Agile Development

The Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come 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**

That is, while there is value in the items on the right, we value the items on the left more."

What is “Agility”?

- Effective (rapid and adaptive) response to change (team members, new technology, requirements)
- Effective communication in structure and attitudes among all team members, technological and business people, software engineers and managers。
- Drawing the customer into the team. Eliminate “us and them” attitude. Planning in an uncertain world has its limits and plan must be flexible.
- Organizing a team so that it is in control of the work performed
- Eliminate all but the most essential work products and keep them lean.
- Emphasize an incremental delivery strategy as opposed to intermediate products that gets working software to the customer as rapidly as feasible.

What is “Agility”?

- Rapid, incremental delivery of software
- The development guidelines stress delivery over analysis and design although these activates are not discouraged, and active and continuous communication between developers and customers.

Why and What Steps are“Agility” important?

- Why? The modern business environment is fast-paced and ever-changing. It represents a reasonable alternative to conventional software engineering for certain classes of software projects. It has been demonstrated to deliver successful systems quickly.

- What? May be termed as “software engineering lite” The basic activities-communication, planning, modeling, construction and deployment remain. But they morph into a minimal task set that push the team toward construction and delivery sooner.

- The only really important work product is an operational “software increment” that is delivered.

An Agile Process

- Is driven by customer descriptions of what is required (scenarios). Some assumptions:
  - Recognizes that plans are short-lived (some requirements will persist, some will change. Customer priorities will change)
  - Develops software iteratively with a heavy emphasis on construction activities (design and construction are interleaved, hard to say how much design is necessary before construction. Design models are proven as they are created.)
  - Analysis, design, construction and testing are not predictable.

- Thus has to Adapt as changes occur due to unpredictability
- Delivers multiple 'software increments', deliver an operational prototype or portion of an OS to collect customer feedback for adaption.

Agility Principles - I

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.

Agility Principles - II

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self–organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

An Agile Process

- Mainly 3 key assumptions:
  1. It is difficult to predict in advance which software requirements will persist and which will change…as well as how customer priorities will change as project proceeds.
  2. Design and construction are interleaved for many software. So both the activities should be performed in tandem so that design models are proven as they created.
  3. Analysis, design, construction and testing are not as predictable .

So the Question is..

How we create a process that can manage unpredictability?

Answer is:

Process Adaptability.

Human Factors

- *Agile Development focuses on talent and skills of individuals, molding the process to specific people and team,* not the other way around
- key traits must exist among the people on an agile team and the team itself:
  - **Competence**. talent, specific software related skills, overall knowledge of process that the team has chosen to apply.
    - Skill and knowledge of process can and should be taught to all people who serve as agile team members.
  - **Common focus.** All team members should be focused on one goal—to deliver a working software increment to the customer within the time promised though they work on different phases.

