles, and practices
6. **Team Rotation**—To frequently change team composition to share skills, maintain dynamism, and enhance performance
7. **Cross Training**—To share technical skills among technical teams to improve synergy productivity, performance, and value
8. **Lean Meetings**—To routinely, regularly, and voluntarily hold a few, sparse, short, essential and time-based meetings
9. **Meeting Etiquette**—To form and improve a set of immutable lean-agile meeting values, principles, and practices
10. **Innovation Time**—To allow extra time for creativity, invention, novelty, thought, uncertainty, and unpredictability
11. **Complete Transparency**—To share all data, information, and communications both upwards and downwards

12. **Iterative Delivery**—To expect all teams to deliver working, valuable products and services at frequent regular intervals
13. **Small Batches**—To expect all teams to deliver small, working, and valuable products and services at frequent intervals
14. **Limited WIP**—To minimize the unnecessary materials and work necessary to quickly deliver value-adding features
15. **Keep It Simple**—To keep construction of working products and services as small, simple, narrow, and easy as possible
16. **Frequent Retrospectives**—To conduct frequent root cause analysis and process improvement activities at regular intervals
17. **Anonymous Feedback**—To collect individually unattributable suggestions for process improvement during retrospectives
18. **Simple Metrics**—To collect the bare-minimum metrics necessary to capture the essence of project and team performance
19. **Visual Dashboards**—To configure a small set of aggregated lean-agile metric, model, and measurement dashboards
20. **Virtual Infrastructure**—To apply a low-cost commercial distributed information technology platform for use by teams
21. **Endpoint Devices**—To provide state-of-the-art devices to teams to access the client's virtual infrastructure and collaborate
22. **Messaging Platform**—To provide low-cost commercial distributed text, chat, and instant messaging services for teams
23. **Teleconference Services**—To provide low-cost commercial distributed video conferencing services for teams
24. **Virtual Minutes**—To provide low-cost commercial distributed easy-to-use applications to capture instant meeting minutes
25. **Productivity Tools**—To provide new, state-of-the-art office productivity suites for communication and collaboration
26. **Agile Tools**—To provide low-cost commercial distributed lean and agile application lifecycle management tools
27. **Visual Intensiveness**—To provide low-cost commercial distributed visualization lean and agile tools and systems
28. **Content Management**—To provide low-cost commercial distributed lean and agile wiki tools and systems for teams
29. **Common Repositories**—To provide low-cost commercial distributed persistent electronic file storage space for teams
30. **Virtual Whiteboards**—To provide low-cost commercial distributed easy-to-use brainstorming collaborative space for teams
31. **Common Desktop**—To provide low-cost commercial distributed persistent virtual desktop for openness and transparency
32. **Emotional Intelligence**—To create an environment of well-managed, constructive, and generative social interactions

The third most important principle of lean-agile distributed projects and teams is to apply an agile framework. There are many lean-agile frameworks, methodologies, practices, techniques, tools, and metrics in the global marketplace. There's a veritable smorgasbord of lean-agile ideas, approaches, and piece-part components from which to choose a la cart. However, there are only a few cohesive frameworks that have reached a level of maturity necessary to exhibit the proper lean thinking behaviors. That is, identify value, map the value stream, create flow, establish pull, and continuously improve. These include, but are not limited to Scrum, Scrumban, and even the Scaled Agile Framework (SAFe) for larger teams of teams, programs, solutions (systems of systems), portfolios, and enterprises. So, beyond establishing a solid, strong, and supportive organizational fabric of lean thinking and lean leadership, individual projects and teams must consistently apply proven cohesive agile frameworks. The teams and individuals must be trained, certified, and experienced in the chosen agile framework. However, it's also important for leaders, middle managers, overseers, and interfacing teams and individuals to be trained, certified, and experienced in the chosen agile framework. The teams and individuals must voluntarily desire to follow, comply with, and utilize the chosen agile framework. There's no sense in having a single individual be willing to apply a given agile framework, but everyone else is diametrically opposed to applying the agile framework. This simply will not work, will be ineffective, and be frustrating at best, which, unfortunately happens all of the time. If the teams select Scrum as their framework, then the teams should perform sprint planning, daily standups, sprint reviews, retrospective, and even backlog refinement (if necessary), no ifs, ands, buts, or excuses.

