Agile Technology

Archana Tellunagi

Department of Computer Science and Engineering, College of Engineering

Third Year Engineering Student

History:

By the late 1990s, a majority of the software development processes that had been developed in the 1980s and 1990s were being criticized as bureaucratic, slow, and overly regimented. In the mid-1990s, in reaction to these *heavyweight* software methods, there was a small contingent of industry thought-leaders promoting innovative approaches to software, enabling development organizations to quickly react and adapt to changing requirements and technologies. They realized that embracing change, and executing in a manner that not only accommodated this change, but fostered it, would result in a much more successful development strategy.

The term "agile software development" emerged from a gathering of these industry thought-leaders in Snowbird, UT in 2001. The term was first used in this manner and published in the now famous (or infamous) Agile Manifesto presented to the right.

This term was used as an umbrella reference to a family of emerging lightweight software development methods such as Scrum, Extreme Programming, DSDM, FDD, Crystal, and Adaptive Software Development. Instead of emphasizing up-front planning and detailed requirements, these methods placed significant emphasis on continual planning, empowered teams, collaboration, emergent design, a test-early and often philosophy, and, most importantly, the frequent delivery of working software in short, rapid iterations.

Since the publication of the Agile Manifesto, other thought-leaders have continued to evolve agile thinking, drawing on lessons learned in other industries – for example, in the ideals and approaches promoted by Lean Development, and most recently, Kanban.

What is Agile?

Agile came about as a “solution” to the disadvantages of the waterfall methodology. Instead of a sequential design process, the agile methodology follows an incremental approach.

Developers start off with a simplistic project design, and then begin to work on small modules. The work on these modules is done in weekly or monthly sprints, and at the end of each sprint, project priorities are evaluated and tests are run. These sprints allow for bugs to be discovered, and customer feedback to be incorporated into the design before the next sprint is run.

The process, with its lack of initial design and steps, is often criticized for its collaborative nature that focuses on principles rather than process.

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.

It is used for time critical applications. Extreme Programming (XP) is currently one of the most well known agile development life cycle model.

Agile software engineering embraces a philosophy that encourages customer satisfaction, incremental software delivery, small project teams (composed of software engineers and stakeholders), informal methods, and minimal software engineering work products. Agile software engineering guidelines stress on-time delivery of an operational software increment over analysis and design

- An agile team is able to respond to changes during project development
- Agile development recognizes that project plans must be flexible
- Agility encourages team structures and attitudes that make communication among developers and customers more facile
- Agility eliminates the separation between customers and developers
- Agility emphasizes the importance of rapid delivery of operational software and de-emphasizes importance of intermediate work products
- Agility can be applied to any software process as long as the project team is allowed to streamline tasks and conduct planning in way that eliminate non-essential work products

When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Agile Processes:

- Are based on three key assumptions
- It is difficult to predict in advance which requirements or customer priorities will change and which will not
- For many types of software design and construction activities are interleaved (construction is used to prove the design)
- Analysis, design, and testing are not as predictable from a planning perspective as one might like them to be

- Agile processes must be adapted incrementally to manage unpredictability
- Incremental adaptation requires customer feedback based on evaluation of delivered software increments (executable prototypes) over short time periods.

Agility Principles:

- Highest priority is to satisfy customer through early and continuous delivery of valuable software
- Welcome changing requirements even late in development, accommodating change is viewed as increasing the customer's competitive advantage
- Delivering working software frequently with a preference for shorter delivery schedules (e.g., every 2 or 3 weeks)
- Business people and developers must work together daily during the project
- Build projects around motivated individuals, given them the environment and support they need, trust them to get the job done
- Face-to-face communication is the most effective method of conveying information within the development team
- Working software is the primary measure of progress
- Agile processes support sustainable development, developers and customers should be able to continue development indefinitely
- Continuous attention to technical excellence and good design enhances agility
- Simplicity (defined as maximizing the work not done) is essential
- The best architectures, requirements, and design emerge from self-organizing teams
- At regular intervals teams reflects how to become more effective and adjusts its behavior accordingly

Human Factors:

- Traits that need to exist in members of agile development teams:
- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy-problem solving ability
- Mutual trust and respect
- Self-organization

Agile Process Models:

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Crystal

- Feature Driven Development (FDD)
- Agile Modeling (AM)

