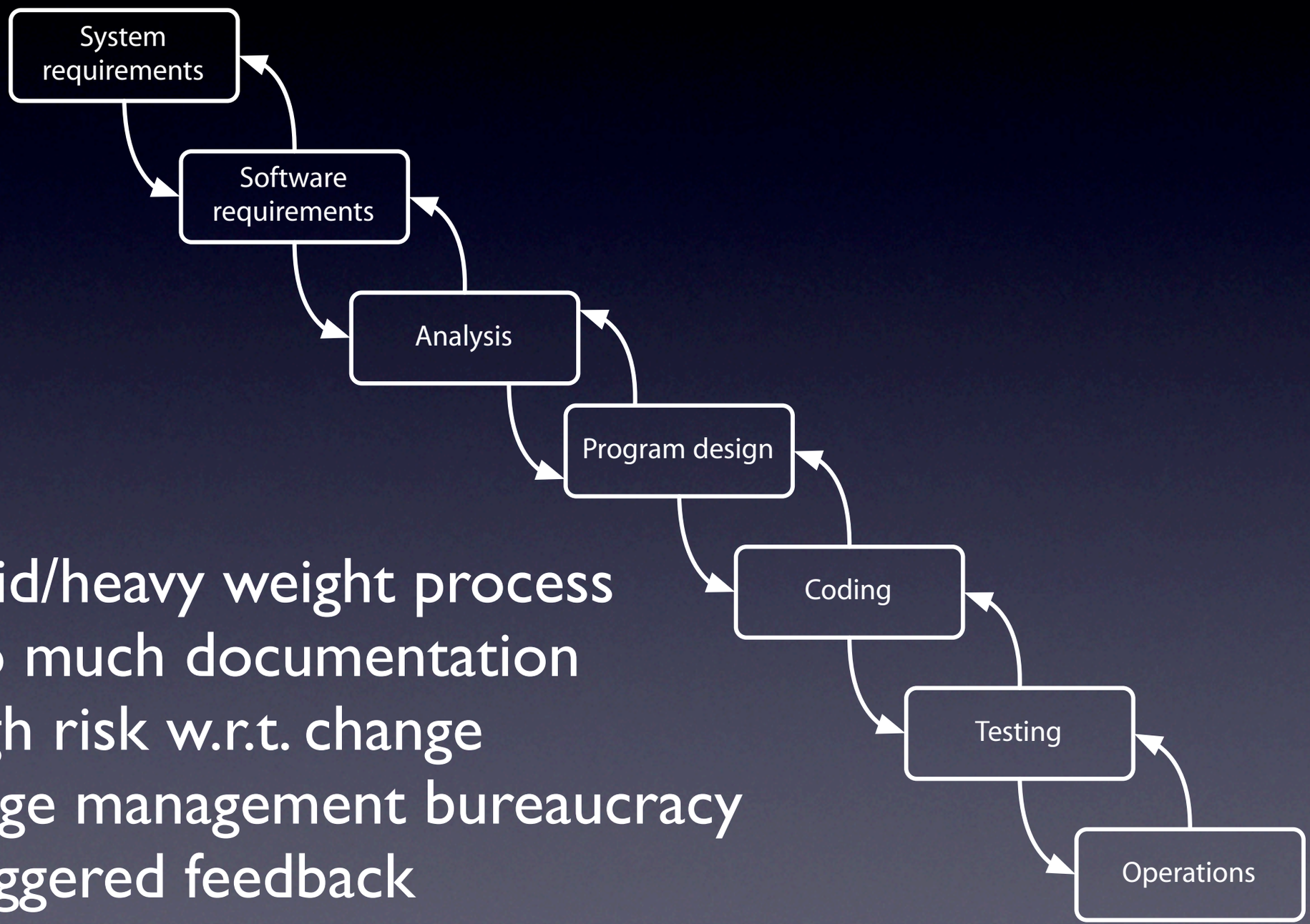
CS 5704: Software Engineering

Agile Methodologies

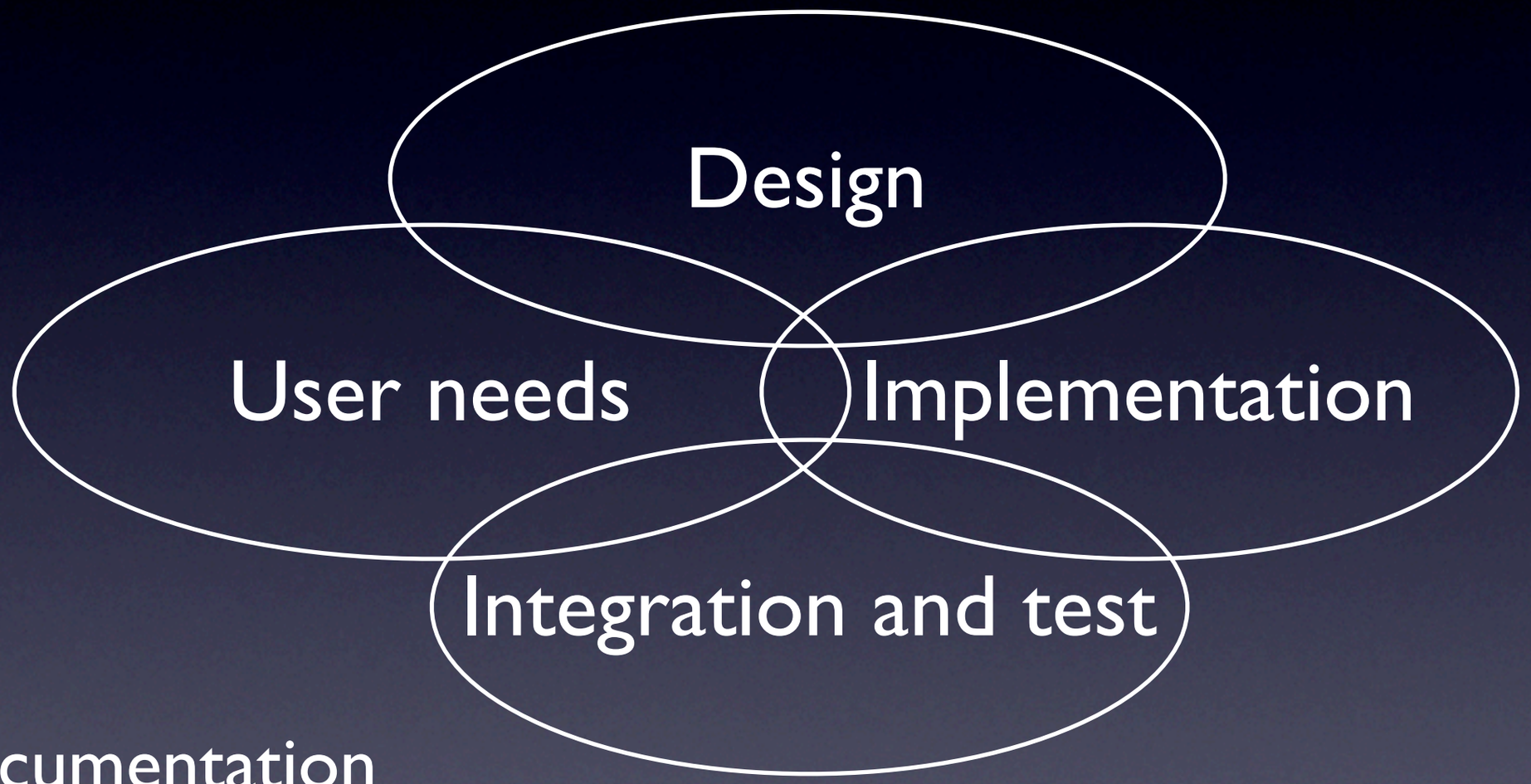
Dr. Pardha S. Pyla

What is wrong with this?



1. Rigid/heavy weight process
2. Too much documentation
3. High risk w.r.t. change
4. Large management bureaucracy
5. Staggered feedback

Turning the controls to other extreme



1. Lean process
2. Almost no documentation
3. Embrace change, minimize risk
4. No management bureaucracy
5. Implicit feedback

* No control over process?
* Change everywhere?

Controls in Agile Methods

- Test driven development
- Short, periodic time increments
  - Deliver working software that adds business value
- Small scale estimation, pair programming, etc.
- "*What would you do if your customer could afford only one day of development?*"

The Agile Manifesto

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*

Agile principles - I

1. Satisfy customer through early and frequent delivery
2. Welcome changing requirements, even late
3. Deliver working software over short cycles
4. Business and software roles work together, daily
5. Build projects around motivated people, provide support
6. Advocate face-to-face communication among team members

Agile principles -2

7. Working software is the primary measure of progress
8. Maintain sustained pace of development
9. Continuos attention to technical excellence and design
10. Simplicity is essential
11. The best products come from self-organizing teams
12. Periodically, teams reflect, assess, and improve

Agile emphasis

- From up-front analysis driving cost and time
  - to cost and time driving what can be built
- From trying to avoid requirements change
  - to embracing changing requirements

1) Adaptive Software Development (ASD)

- Three phases:
  - *Speculation*: project overview, constraints, basic requirements, time-boxed release plan
  - *Collaboration*: requirements gathering, mini-specs
  - *Learning*: components implemented, tested (focus groups, formal tech reviews, postmortems)