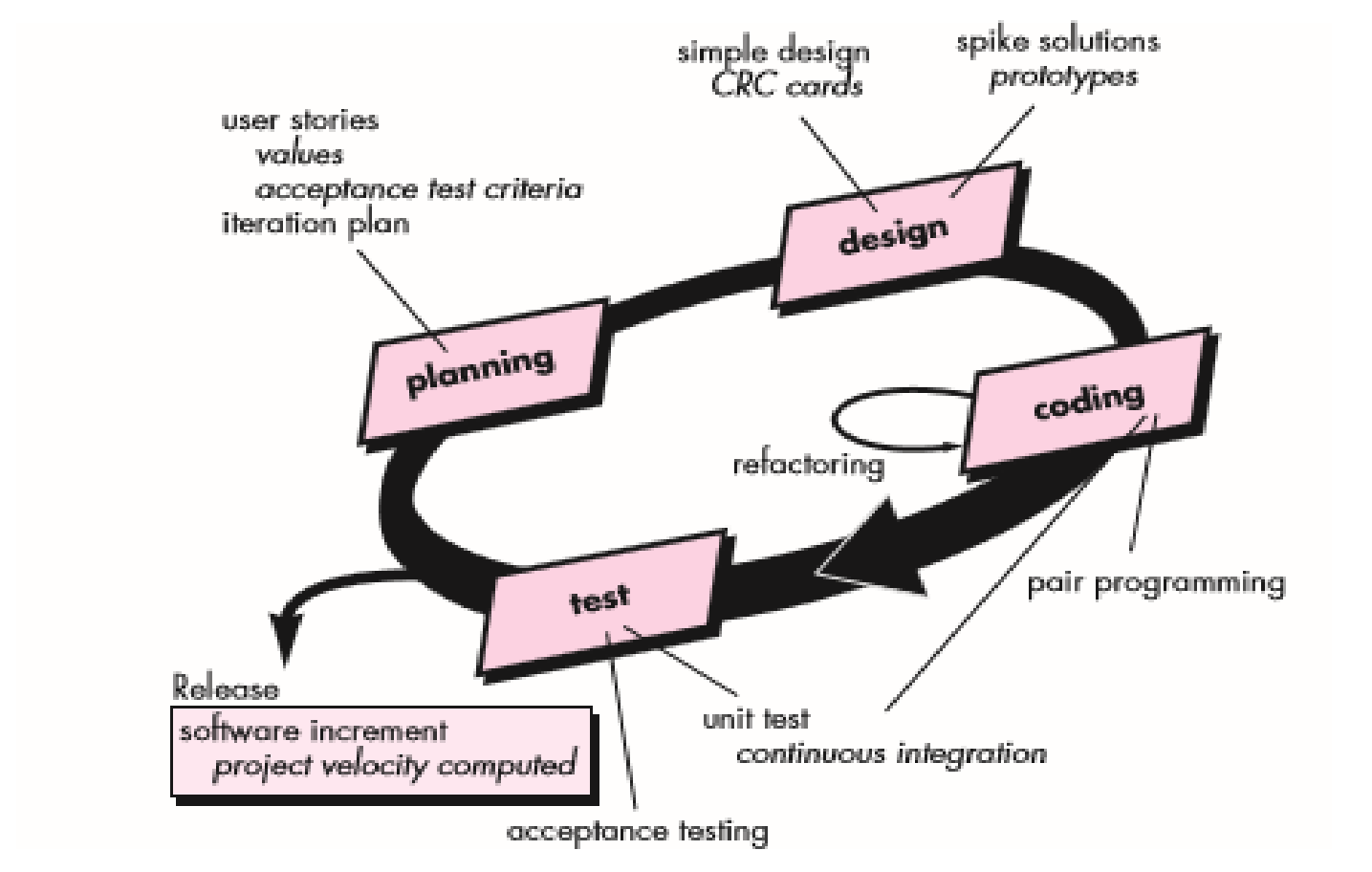
Human Factors

- **Collaboration**. Software engineering is about assessing, analyzing, and using information that is communicated to the software team; creating information that will help all stakeholders understand the work of the team; building information (computer software and relevant databases) that provides business value for the customer
  - **To accomplish these tasks, team members must collaborate with one another and all other stakeholders.**
- **Decision-making ability.** freedom to control its own destiny
- **Fuzzy problem-solving ability.** ambiguity and constant changes, today problem may not be tomorrow's problem
- **Mutual trust and respect.**
- **Self-organization.**

1. Organize itself for the work to be done,
2. Organize the process to best accommodate its local environment,
3. Organize the work schedule to best achieve delivery of s/w increment.

Agile Process Models

Extreme Programming (XP)



Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck in 2004. It uses an object-oriented approach.

- **XP Planning**
  - Begins with the listening, leads to creation of "user stories" that describes required output, features, and functionality. Customer assigns a value(i.e., a priority) to each story.
  - Agile team assesses each story and assigns a cost (development weeks. If more than 3 weeks, customer asked to split into smaller stories)
  - Working together, stories are grouped for a deliverable increment next release.
  - A commitment (stories to be included, delivery date and other project matters) is made. Three ways: 1. Either all stories will be implemented in a few weeks,
    2. high priority stories first, or
    3. the riskiest stories will be implemented first.
  - After the first increment "project velocity", namely number of stories implemented during the first release is used to help define subsequent delivery dates for other increments. Customers can add stories, delete existing stories, change values of an existing story, split stories as development work proceeds.

