Software Development Methodologies

Lecturer: **Raman Ramsin**

Lecture 8

Agile Methodologies: XP

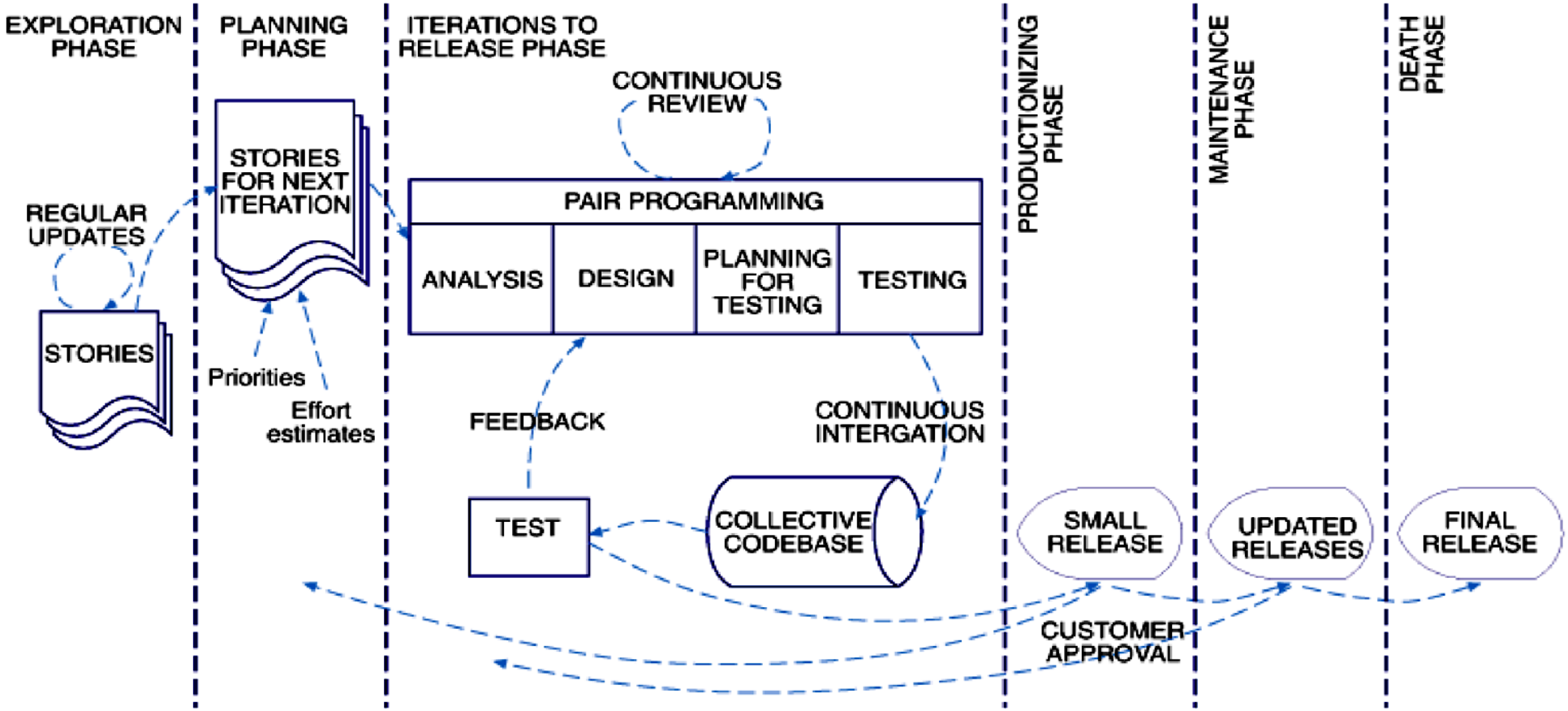
eXtreme Programming (XP)

- Developed by Beck in 1996.
- The first authentic XP book appeared in 1999, with a revised and refined version appearing in 2004.
- Although some of the methodologies that are nowadays dubbed as agile are older than XP, it was the advent of XP that sparked the agile movement.
- XP considers itself a software engineering *discipline* rather than a *methodology*, yet it does incorporate a process.

