Overview

- Agile software engineering represents a reasonable compromise between conventional software engineering for certain classes of software and certain types of software projects.
- Agile development processes can deliver successful systems quickly.
- Agile development stresses continuous communication and collaboration among developers and customers.
- 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.

Manifesto for Agile Software Development

- Proposes that it may be better to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- While the items on the right are still important the items on the left are more valuable under this philosophy.
- Note: although most practitioners agree with this philosophy in theory, many pragmatic issues surface in the real world that may cause items on the right to be as important as items on the left.

Agility

- 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 a way that eliminates non-essential work products.

Agile Processes

- Are based on three key assumptions:
  1. It is difficult to predict in advance which requirements or customer priorities will change and which will not.
  2. For many types of software design and construction activities are interleaved (construction is used to prove the design).
  3. 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:
  o Competence
  o Common focus
  o Collaboration
  o Decision-making ability
  o Fuzzy-problem solving ability
  o Mutual trust and respect
  o 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)

Extreme Programming

• Relies on object-oriented approach
• Key activities
  o Planning (user stories created and ordered by customer value)
  o Design (simple designs preferred, CRC cards and design prototypes are only work products, encourages use of refactoring)
  o Coding (focuses on unit tests to exercise stories, emphasizes use of pairs programming to create story code, continuous integration and smoke testing is utilized)
  o 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
  o Mission-driven
  o Component-based
  o Iterative
  o Time-boxed
  o Risk driven and change-tolerant
  o Speculation (project initiated and adaptive cycle planning takes place)
  o Collaboration (requires teamwork from a jelled team, joint application development is preferred requirements gathering approach, minispecs created)
  o Learning (components implemented and tested, 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
  o Active user involvement
  o Teams empowered to make decisions
  o Fitness for business purpose is criterion for deliverable acceptance
  o Iterative and incremental develop needed to converge on accurate business solution
  o All changes made during development are reversible
  o Requirements are baselined at a high level
  o Testing integrates throughout life-cycle
  o Collaborative and cooperative approach between stakeholders
• Life cycle activities
  o Feasibility study (establishes requirements and constraints)
  o Business study (establishes functional and information requirements needed to provide business value)
  o Functional model iteration (produces set of incremental prototypes to demonstrate functionality to customer)
  o Design and build iteration (revisits prototypes to ensure they provide business value for end users, may occur concurrently with functional model iteration)
  o Implementation (latest iteration placed in operational environment)

Scrum
• Scrum principles
  o Small working teamed used to maximize communication, minimize overhead, and maximize sharing of informal knowledge
  o Process must be adaptable to both technical and business challenges to ensure bets product produced
  o Process yields frequent increments that can be inspected, adjusted, tested, documented and built on
  o Development work and people performing it are partitioned into clean, low coupling partitions
  o Testing and documentation is performed as the product is built
  o Provides the ability to declare the product done whenever required
• Process patterns defining development activities
  o Backlog (prioritized list of requirements or features that provide business value to customer, items can be added at any time)
  o Sprints (work units required to achieve one of the backlog items, must fit into a predefined time-box, affected backlog items frozen)
  o 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?
  o Demos (deliver software increment to customer for evaluation)

Crystal

• Development approach that puts a premium on maneuverability during a resource-limited game of invention and communication with the primary goal of delivering useful software and a secondary goal of setting up for the next game

• Crystal principles
  o It’s always cheaper and faster to communicate face-to-face
  o As methodologies become more formal teams become weighed down and have trouble adapting to project work vagaries
  o As projects grow in size, teams become larger and methodologies become heavier
  o As projects grow in criticality some degree of formality will need to be introduced in parts of the methodology
  o As feedback and communication become more efficient the need for intermediate work products is reduced
  o Discipline, skills, and understanding counter process, formality, and documentation
  o Team members not on the critical project path can spend their excess time improving the product or helping people who are on the critical path

• Incremental development strategy used with 1 to 3 month time lines
• Reflection workshops conducted before project begins, during increment development activity, and after increment is delivered

• Crystal methodologies
  o Clear (small, low criticality projects)
  o Orange (larger, moderately critical projects)
  o Orange Web (typical e-business applications)

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
  o Emphasizes collaboration among team members
  o Manages problem and project complexity using feature-based decomposition followed by integration of software increments
  o Technical communication using verbal, graphical, and textual means
  o Software quality encouraged by using incremental development, design and code inspections, SQA audits, metric collection, and use of patterns (analysis, design, construction)

• Framework activities
  o Develop overall model (contains set of classes depicting business model of application to be built)
  o Build features list (features extracted from domain model, features are categorized and prioritized, work is broken up into two week chunks)
  o Plan by feature (features assessed based on priority, effort, technical issues, schedule dependencies)
  o 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)
  o Build by feature (class owner translates design into source code and performs
Agile Modeling

- Practice-based methodology for effective modeling and documentation of software systems in a light-weight manner
- Modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light (only keep models with long-term value)
  - Content is more important than representation
  - Know the models and tools you use to create them
  - Adapt locally
- Requirements gathering and analysis modeling
  - Work collaboratively to find out what customer wants to do
  - Once requirements model is built collaborative analysis modeling continues with the customer
- Architectural modeling
  - Derives preliminary architecture from analysis model
  - Architectural model must be realistic for the environment and must be understandable by developers