Extreme Programming (XP)

- **XP Design (occurs both before and after coding as refactoring is encouraged)**
  - Follows the Keep it simple principle Nothing more nothing less than the story.
  - Encourage the use of CRC (class-responsibility-collaborator) cards in an object-oriented context. The only design work product of XP. They identify and organize the classes that are relevant to the current software increment.
  - For difficult design problems, suggests the creation of "**spike solutions**"—a design prototype for that portion is implemented and evaluated.
  - Encourages "**refactoring**"—an iterative refinement of the internal program design. Does not alter the external behavior yet improve the internal structure. Minimize chances of bugs. More efficient, easy to read.

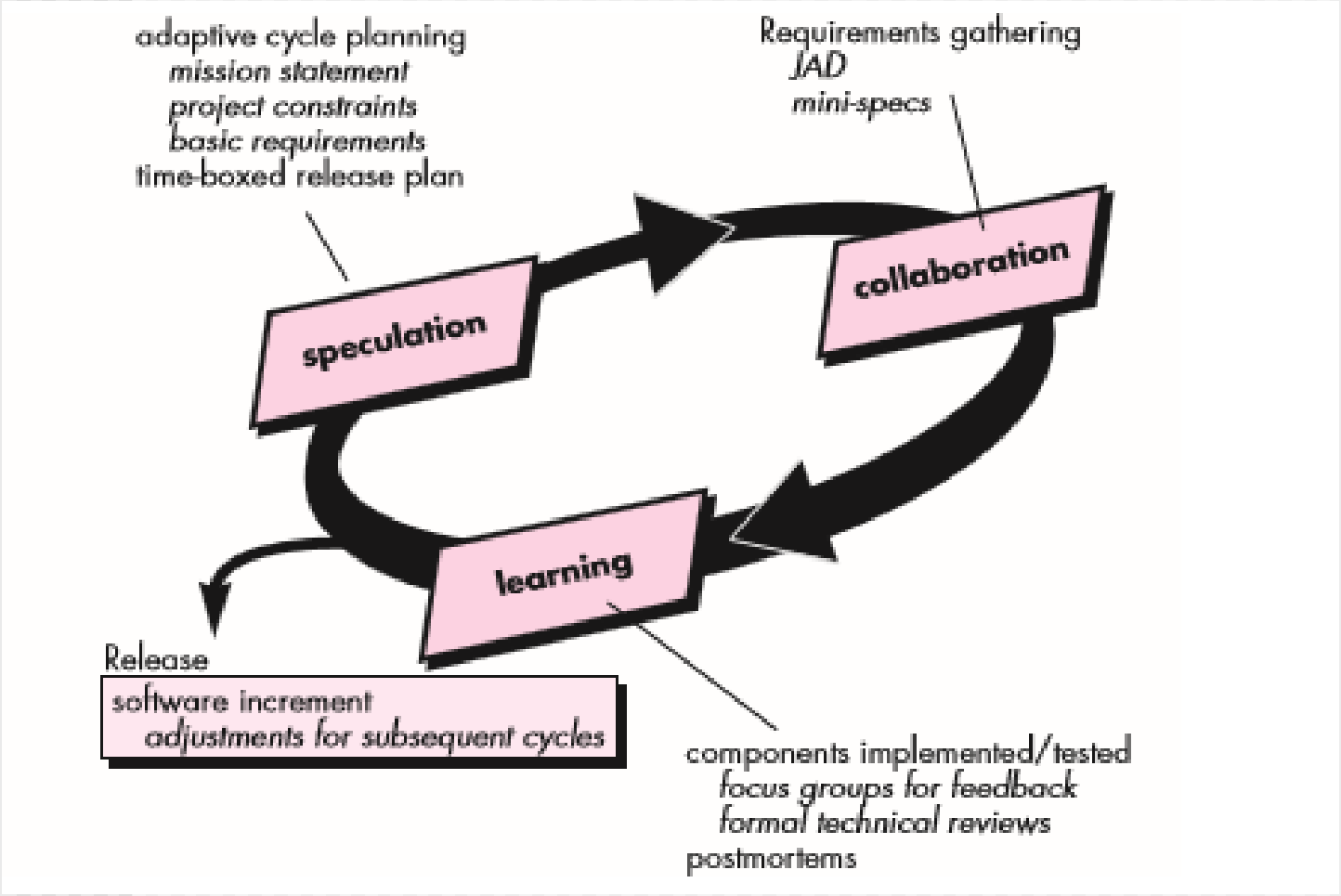
Extreme Programming (XP)

- **XP Coding**
  - Recommends the construction of a unit test for a story *before* coding commences. So implementer can focus on what must be implemented to pass the test.
  - Encourages "pair programming". Two people work together at one workstation. Real time problem solving, real time review for quality assurance. Take slightly different roles.