A critically important principle of lean-agile distributed projects and teams is to form very small agile delivery teams. The best team size is around two to three people with complementary skills, abilities, experience, and desires. An absolutely essential principle of lean-agile distributed projects and teams is lean meetings. That is, institute only the bare minimum number of fast, efficient, and critical lean-agile ceremonies and meetings. One of the most critically important principles of lean-agile distributed projects and teams is innovation time. The more uncertain, volatile, risky, and innovative the problem domain is, the more innovation time is required to address it (40% to 50% buffer). An essential principle of lean-agile distributed projects and teams is complete transparency, which means 360° visibility into all program, project, and team data. Another important principle of lean-agile distributed projects and teams is iterative delivery. Agile teams should deliver something of value about every two weeks. This may be one or more small business experiments, a small minimum viable product (MVP), or a solution that satisfies a two-week iteration or sprint goal. Another essential principle of lean-agile distributed projects and teams is small batches. That is, programs, projects, and teams should plan a series of small business experiments, epics, minimum viable products (MVPs), or features. Another critical principle of lean-agile distributed projects and teams is limited WIP. Focus on a few small business experiments; and build utterly simple minimum viable products (MVPs) to iteratively tease tacit, hidden, and inexpressible market, customer, and end-user needs a little bit at time. One of the most valuable principles of lean-agile distributed projects and teams is frequent retrospectives, which are the truly the secret sauce to distributed team performance (especially if it is gamified and anonymous).

Agile teams should consistently apply simple agile (gamified) metrics, visual dashboards, and visual Scrum or Kanban boards. It goes without saying that successful distributed lean-agile teams must be provided with state-of-the-art information technologies. These include, but are not limited to endpoint devices (laptops), messaging platforms (Skype), teleconference services (Zoom), virtual minutes (OneNote), productivity tools (Microsoft Office), agile tools (Rally), content management (Confluence), common repositories (SharePoint), virtual whiteboards (Mural), etc. The single most important principle of lean-agile distributed projects and teams is emotional intelligence. It is the personal skill, talent, ability, maturity, and wisdom to carefully manage your base emotions to deflect stress, anger, and conflict. It involves listening to and empathizing with others before speaking. It involves communicating calmly, rationally, intellectually, and professionally at all times to arrive at mutually beneficial outcomes. It simply means to exhibit a calm, cool, and collected demeanor, as well as exemplary soft, people, and social skills. The goal of agile teams is to create innovatively new products and services on a short deadline and shoestring budget with people you've never met, so you'll have to muster all of the social skills necessary to form, storm, norm, perform, and adjourn (deliver) in 3 months.

Further Reading

- Ambler, S., & Aguanno, K. (2010). *Adapting agile for use with distributed teams*. Oshawa, Canada: Multi-Media Publications.
- Anderson, R. M. (2020). *Leading in the next normal*: *A guide to building an engaged, resilient, and agile virtual workforce*. Haslemere, UK: Executive Joy Publishing.
- Bolboaca, A. (2021). *Practical remote pair programming*: *Best practices, tips, and techniques for collaborating productively with distributed development teams*. Birmingham, UK: Packt Publishing.
- Bourgault, M., & Drouin, N. (2009). *Understanding decision making within distributed project teams*. Newtown Square, PA: PMI.
- Chellappa, S. (2011). *The black book of agile project delivery with distributed teams*. Nashville, TN: eMIDS Technologies.
- Cunha, M. M. (2006). *Agile virtual enterprises*: *Implementation and management support*. Hershey, PA: Idea Group Publishing.
- Douglas, T., Gordon, H., & Webber, M. (2020). *Working remotely*: *Secrets to success for employees on distributed teams*. NY, NY: Kaplan.
- Eckstein, J. (2010). *Agile software development with distributed teams*: *Staying agile in a global world*. New York, NY: Dorset House.
- Goldman, S. L., & Preiss, K. (1995). *Agile competitors and virtual organizations*: *Strategies for enriching the customer*. New York, NY: Van Nostrand Reinhold.
- Goncalves, P. (2009). *Interfaces between end-users in agile and virtual enterprises*. Saarbrucken, DE: VDM Verlag.
- Gotz, O. (2017). *Distributed agile outsourcing*. *An overview of methods and success factors*. Norderstedt, DE: GRIN Verlag.
- Kasriel, S., & Morgan, J. (2014). *Hire fast and build things*: *How to recruit and manage a top-notch team of distributed engineers*. Santa Clara, CA: Elance-oDesk.
- Konig, V. J. (2019). *Agile software development in distributed teams*: *A systematic analysis of the challenges in relation to different cultures*. Charleston, SC: CreateSpace.
- Martinelli, R. J., Waddell, J. M., & Rahschulte, T. J. (2017). *Projects without boundaries*: *Successfully leading teams and managing projects in a virtual world*. Hoboken, NJ: John Wiley& Sons.
