Traditional vs. Agile Methods — Assumptions and Theoretical Tenets

Traditional Assumptions

- Good methodologies should be proprietary in order to maximize their efficiency and effectiveness (untrue).
- 100% of project/product scope and requirements can be known and specified in-advance (untrue).
- Productivity and quality do not break down with scale and size (untrue).
- Perfection is achieved with perfect specifications and no customer interaction, demonstrations, and involvement (untrue).
- Traditional Methods are scalable to any size problem (untrue).
- Manual processes and inspections are superior practices (untrue).
- Customers don't change their mind during the project or after the project is done (untrue).
- All knowledge is explicit and can be captured in documentation (untrue).
- A disciplined process can be substituted for talented engineers (untrue).
- Most people are left-brained, analytical types that naturally succeed with data, facts, figures, math, and statistics (untrue).
- Zero defects is the most important measure of market/business value, project success, and quality (untrue).
- Zero defects implicitly translate into secure software (untrue).
- Customer satisfaction is achieved by using in-process reviews of interim work-in-process/products (i.e., requirements, designs, code, etc.) to bake-in quality over a long period of time until the product is just right long before the customer sees it (untrue).
- Well defined processes, requirements, architectures, and tests, lead to better cost and schedule accuracy, greater near-term quality and reliability, and lower long term maintenance costs (untrue).
- ROI is achieved by spending a little now on upfront systems engineering to reap a deterministic delivery schedule, make customers happy during operational use, and save tons of money with low-maintenance costs (untrue).
- Upfront systems engineering, bureaucratic processes, elongated development cycles, and enormous budgets are justified when producing mission and safety-critical systems (untrue).

Agile Assumptions

- Good methodologies should be free (so they can be refined, perfected, improved, extended, and free for everyone to use).
- Only about 1 to 3% of scope/requirements should be attempted (70% tacit, 20% wrong, and 7% not needed).
- Direct customer interaction is needed to capture the 1% to 3% of valid, high-priority requirements that exist as tacit knowledge.
- Requirements should be implemented and validated in small iterations (to continue eliciting tacit knowledge and requirements).
- Small teams should work together to maximize exchange of tacit knowledge (which can't be captured explicitly in documentation no matter how hard you try, how many years take, how many people you use, or how much money you spend).
- Processes and documents should be kept to a bare minimum and managed electronically (to minimize the overhead cost of creating wasteful documentation that is instantly obsolete after it is created and before the ink dries anyway).
- Low-cost, flexible technologies should be used to minimize overhead expenses, maximize delivery speed and quality, and support rapid, iterative development, procurement, acquisition, and engineering.
- Talented engineers are needed to succeed with disciplined processes.
- Most people are right-brained, conceptual types who thrive on verbal communications, pictures, visions, motivation, empowerment, and making an impact on business and society.
- Automate 90% or more of the process with free and open source technologies and perform 50 times more quality engineering tests than can be performed manually by placing the burden on machines to achieve zero defects (rather than manual ones).
- Software must be made secure on purpose, using deliberate software security engineering policies, staffing, skills, personnel, practices, tools, and measurements (i.e., it isn't an incidental or accidental side-effect or by-product of having zero defects).
- Customer satisfaction is achieved by iteratively developing working products to obtain customer and market feedback as early and often as possible in order to rapidly refine the end-product, rather than the interim work-in-process/work products.
- It's more important to focus on delivering a small set of customer needs that will improve profitability or mission effectiveness (than it is to have a large zero defect code base that nobody needs).
- Profitability, business goal/objective satisfaction, mission effectiveness, capability validation, customer satisfaction, teamwork, process effectiveness and efficiency, and flexibility and adaptability to change are more important than "zero defects."
- Long-term plans are expensive to produce and fruitless; heavy processes and documents are expensive, unnecessary, cause delays, and create more problems than they solve; and manually-intensive quality engineering is too expensive with little effect.
- You can go bankrupt embracing with traditional methods, because 70% to 90% of requirements exist as tacit knowledge, large scopes are attempted, and the result is a 70% to 90% failure rate (out of blind-trust in the traditional methods).
- Success comes from a small scope, near-term plans, few high-priority requirements, lightweight design, and frequent demos to elicit tacit requirements, reduce the size risk, and converge on a solution in a shorter timeframe, lower cost, and higher-quality.
- Agile Methods tackle ROI from a two perspectives: (1) lower-cost, faster-cycles, higher-quality, and satisfied customers; and (2) higher business value by rapidly converging on a solution to improve market share, revenues, profits, and mission success.
- There's also the feasibility factor, i.e., large scope projects fail more often, whereas smaller scope projects are more successful, therefore, more is less with traditional methods, and less is more with agile methods.
- Everything in moderation is good (i.e., plans, processes, requirements, architecture, tests, etc.), but the key term is "moderation."