- **XP Testing**
  - All unit tests are executed daily and ideally should be automated. Regression tests are conducted to test current and previous components.

  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

Adaptive Software Development



Three Phases of ASD

- 1. Speculation: project is initiated and adaptive cycle planning is conducted.
- Adaptive cycle planning uses project initiation information-the customer's mission statement, project constraints (e.g. delivery date), and basic requirements to define the set of release cycles (increments) that will be required for the project.
- Based on the information obtained at the completion of the first cycle, the plan is reviewed and adjusted so that planned work better fits the reality.

Three Phases of ASD

- **2. Collaborations:** People worked together to multiply their talent and creative output beyond absolute number.
- It encompasses communication and teamwork, but it also emphasizes individualism, because individual creativity plays an important role in collaborative thinking.
- It is a matter of trust. 1) criticize without animosity, 2) assist without resentments, 3) work as hard as or harder than they do. 4) have the skill set to contribute to the work at hand, 5) communicate problems or concerns in a way that leas to effective action.

Three Phases of ASD

3. Learning: As members of ASD team begin to develop the components, the emphasis is on "learning".

- High smith argues that software developers often overestimate their own understanding of the technology, the process, and the project and that learning will help them to improve their level of real understanding.
- ASD team learns in 3 ways

1. **focus groups**: Feed back from customers and/or end users
2. **technical reviews :** Team members review s/w components
3. **project postmortems:** ASD team becomes introspective, addressing its own performance and process.

Dynamic Systems Development Method

- It is an agile software development approach that provides a framework for "building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment."
- Works on the principle that "80% of application can be delivered in 20% of time it would take to deliver the complete 100% application
- DSDM—distinguishing features
  - Similar in most respects to XP and/or ASD
  - Also suggest an iterative software process.

DSDM Life Cycle

- **Feasibility study:** establishes the basic business requirements and constraints associated with the application to be built and then assesses whether the application is a viable candidate for the DSDM process.

- **Business study: Es**tablishes the functional and information requirements that will allow the application to provide business value; also, defines the basic application architecture and identifies the maintainability requirements for the application.

- **Functional model iteration: P**roduces a set of incremental prototypes that demonstrate functionality for the customer.
- gather additional requirements by eliciting feedback from users as they exercise the prototype.

