Common Myths & Important Realities of Agile Methods

Common Myths

- **Lacks Formal Documentation** - Project, architecture, design, test, CM, QA, etc.
- **Lacks Requirements** - Prototyping approach used when there are no formal requirements.
- **Lacks Architecture** - Doesn't take the time to develop a formal broad architectural framework.
- **Doesn't Scale Up or Out** - Only for small collocated F2F non-virtual teams using verbal communications.
- **Lacks Quality Focus** - Sacrifices quality for speed, productivity, programming, cycle time, etc.
- **Increases Risk** - Lacks rigor and discipline necessary for mission and safety-critical systems.
- **Lacks Governance** - Doesn't use groups like configuration management, quality assurance, testing, etc.
- **Lacks Discipline** - Approach used by coding cowboys who don't want to use disciplined processes.
- **Doesn't Consider Maintenance** - Only for rapid prototypes that don't require long-term documentation and quality.
- **Only for Computer Programmers** - Can't be used for functions such as business, administration, engineering, hardware, etc.
- **Only for Simple IT Systems** - Not for large and complex systems with large budgets, requirements, groups, timelines, etc.
- **Only for Software Systems** - Not for expensive hardware or embedded systems that don't support rapid iterative development.
- **Not for Regulated Markets** - Lacks rigor, discipline, and formality for DoD, FAA, FDA, NASA, and other safety-critical domains.
- **Not for Rigid Contracting** - Only for level of effort time-and-materials labor contracts with flexible scope, budgets, timelines, and expectations.
- **Doesn't Support Non-Functional Requirements** - No support for quality, reliability, safety, dependability, usability, security, maintainability, etc.
- **Not Aligned with Government Culture** - Government contracting locked into firm fixed-price contracting culture aligned with rigid traditional methods.

Important Realities

- **Cultural Mismatch** - New systems development paradigm to which people are unaccustomed causing resentment and distrust.
- **Resistance to Change** - People will reject a NEW approach whether the edict comes from executives, middle managers, or technical personnel.
- **Top-Down Organizational Change** - Attempt another top-down big bang organizational rollout, which may increase chaos, fear, and resistance.
- **Ignore Training** - Projects, teams, and individuals expected to apply them without formal training, learning, coaching, mentoring, or experience.
- **Business Misalignment** - Failure to elicit high-priority requirements from key stakeholders and deliver those first (*and use projects to learn new skills*).
- **Scale Too Big** - Fail to de-scope, downsize, and focus upon a smaller set of customer needs, requirements, scope, architecture, implementation, etc.
- **Minimalistic Guidelines** - Only has a broad lightweight project framework so rigor and discipline is voluntary, skill, and experience based.
- **Ignore Quality Control** - Teams don't have experience, skill, training, or motivation to apply advanced testing practices to verify and validate systems.
- **Traditional Focus** - Use agile methods to incrementally implement a large project scope, requirements document, or formal system architecture.
- **Backsliding** - Gradually backslide into a traditional, long-term plan and document intensive paradigm out of fear, lack of trust, and lack of courage.
- **Scrummerfalling** - Incrementally produce plans, requirements, architectures, designs, and tests instead of developing validated code each iteration.
- **Hardware Focus** - Design customized FPGA hardware boards vs. running signal processing algorithms as application software on commodity PCs.
- **Ignore Infrastructure & Automation** - Assume agile methods are a simple manual process and don't establish an IT infrastructure with FOSS tools.
- **Individualism** - Fail to engage customers, users, and teammates in critically-important communications, conversations, and decision-making.
- **Plan Driven** - Follow rigid process instead of adjusting the project scope, processes, tools, and documents to converge on a valid set of system needs.
- **Adversarial Contracting** – Continue to use traditional master-slave legalistic structures vs. collaboration, cooperation, egalitarianism, and risk-sharing.

Bottom Line – *Agile methods require training, skill, experience, discipline, tools, and time to yield optimal results* !!!

(*A butterfly flapping its wings in one part of the globe can cause a hurricane in another part, i.e., even small changes have positive or negative impacts ...*)