XP: Process

1. *Exploration:*
    1. developing an initial list of high-level requirements
    2. determining the overall design of the system through prototyping
2. *Planning (Release Planning):*
    1. estimating the time needed for the implementation of each requirement
    2. prioritizing the requirements
    3. determining a schedule and a minimal, select set of requirements for the first release of the system
3. *Iterations to First Release:* iterative development of the first release of the system using the specific rules and practices prescribed by XP; iterations are typically between 1 to 3 weeks in duration.
4. *Productionizing:* system-wide verification and validation of the first release, and its deployment into the user production environment
5. *Maintenance:* implementing the remaining requirements (including any resulting from post-deployment maintenance needs) into the running system by iterating phases 2, 3 and 4.
6. *Death:* closing the project and conducting post-mortem review and documentation

XP: Process

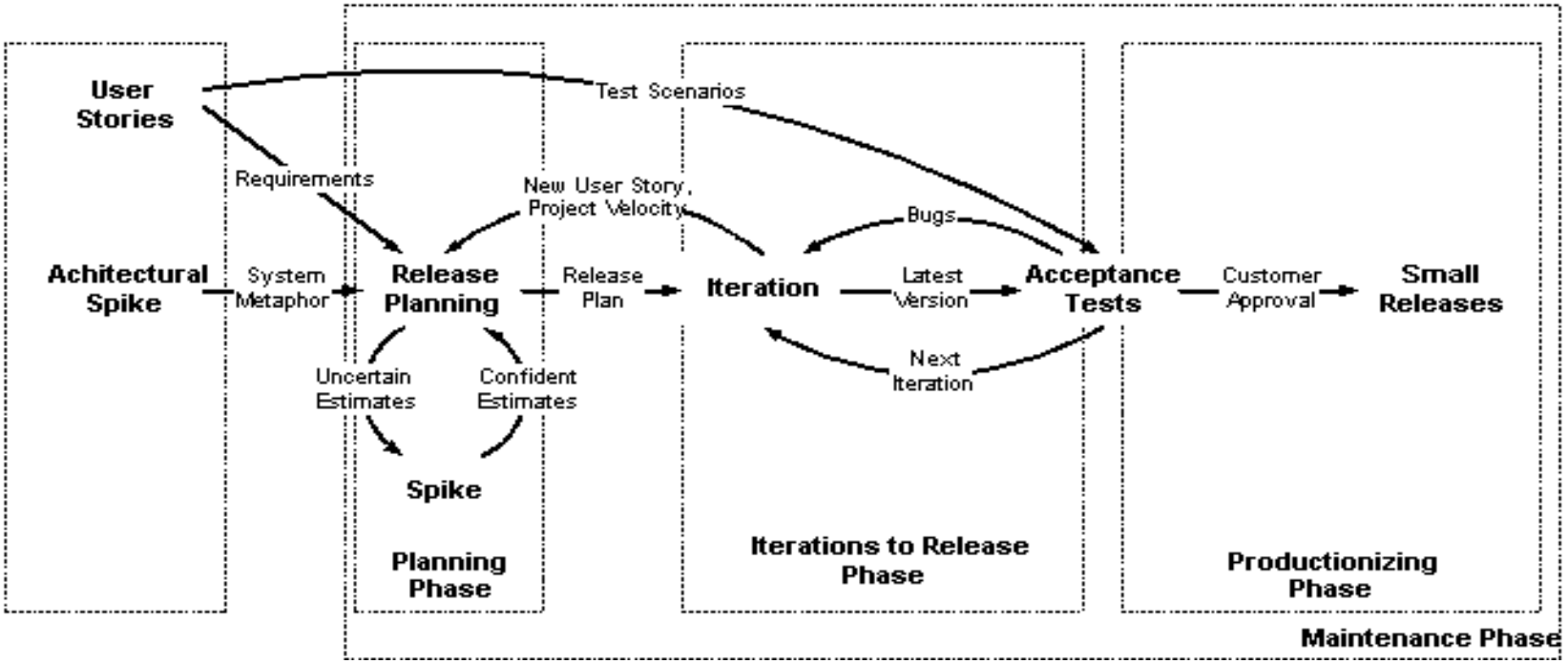


[Abrahamsson et al. 2002]

XP: Process – Development Engine

- The activities performed in the second, third and fourth phases of the process constitute the development "engine" of the XP methodology.
- Each execution (run) of these phases produces a new release.
- A first release of the system is initially produced and deployed, which is then incrementally improved and complemented during the *maintenance* phase through further iterations (runs) of the development engine.