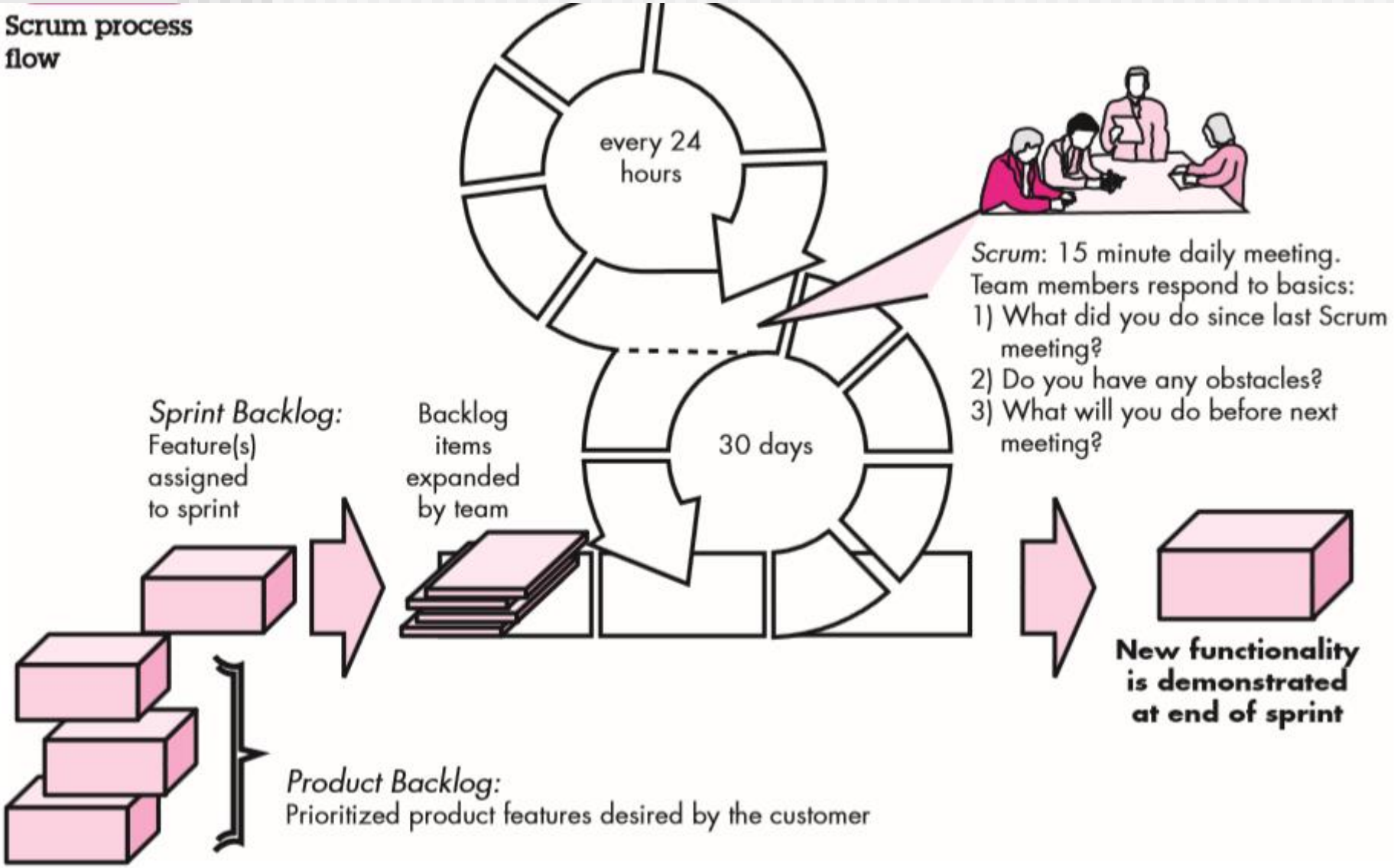
- **Design and build iteration: R**evisits prototypes built during functional model iteration to ensure that each has been engineered in a manner that will enable it to provide operational business value for end users.

- **Implementation:** Places the latest software increment into the operational environment.
  - It should be noted that (1) the increment may not be 100 percent complete or (2) changes may be requested as the increment is put into place.

Scrum

- A software development method Originally proposed by Schwaber and Beedle (an activity occurs during a rugby match) in early 1990.
- Scrum—distinguishing features
  - Development work is partitioned into "packets"
  - Testing and documentation are on-going as the product is constructed
  - Work units occurs in "sprints" and is derived from a "backlog" of existing changing prioritized requirements
  - Changes are not introduced in sprints (short term but stable) but in backlog.
  - Meetings are very short (15 minutes daily) and sometimes conducted without chairs (what did you do since last meeting? What obstacles are you encountering? What do you plan to accomplish by next meeting?)
  - "demos" are delivered to the customer with the time-box allocated. May not contain all functionalities. So customers can evaluate and give feedbacks.

