COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 4

EXTREME PROGRAMMING (XP)

EXTREME PROGRAMMING

- Evolved from the problems caused by the long development cycles of traditional development models (Beck 1999a).
- First started as 'simply an opportunity to get the job done' (Haungs 2001) with practices that had been found effective in software development processes during the preceding decades (Beck 1999b)

- Method is formed around common sense principles and simple to understand practices
  - No process fits every project, rather, simple practices should be tailored to suit an individual project

XP VALUES

- **Communication:** XP has a culture of oral communication and its practices are designed to encourage interaction.

  "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important."

- **Simplicity:** Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.

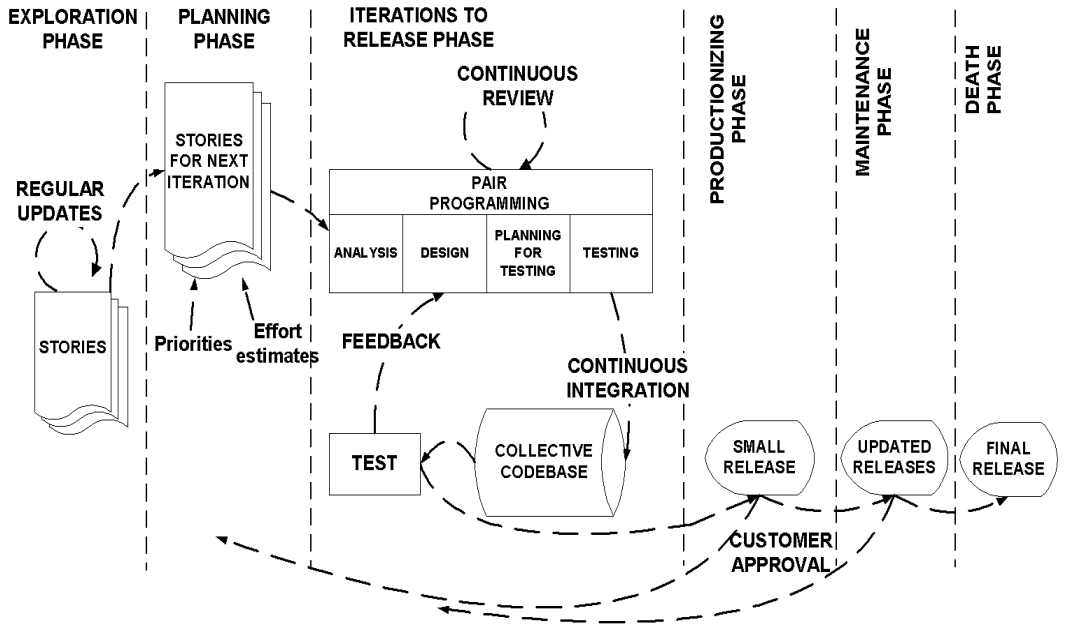
XP VALUES

- **Feedback:** The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration.
- **Courage**: Allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.
- **Respect:** Team members need to care about each other and about the project.

XP PROCESS

- The life cycle of XP consists of five phases:

1. Exploration
2. Planning
3. Iterations to Release
4. Productionizing
5. Maintenance and Death



XP PROCESS – EXPLORATION PHASE

- The customers write out the story cards that they wish to be included in the first release
- At the same time the project team familiarize themselves with the tools, technology and practices they will be using in the project
- The exploration phase takes between a few weeks to a few months, depending largely on how familiar the technology is to the programmers

XP PROCESS – PLANNING PHASE

- Users stories are written
- Estimate the effort of working with the user stories
- Priorities are given to the user stories to be implemented
- Release planning creates the release schedule

XP PROCESS – ITERATIONS TO RELEASE PHASE

- Includes several iterations of the systems before the first release
- Each take one to four weeks to implement
- The first iteration creates a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system
- The customer decides the stories to be selected for each iteration
- At the end of the last iteration the system is ready for production

XP PROCESS – PRODUCTIONIZING PHASE

- Requires extra testing and checking of the performance of the system before the system can be released to the customer
- New changes may still be found and the decision has to be made if they are included in the current release
- The iterations may need to be quickened from three weeks to one week
- The postponed ideas and suggestions are documented for later implementation

XP PROCESS – MAINTENANCE PHASE

- After the first release is productionized for customer use, the XP project must both keep the system in the production running while also producing new iterations
- Requires an effort also for customer support tasks
- Development velocity may decelerate after the system is in production
- May require incorporating new people into the team and changing the team structure

XP PROCESS – DEATH PHASE

- When the customer does no longer have any stories to be implemented
- System satisfies customer needs also in other respects (e.g., concerning performance and reliability)
- Necessary documentation of the system is finally written as no more changes to the architecture, design or code are made
- Death may also occur if the system is not delivering the desired outcomes, or if it becomes too expensive for further development