Fig:Agile Models

Extreme Programming

- Relies on object-oriented approach
- Key activities
- Planning (user stories created and ordered by customer value)
- Design (simple designs preferred, CRC cards and design prototypes are only work products, encourages use of refactoring)
- Coding (focuses on unit tests to exercise stories, emphasizes use of pairs programming to create story code, continuous integration and smoke testing is utilized)
- Testing (unit tests created before coding are implemented using an automated testing framework to encourage use of regression testing, integration and validation testing done on daily basis, acceptance tests focus on system features and functions viewable by the customer)

Adaptive Software Development

- Self-organization arises when independent agents cooperate to create a solution to a problem that is beyond the capability of any individual agent
- Emphasizes self-organizing teams, interpersonal collaboration, and both individual and team learning
- Adaptive cycle characteristics
- Phases
- Mission-driven
- Component-based
- Iterative
- Time-boxed

- Risk driven and change-tolerant
- Speculation (project initiated and adaptive cycle planning takes place)
- Collaboration (requires teamwork from a jelled team, joint application development is preferred requirements gathering approach, minispecs created)
- Learning (components implemented and testes, focus groups provide feedback, formal technical reviews, postmortems)

Dynamic Systems Development Method

- Provides a framework for building and maintaining systems which meet tight time constraints using incremental prototyping in a controlled environment
- Uses Pareto principle (80% of project can be delivered in 20% required to deliver the entire project)
- Each increment only delivers enough functionality to move to the next increment
- Uses time boxes to fix time and resources to determine how much functionality will be delivered in each increment
- Guiding principles
- Active user involvement
- Teams empowered to make decisions
- Fitness foe business purpose is criterion for deliverable acceptance
- Iterative and incremental develop needed to converge on accurate business solution
- All changes made during development are reversible
- Requirements are baselined at a high level
- Testing integrates throughout life-cycle
- Collaborative and cooperative approach between stakeholders
- Life cycle activities
- Feasibility study (establishes requirements and constraints)
- Business study (establishes functional and information requirements needed to provide business value)
- Functional model iteration (produces set of incremental prototypes to demonstrate functionality to customer)
- Design and build iteration (revisits prototypes to ensure they provide business value for end users, may occur concurrently with functional model iteration)
- Implementation (latest iteration placed in operational environment)

Scrum

- Scrum principles
- Small working teamed used to maximize communication, minimize overhead, and maximize sharing of informal knowledge
- Process must be adaptable to both technical and business challenges to ensure bets product produced
- Process yields frequent increments that can be inspected, adjusted, tested, documented and built on
- Development work and people performing it are partitioned into clean, low coupling partitions

- Testing and documentation is performed as the product is built
- Provides the ability to declare the product done whenever required
- Process patterns defining development activities
- Backlog (prioritized list of requirements or features the provide business value to customer, items can be added at any time)
- Sprints (work units required to achieve one of the backlog items, must fir into a predefined time-box, affected backlog items frozen)
- Scrum meetings (15 minute daily meetings) addressing these questions: What was done since last meeting? What obstacles were encountered? What will be done by the next meeting?
- Demos (deliver software increment to customer for evaluation)

Feature Driven Development

- Practical process model for object-oriented software engineering
- Feature is a client-valued function, can be implemented in two weeks or less
- FDD Philosophy
- Emphasizes collaboration among team members
- Manages problem and project complexity using feature-based decomposition followed integration of software increments
- Technical communication using verbal, graphical, and textual means
- Software quality encouraged by using incremental development, design and code inspections, SQA audits, metric collection, and use of patterns (analysis, design, construction)
- Framework activities
- Develop overall model (contains set of classes depicting business model of application to be built)
- Build features list (features extracted from domain model, features are categorized and prioritized, work is broken up into two week chunks)
- Plan by feature (features assessed based on priority, effort, technical issues, schedule dependencies)
- Design by feature (classes relevant to feature are chosen, class and method prologs are written, preliminary design detail developed, owner assigned to each class, owner responsible for maintaining design document for his or her own work packages)
- Build by feature (class owner translates design into source code and performs unit testing, integration performed by chief programmer)

Advantages of Agile model:

There are eight most important benefits of Agile Model.These are following:

1. Stakeholder Engagement
2. Transparency
3. Early and Predictable Delivery
4. Predictable Costs and Schedule
5. Allows for change

6. Focusing on Business Value

7. Focusing on Customers

8. Improving Quality

Refenence:

- www.e-zest.net/agile_software_development
- ieeexplore.ieee.org
- www.google.co.in
- en.wikipedia.org