Scrum


Scrum process flow
every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

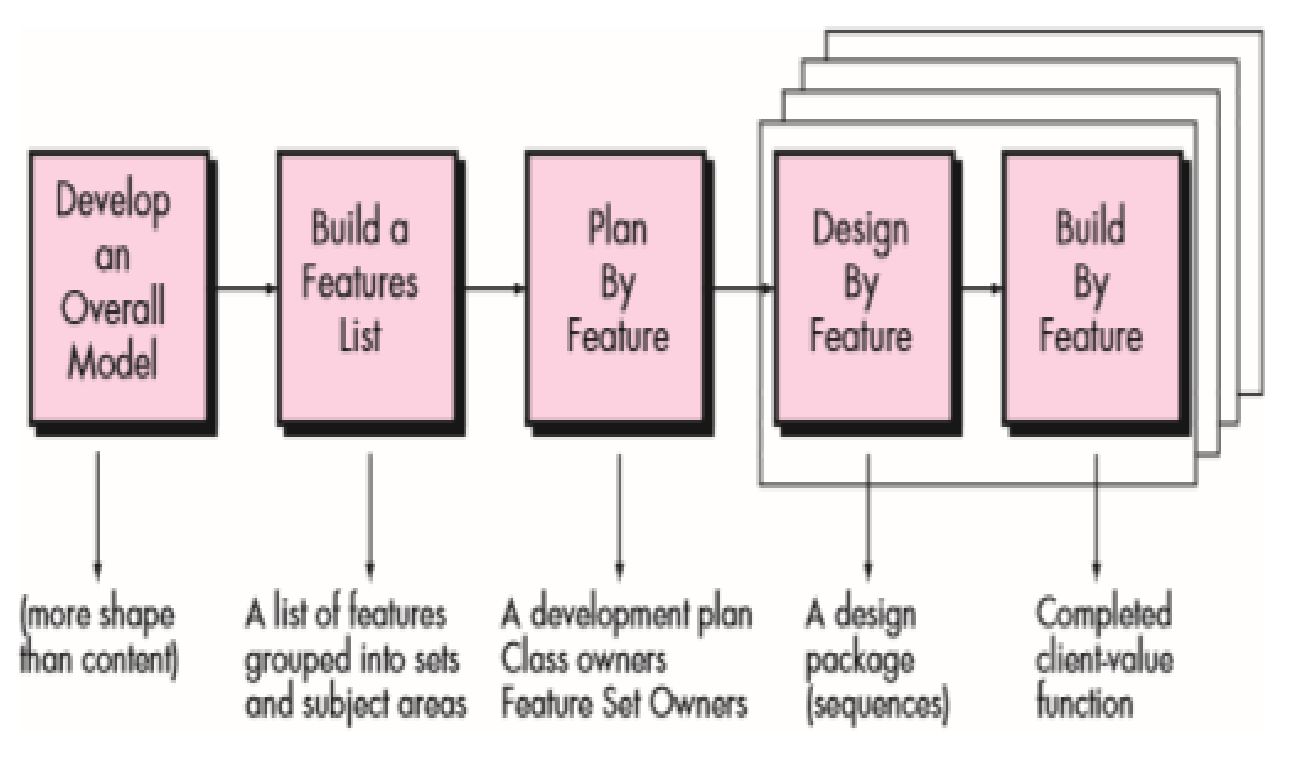
Crystal

- Proposed by Cockburn and Highsmith
- Crystal—distinguishing features
  - Actually a family of process models that allow "maneuverability" based on problem characteristics
  - Defined a set of methodologies, each with core elements that are common to all, and roles, process patterns, work products, and practice that are unique to each.
  - Face-to-face communication is emphasized
  - Suggests the use of "reflection workshops" to review the work habits of the team

Feature Driven Development

- Originally proposed by Peter Coad et al as a object-oriented software engineering process model.
- FDD—distinguishing features
  - Emphasis is on defining "features" which can be organized hierarchically.
    - a *feature* "is a client-valued function that can be implemented in two weeks or less."
  - Uses a feature template
    - <action> the <result> <by | for | of | to> a(n) <object>
    - E.g. Add the product to shopping cart.
    - Display the technical-specifications of the product.
    - Store the shipping-information for the customer.
  - A features list is created and "plan by feature" is conducted
  - Design and construction merge in FDD

Feature Driven Development



Agile Modeling

- Originally proposed by Scott Ambler
- Suggests a set of agile modeling principles
  - Model with a purpose: A developer who uses AM should have a specific goal (e.g., to communicate information to the customer or to help better understand some aspect of the software) in mind before creating the model.
  - Use multiple models: There are many different models and notations that can be used to describe software.
  - Travel light: As software engineering work proceeds, keep only those models that will provide long-term value and jettison the rest.

- Content is more important than representation: Modeling should impart information to its intended audience.
- Know the models and the tools you use to create them: Understand the strengths and weaknesses of each model and the tools that are used to create it.
- Adapt locally: The modeling approach should be adapted to the needs of the agile team.