XP - ROLES AND RESPONSIBILITY

- **Customer:** writes the stories and functional tests, and decides when each requirement is satisfied. The customer sets the implementation priority for the requirements
- **Programmer:** keeps the program code as simple and definite as possible
- **Tester:** helps the customer write functional tests, also run functional tests regularly, broadcast test results and maintain testing tools

XP - ROLES AND RESPONSIBILITY

- **Tracker:** gives feedback in XP. He traces the estimates made by the team (e.g. effort estimates) and gives feedback on how accurate they are in order to improve future estimations. He also traces the progress of each iteration and evaluates whether the goal is reachable within the given resource and time constraints or if any changes are needed in the process
- **Coach:** Coach is the person responsible for the process as a whole. A sound understanding of XP is important in this role enabling the coach to guide the other team members in following the process
- **Consultant:** Consultant is an external member possessing the specific technical knowledge needed
- **Manager (Big Boss):** Manager makes the decisions

XP - PRACTICES

- **Interaction:** Close interaction between the customer and the programmers. The programmers estimate the effort needed for the implementation of customer stories and the customer then decides about the scope and timing of releases.
- **Small/short releases:** A simple system is "productionized“ rapidly – at least once in every 2 to 3 months. New versions are then released even daily, but at least monthly.
- **Metaphor:** The system is defined by a metaphor/set of metaphors between the customer and the programmers. This "shared story" guides all development by describing how the system works.

XP - PRACTICES

- **Simple design:** The emphasis is on designing the simplest possible solution that is implementable at the moment
- **Testing:** Software development is test driven. Unit tests are implemented continuously
- **Refactoring:** Restructuring the system by removing duplication, improving communication, simplifying and adding flexibility
- **Collective ownership:** Anyone can change any part of the code at any time

XP - PRACTICES

- **Pair programming**
  - Two people write the code at one computer
  - One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical defects (syntactic, spelling, etc.) and also thinks strategically about the direction of the work.
  - Two programmers can brainstorm any challenging problem. Because they periodically switch roles.

- **Continuous integration:** A new piece of code is integrated into the code-base as soon as it is ready.

XP - PRACTICES

- **40-hour week:** A maximum of 40-hour working week
- **On-site customer:** Customer has to be present and available full-time for the team
- **Coding standards:** Coding rules exist and are followed by the programmers (e.g. error message by exception handling). Communication through the code should be emphasized
- **Open workspace:** A large room with small cubicle (compartment) is preferred
- **Just rules:** Team has its own rules that are followed, but can also be changed at any time. The changes have to be agreed upon and their impact has to be assessed

XP - ARTEFACTS

- **User story cards**
  - Paper index cards which contain brief requirement (features, non-functional) descriptions
  - Not a full requirement statement
  - Commitment for further conversation between the developer and the customer
  - During this conversation, the two parties will come to an oral understanding of what is needed for the requirement to be fulfilled
  - Customer priority and developer resource estimate are added to the card
  - The resource estimate for a user story must not exceed the iteration duration

XP - ARTEFACTS

- **Task list**
  - A listing of the tasks (one-half to three days in duration) for the user stories that are to be completed for an iteration
  - Tasks represent concrete aspects of a user story
  - Programmers volunteer for tasks rather than are assigned to tasks
- **CRC (Class-Responsibility-Collaboration) cards** (optional)
  - Paper index card on which one records the responsibilities and collaborators of classes which can serve as a basis for software design
  - The classes, responsibilities, and collaborators are identified during a design brainstorming/role-playing session involving multiple developers

XP - ARTEFACTS

- **Customer Acceptance Tests**
  - Textual descriptions and automated test cases which are developed by the customer
  - The development team demonstrates the completion of a user story and the validation of customer requirements by passing these test cases.

- **Visible Wall Graphs**
  - To foster communication and accountability, progress graphs are usually posted in team work area. These progress graphs often involve how many stories are completed and/or how many acceptance test cases are passing.

XP - ARTEFACTS

Scope and use & Limitation

- XP is aimed at small to medium-sized teams
- Requires the development team to be located in one place. Scattering of programmers on two floors or even on one floor is intolerable for XP
- Communication and coordination between project members should be enabled at all times
- Require experienced team member through the project development time (XP is people based rather than plan based)

HOW XP SOLVES SOME SE PROBLEMS

Problem	Solution
Slipped schedule	Short development cycles
Cost of changes	Extensive, ongoing testing, system always running
Defect rates	Unit tests, customer tests
Misunderstand the business	Customer is a part of the team
Business changes	Changes are welcome
Staff turnover (replacement)	Intensive teamwork

REFERENCES

- R.S. Pressman & Associates, Inc. (2010). *Software Engineering: A Practitioner's Approach.*
- Kelly, J. C., Sherif, J. S., & Hops, J. (1992). An analysis of defect densities found during software inspections. *Journal of Systems and Software*, *17*(2), 111-117.
- Bhandari, I., Halliday, M. J., Chaar, J., Chillarege, R., Jones, K., Atkinson, J. S., & Yonezawa, M. (1994). In-process improvement through defect data interpretation. *IBM Systems Journal*, *33*(1), 182-214.