XP: Process

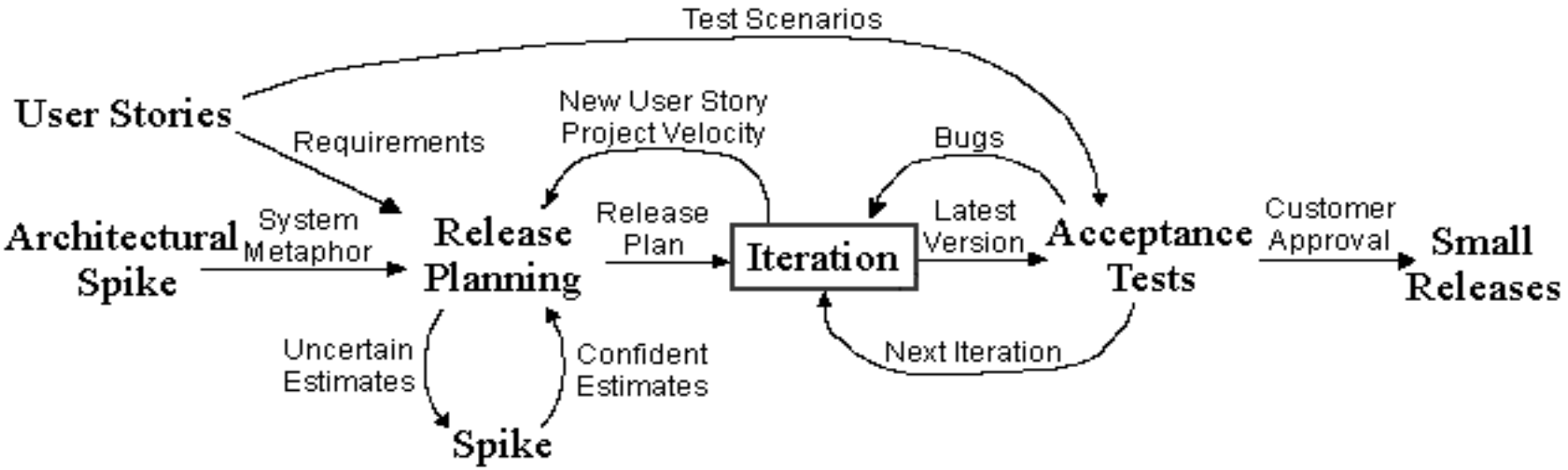


[Ambler 2002]

XP Process: Exploration

1. *Formation of the development team:* the team typically consists of
   - A *coach* acting as monitor and facilitator, a number of *programmers,* and a business representative (*customer*) participating in project activities and supplying the team with information and feedback
   - [Optional] A number of *analysts* to help elicit the requirements, a number of *testers* helping define acceptance tests, and a *resource manager*
2. *Development of the initial set of User Stories:* A *User Story*
   - defines a system feature as seen from the customer's point of view.
   - is written by the customer in his own terminology on index cards.
   - Is nothing but a short description (around three sentences) of a certain chunk of functionality needed to be delivered by the system.
3. *Creation of the system Metaphor:*
   - A prototype (called *Spike* or *Spike Solution*) is developed, exploring potential architectures for the system.
   - The prototype helps the team define the system *Metaphor*, which is typically a very simple, high-level description of how the system works.
   - It usually takes the form of a description-by-analogy in order to be easily understandable to all the team members.
   - Though informal, the metaphor gives an extremely useful idea of the overall architecture of the system without setting too many constraints.