- Mayer, M. (1998). *The virtual edge*: *Embracing technology for distributed project team success*. Sylva, NC: PMI.
- Mazur, M., & Gonzalez, A. (2020). *Managing remote employees*: *The ultimate guide*. Bogota, CU: Mazgobooks.
- McNeese, M., Salas, E., & Endsley, M. R. (2021). *Fields of practice and applied solutions within distributed team cognition*. Boca Raton, FL: CRC Press.
- Messer, H., Pahuja, S., & Okoro, J. (2018). *A guide to distributed agile framework*. Charleston, SC: CreateSpace.
- Milhauser, K. L. (2011). *Distributed team collaboration in organizations*: *Emerging tools and practices*. Hershey, PA: Business Science Ref.
- Mite, D., Moe, N. B., & Ågerfalk, P. J. (2010). *Agility across time and space*: *Implementing agile methods in global software projects*. Berlin, Germany: Springer-Verlag.
- Nayab, S., Anwar, S., & Sagheer, S. (2015). *Challenges of implementation of agile in distributed environment*. Norderstedt, DE: GRIN Verlag.
- O'Duinn, J. (2021). *Distributed teams*: *The art and practice of working together while physically apart*. San Francisco, CA: Release Mechanix.
- Orti, P., & Middlemiss, M. (2019). *Thinking remote: Inspiration for leaders of distributed teams*. London, UK: Virtual Not Distant.
- Pearson, C., & Porath, C. (2009). *The cost of bad behavior*: *How incivility is damaging your business and what to do about it*. New York, NY: Penguin Group.
- Pfeiffer, S., Nicklich, M., & Sauer, S. (2021). *The agile imperative*: *Teams, organizations, and society under reconstruction*. London, UK: Palgrave-MacMillen.
- Qu, Z. (2021). *Remote delivery*: *A guide to software delivery through collaboration between distributed teams*. Boca Raton, FL: CRC Press.
- Rico, D. F. (2010). *The paradox of agile project mgt. and virtual teams*. Fairfax, VA: Gantthead, http://davidfrico.com/rico-apm-virtual.pdf
- Rico, D. F. (2011). *Lean & agile project mgt. for large distributed virtual teams*. Retrieved, May 19, 2011, from http://davidfrico.com/rico16i.pdf
- Rico, D. F. (2016). *The 10 attributes of successful teams, teamwork, and projects*. Retrieved September 26, 2016 from http://davidfrico.com/teamwork-attributes-2.pdf
- Rico, D. F. (2016). *The 12 attributes of successful collaboration between highly creative people*. Retrieved February 29, 2016, from http://davidfrico.com/collaboration-attributes.pdf
- Rico, D. F. (2018). *Lean & agile contracts*: *21 principles of collaborative contracts and relationships*. Retrieved June 29, 2018, from http://davidfrico.com/collaborative-contract-principles.pdf
- Rico, D. F. (2018). *Using SAFe 4.5 to transform a $200 million U.S. healthcare portfolio*. Retrieved November 19, 2018 from http://davidfrico.com/safe-case-study-ii.pdf
- Rico, D. F. (2019). *Business value of lean leadership*: *22 Attributes of successful business executives, directors, and managers*. Retrieved April 27, 2019, from http://davidfrico.com/rico19a.pdf or http://davidfrico.com/lean-leadership-principles.pdf
- Rico, D. F. (2019). *Piloting the scaled agile framework (SAFe) in a top 10 U.S. energy firm*. Retrieved May 15, 2019, from http://davidfrico.com/x-safe-case-study-iii.pdf
- Rico, D. F. (2020). *32 principles and practices of highly successful SAFe implementations*. Retrieved February 16, 2020, from http://davidfrico.com/safe-principles.pdf
- Rico, D. F. (2020). *Business value of lean thinking*: *Capitalizing upon lean thinking principles to rapidly create innovative products and services*. Retrieved January 29, 2020, from http://davidfrico.com/rico20b.pdf or http://youtu.be/wkMfaPAxO6E
- Rico, D. F. (2020). *Using large solution SAFe in a high-profile U.S. civilian public sector agency*. Retrieved January 29, 2020, from http://davidfrico.com/y-safe-case-study-iv.pdf
- Rico, D. F. (2021). *32 principles and practices for a highly successful agile contract statement of work (SOW)*. Retrieved March 22, 2021, from http://davidfrico.com/agile-sow-principles.pdf
- Rico, D. F. (2021). *32 principles and practices for maximizing the ROI of SAFe program increment (PI) planning*. Retrieved March 14, 2021, from http://davidfrico.com/safe-roi-principles.pdf
- Rico, D. F. (2021). *Business value of organizational agility*. Retrieved March 26, 2021 from http://davidfrico.com/rico21a.pdf or http://davidfrico.com/value-of-business-agility.pdf or http://youtu.be/HOzDM5krtes
- Rico, D. F., Sayani, H. H., & Sone, S. (2009). *The business value of agile software methods*: *Maximizing ROI with right-sized processes and documentation*. Ft. Lauderdale, FL: J. Ross Publishing.