XP Process: Development Engine – *Top-Level Activities*



[Wells 2003]

XP Process: Planning (*Release Planning*)

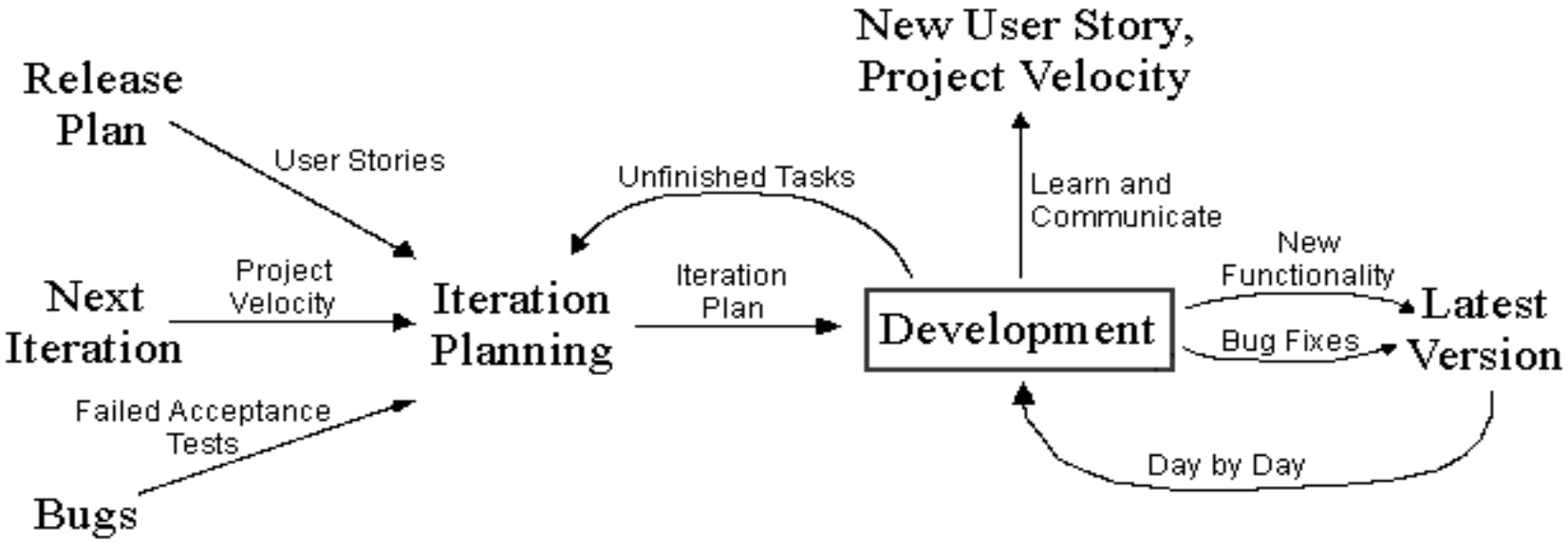
1. *Estimation of development time:*
    1. Developers estimate the time needed to develop each of the user stories as conforming to the system metaphor, and write the estimates down on the user-story index cards.
    2. User stories that need more than 3 weeks to develop are broken down into smaller ones, and user stories taking less than 1 week are merged.
    3. In cases where estimates are not reliable enough, spike solutions are developed in order to improve the estimates.
2. *Prioritization of user stories:* the customer prioritizes the user stories according to their business value.
3. *Planning the first release:*
    1. the team selects a minimal, most valuable set of user stories for implementation in the first release, and agrees on the release date.
    2. The team also decides on the iteration duration (between 1 to 3 weeks), which once determined, will be the same for all iterations.

XP Process: Iterations to First Release

1. *Iteration planning*: at the start of each iteration, a planning meeting is held, performing:
   1. Selection of user stories to implement, and failed acceptance tests to rectify
   2. Identification of programming tasks
   3. Task sign-up and estimation

2. *Development*: the development activity in each iteration is itself an iterative process with daily cycles consisting of the following activities:
   1. Holding daily stand up meetings
   2. Analysis, design, coding, testing and integration in a Collective-Code-Ownership environment

XP Process: Development Engine – *Iteration* Activities



[Wells 2003]

XP Process: Iterations to First Release – *Iteration Planning*

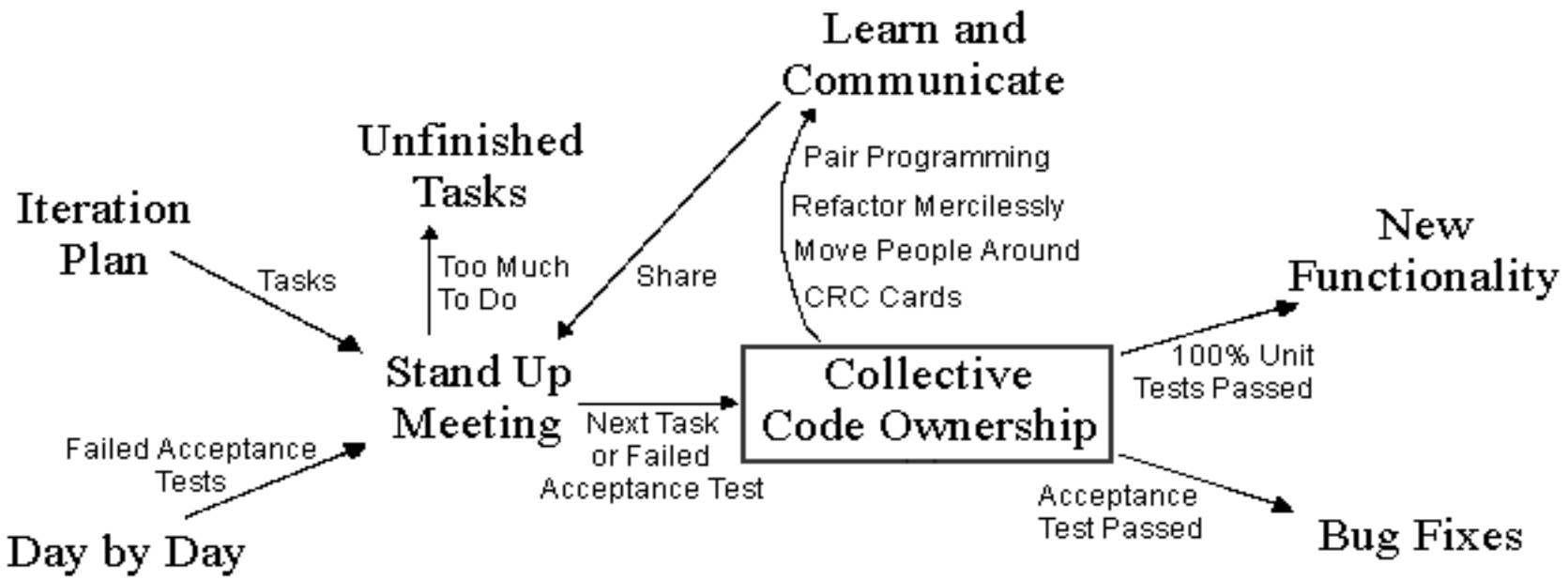
1. *Selection of user stories to implement and failed acceptance tests to rectify:*
   - based on the release plan, the customer selects user stories (according to their business value) for development in the coming iteration.
   - Failed acceptance tests encountered during previous iterations are also considered for inclusion in the list of jobs to be attended to.
   - Special attention is given to the experience gained during previous iterations as to the team's development speed (*Project Velocity*), to ensure that the selected jobs can be completed in the iteration.
2. *Identification of programming tasks:*
   - the developers on the team break down the selected user stories and debugging jobs into programming tasks.
   - Tasks are then written down on the user-story index cards.
3. *Task sign-up and estimation:*
   - programmers sign-up to do the tasks.
   - Each developer estimates the time needed for completion of each of the tasks undertaken, making sure that all of them can be developed in the time available. Each task should take between 1 to 3 days to complete.

XP Process: Iterations to First Release - *Development*

1. *Holding daily stand up meetings:*
   - A short stand up meeting is held every morning in order to communicate problems and solutions, and help the team keep on track.

2. *Analysis, design, coding, testing and integration in a Collective-Code-Ownership environment:*
   - *Collective Code Ownership* means that all the code developed is put in a shared code repository, and any developer can change his or others' code in order to add functionality, fix bugs, or refactor.
   - test-driven development to make collective code ownership possible:
     - the developers have to create unit tests for their code as they develop it.
     - All the code in the code repository includes unit tests, forming a suite of tests that is automatically applied by test tools whenever code is added or changed.
     - The test suite safeguards the repository from malignant change: for code to be allowed integration into the repository, it must pass the entire test suite.
     - Black-box acceptance tests based on the user stories are defined by the customer and developed by the team during the iteration, and are frequently applied (by automated tools) to the code.