- Rothman, J., & Kilby, M. (2019). *From chaos to successful distributed agile teams*: *Collaborate to deliver*. Victoria, Canada: Practical Ink.
- Silviera, A. (2021). *Building and managing high-performance distributed teams*: *Navigating the future of work*. New York, NY: Apress Media.
- Singh, A., Singh, K., & Sharma, N. (2015). *Distributed agile development system*: *A paradigm shift*. Chisinau, MLD: LAP Lambert.
- Sroka, J. (2007). *Distributed teams*: *An introduction to virtual teams and organizations*. Saarbrucken, DE: VDM Verlag.
- Steele, C. (2014). *Appraisal and motivation techniques in distributed agile teams*. Chisinau, MLD: LAP Lambert.
- Upadrista, V. (2008). *Managing offshore development projects*: *An agile approach*. Oshawa, Canada: Multi-Media Publications.
- Venkatesh, U. (2011). *Distributed agile DH2A*: *The proven agile software development approach and toolkit for geographically distributed teams*. Westfield, NJ: Technics Publications.
- Woodward, E., Surdek, S., & Ganis, M. (2010). *A practical guide to distributed scrum*. Indianapolis, IN: IBM Press.

Case Studies of Successful Lean-Agile Distributed Programs, Projects, and Teams

- Public Sector Manufacturer. *Around 2010, a top 5 U.S. public sector manufacturer began its journey to lean and agile thinking. This was rather surprising, since this manufacturer was one of the most staunchly traditional firms in the U.S. if not the world. Its culture was thoroughly steeped in integrated master schedules (IMSs), earned value management (EVM), and linear systems and software engineering lifecycles. Many of its products and services were extremely complex three-and-four-decade long hardware-intensive systems with unusually long lead time components. In other words, it wasn't in a big hurry, markets weren't going anywhere anytime soon in its segment, and its customers were unusually deep-pocketed and patient when it came to high-risk never-ending multibillion dollar acquisitions. Commitment to lean and agile thinking began in its information technology services sector, which is no surprise and spread like wildfire throughout its corporation within 10 years by 2020. About that time, it began in earnest to convert all of its acquisition programs to lean and agile thinking principles, practices, and tools, even if the buyers did not explicitly ask to do so. Small 2-3 person teams of agile coaches were hired to help convert as many of its hardware and software intensive acquisition programs to lean and agile thinking as possible. Although these teams were meant to be small, collocated groups, a global pandemic altered the course and forced all of them to work in a distributed fashion from the very start. One of the teams was assigned to transform a 100-person information technology system from traditional to a largescale lean-agile framework. The team was trained, certified, and experienced in that framework. They were also highly motivated and began in earnest to help transform the program. Fortunately, the acquisition program's leadership was all in, convinced the buyer (government) to go along with the transformation and helped drive the change personally. The small coaching team trained all of the program's participants in the agile framework and helped kickoff its planning ceremonies, as well as coach its sprint or iteration level ceremonies too. The coaching team itself used Scrum, Rally, and Confluence as its basic operating model, and applied sprint planning, daily standups, demos, retrospectives, and backlog refinement. It used visual Scrumban boards to plan and manage its own work. All of the transformation activities were performed as a team using pair programming and one-piece workflow principles. Intensive teamwork, collaboration, and emotional intelligence were used to successfully form, norm, storm, perform, and adjourn very quickly. The manufacturer provided all agile coaching teams with state-of-the-art laptops, virtual infrastructure, and a variety of other virtual application services. The acquisition program itself did well, improved productivity and output by up to five times, and responded well to the virtual coaching team. In fact, 100% virtual coaching helped enforce emotional intelligence principles and minimize team conflict, but not eliminate it. Daily conflict management was necessary to keep the team as stable as possible at all times.*
- Public Sector Services Supplier. *Around 2010, a top 15 U.S. public sector services supplier also began its journey to lean and agile thinking. It successfully won a large U.S. government acquisition. Although its parent corporation was a bit slow to adopt lean and agile thinking as an enterprise, this acquisition program clearly one of the flagship lean and agile programs. It began slowly with Scrum practices on a few software development teams and made headway very quickly. By 2017, it graduated to a larger agile framework and began in earnest to apply lean and agile thinking to all of teams on its acquisition program. The program operated in a face-to-face mode for much of its existence but was forced to become a distributed program in 2020 with the onset of the global pandemic. At this point, the acquisition program was forced to perform as a large, distributed team. Many of its components were already distributed domestically and internationally before the pandemic, but merely forced all teams to operate in a distributed manner in 2020. The teams successfully ran large scale distributed planning events throughout 2020, as well as team level events such as sprint planning, daily standups, demos, retros, backlog refinement, and even systems engineering and architecture. The buyer (government) was very pleased with the supplier's performance and awarded them yet another five-year contract to continue using its large-scale agile framework to help modernize the legacy components. Following the win, the acquisition contract performed yet another large virtual distributed planning event. The basic platform was Microsoft Windows, with Skype, productivity tools, and other modern content management and agile application lifecycle management tools. The largescale agile framework provided instant visualization and situational awareness into the largescale complex acquisition program at all times. Because of the virtual information technology infrastructure, highly cohesive team culture, above average emotional intelligence, and highly competent technical skills, this acquisition program was able to quickly switch over from face-to-face agile team operation to fully distributed team operation rather seamlessly. The acquisition program focused on intense teamwork and collaboration, frequent teleconferencing, openness, transparency, empowerment, self-organization, and visualization through the use of heavy investments in agile ALM tools and content management systems. The scrum of scrums meetings were also held virtually, in addition to program management, systems engineering, architecture, infrastructure, and agile transformation teams. Virtual messaging is one of he most prevalent tools, along with teleconferencing services, and of course the agile ALM tool itself. Lean thinking is applied at all levels and innovation time is provided for all teams implicitly and explicitly, which is sort of a norm for agile teams in general. The program is quickly expanding its commitment to largescale agile frameworks, expanding its agile transformation teams, and transitioning to the use of DevOps to help reduce lead and cycle times, as well as forming high-performing cross-functional development and IT teams.*
- Private Sector Energy Firm. *Around 2019, one of the largest energy firms in the world began a largescale digital transformation initiative. Much of his involved its centralized corporate-level information technology department and chief information officer (CIO), since most of the IT departments from each of its operating units had been consolidated into the central IT department. The goal of the digital transformation initiative was to create a new line of mobile applications for providing consumers with access to its energy products and services (private home energy utility services). Several dozen Scrum teams were kicked off in order to rapidly deliver as may mobile and web apps as possible in an effort to defeat its legacy traditional thinking culture that planned project delivery in 5-10-and-15-year cycles. Most of the Scrum leadership teams were domestic groups, while most of the development and delivery teams were offshore Scrum teams. However, all of the teams, both leadership and development and delivery were distributed teams. The delivery teams themselves, as well as the Scrum leadership teams were outsourced to an external management consulting services firm. Not only that, but Scrum leadership teams multitasked on multiple client engagements other that energy firm's digital transformation initiative. Therefore, all Scrum sprint planning, daily standup, demo, retro, backlog refinement, and even Scrum of Scrums (SoS) meetings were held virtually, most of the time. Sometimes the Scrum leadership teams were brought to the client's site but remained within office hotel sweets with little to no interaction, continuing to hold Scrum ceremonies via Skype. In other words, there was little to no face-to-face interaction, even when the Scrum leadership teams were at the client site. Eventually after the first year of the energy firm's digital transformation initiative, a larger scale agile framework was applied to integrate the work of multiple Scrum teams into a cohesive whole. The transition from basic Scrum to the larger scale agile framework was relatively seamless, as the first year was spent mastering Scrum, so the initial group planning event was held in only one day. Overall, the large group planning event went well, and both buyer and supplier had unprecedented transparency, insight, and situational awareness into the scale, scope, size, health, and status of most of the Scrum teams simultaneously for the first time. Not all of the Scrum teams were as successful as those at the central client site where the digital transformation team operated. Some agile teams were stuck in first gear and simply couldn't get their delivery sprints started. Other teams at their corporate headquarters struggled, where traditional thinking was very strong. Overall, there was a very traditional thinking fabric operating behind the scenes, which was only temporarily suppressed by the presence of the digital transformation team. The digital transformation team made remarkable progress with Scrum, distributed teams, self-organizing offshore delivery teams, and implementation of the large scale agile framework. However, it was only skin deep as a traditional thinking fabric existed in the background that took longer to change.*