XP Process: Development Engine – Iteration *Development* Activities



[Wells 2003]

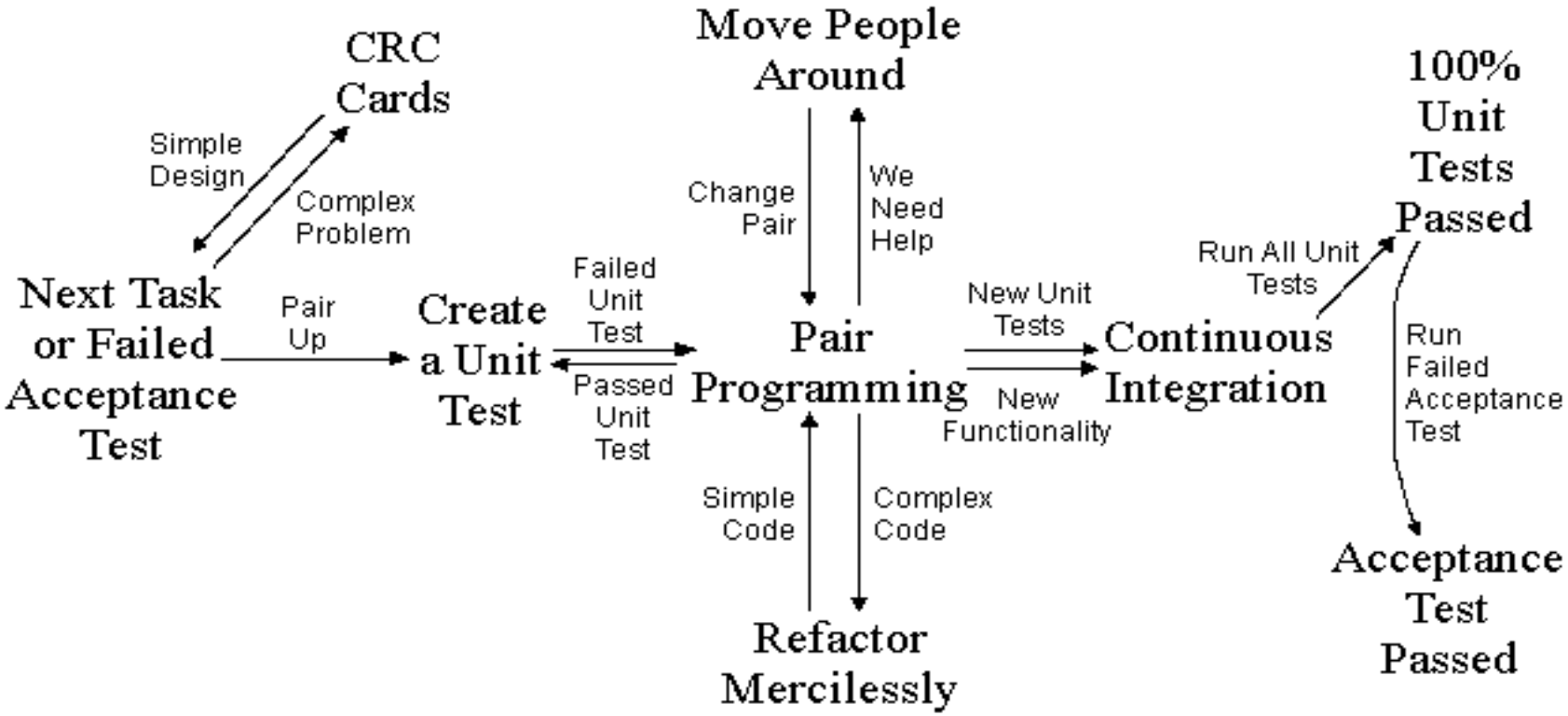
XP Process: Development Engine – Iteration *Development* Activities

Activities in the Collective-Code-Ownership Environment

- Programmers work in pairs, each pair on one machine (a practice called *Pair Programming*).
- Programmers use CRC cards in order to come up with the simplest design possible for the programming task in hand.
- Refactoring is constantly done in order to simplify the code and eliminate redundancy.
- A common coding standard is enforced in order to promote code legibility, which in turn enhances communication among developers.
- Developers are moved around so that they acquire knowledge about all parts of the system; this will reduce the cost of changes made to the team structure and will help relieve overloading and coding bottlenecks.
- Builds are frequent in XP, and continuous integration is encouraged.
- Developers are to work at a sustainable pace, with forty hours a week as the norm; nobody is allowed to work overtime for two weeks in a row.

XP Process: Development Engine – Iteration *Development* Activities

Activities in the Collective-Code-Ownership Environment



[Wells 2003]

XP Process: Productionizing

1. *System-wide verification and validation:* The release is tested to make sure of the user's approval and the system's readiness for deployment.
   - Acceptance tests, mostly developed during the iterations-to-first-release phase, are used here as regression tests.
   - Defects found are resolved through iterations of the main development cycle.
2. *Deployment into the production environment:* the release is introduced into the user environment.
   - Involves the usual integration, conversion, tuning, training, and documentation activities typical of deployment efforts.
   - Any tuning and stabilization action on the release itself is regarded as a development activity and is conducted through short iterations (typically weekly) of the development cycle.

XP Process: Maintenance

- The maintenance phase is when the remaining user stories are implemented into the system and the system is maintained as such.
- Encompasses the same activities as those in the phases of the *development engine*; i.e. Planning, Iterations to Release, and Productionizing.
- Still dependent on the evolving set of user stories and the system metaphor.
- The small releases produced during maintenance are integrated into an already running and operational system.
- Requirements arising as a result of maintenance are treated as ordinary requirements (expressed as user stories) and implemented through the same iterative development process.
- The maintenance phase continues until either there are no more user-stories to develop and none are anticipated in the future, or the system in no way lends itself to necessary evolution any more.

XP Process: Death

- The project is declared dead when evolution is either unnecessary or impossible.

- The main activities performed in this phase of the XP process are:
  1. *Declaring the project as closed:* this involves wrapping up the usual legal, financial and social loose ends.
  2. *Post-mortem documentation and review:* preparing a short document (no longer than ten pages) providing a brief tour of the system, and writing a review report summarizing the lessons learned from the project.

XP: Strengths and Weaknesses

- ***Strengths***
  - Iterative-incremental process
  - Based on system functionality captured in *User Stories*
  - The process is tuned according to feedback during its execution
  - Traceability to requirements through the use of *user stories* throughout the process as the basis for tasks and tests
  - Based on system architecture (*Metaphor*) identified through prototyping
  - Active user involvement
  - Test-based development

XP: Strengths and Weaknesses

- ***Strengths*** **(Contd.)**
  - Stringent standards enforced on coding
  - Early and frequent releases
  - Requirements are allowed to evolve over time
  - Iterative development engine governed by careful planning and reviewing
  - Explicit coverage of maintenance and project retirement (*Death*) phases; maintenance in fact comprises the bulk of the development effort
  - Continuous validation
  - Continuous integration
  - Refactoring exercised in order to acquire the simplest code possible

XP: Strengths and Weaknesses

- ***Weaknesses***
  - Process is rather vague: the process commonly introduced as the "XP Process" is just a typical example.
  - More intended as a set of principles and practices rather than a methodology
  - Limited evidence of scalability
  - Seamlessness is not addressed: development is more or less a jump from *user stories* to code, and the little design that is done (if at all) does not have to conform to any standard.

XP: Strengths and Weaknesses

- ***Weaknesses*** **(Contd.)**
  - Requires the use of automated tools and enforcement of discipline for "*Collective-Code-Ownership*" to be practicable.
  - No clear-cut design effort
  - Model-phobic
  - Except for CRC cards, models are not prescribed, leaving it to the individual developer to decide what model is useful to him.
  - Lack of formalism

References

- Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., *Agile Software Development Methods: Review and Analysis*. VTT Publications, 2002.
- Wells, D., Extreme programming: A gentle introduction. Published on the Web at: *http://www.extremeprogramming.org*, 2013, Last visited in November 2017.
- Beck, K., and Andres, C., *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2004.
- Ambler, S. W., AM Throughout the XP Lifecycle, Published on the Web at: *http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm*, 2002, Last visited in November 2017.