2) Dynamic Systems Development Method

- 80% rule: enough work to move to next increment
- Four phases:
  - *Feasibility study*: candidate for DSDM?
  - *Business study*: id's requirements to add business value, define basic application architecture
  - *Functional model iteration*: evolutionary prototypes to elicit feedback and more details
  - *Design and build iteration,* and *Implementation* (deployment)

3) Feature Driven Development (FDD)

- Feature: *a client valued function that can be implemented in two weeks or less*
- Templates for features and feature sets:

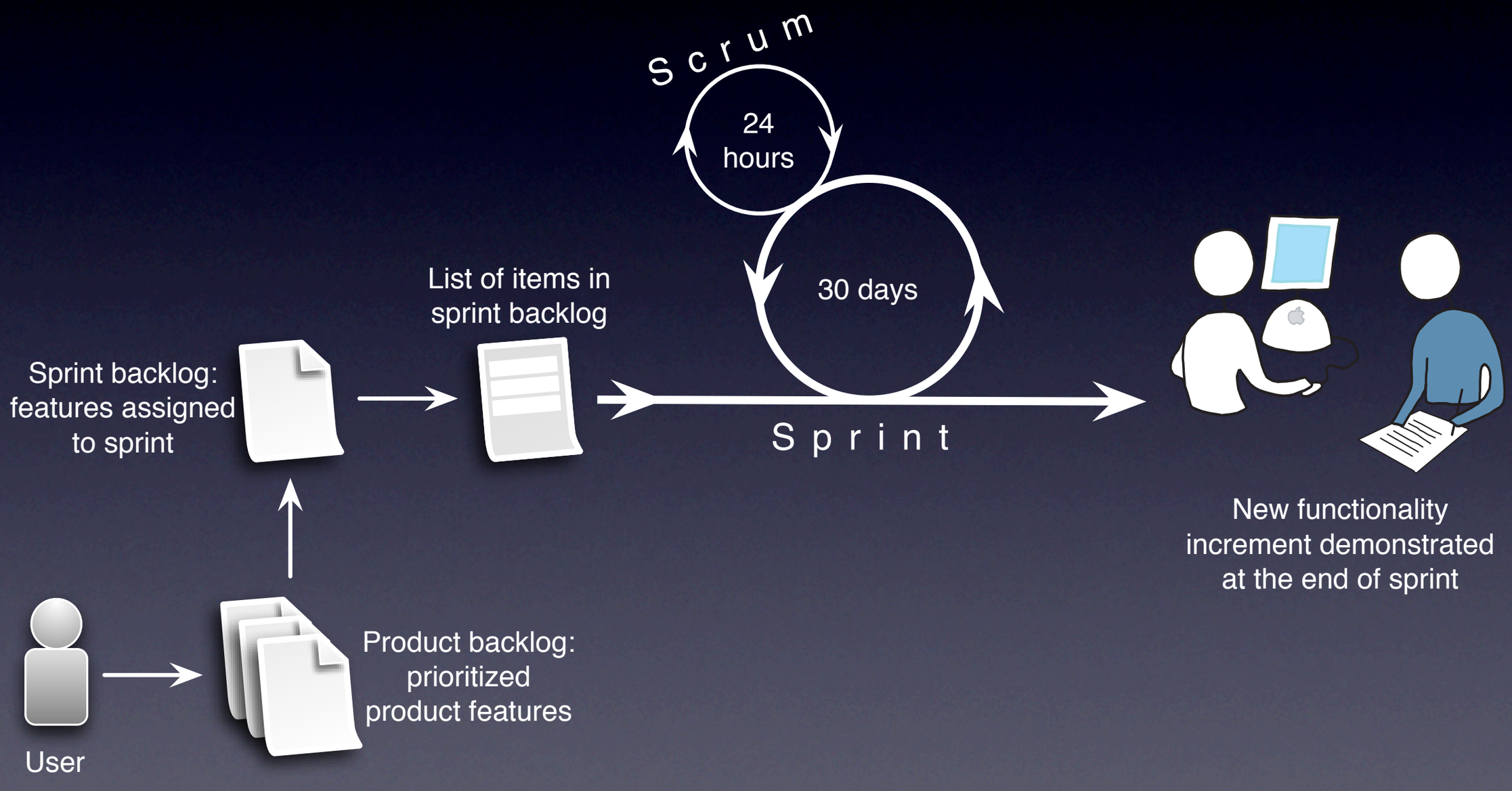
<*action*> the <*result*> <*by|for|to*> a(n) <*object*>

e.g. *Add the product to a cart*

<*action*><*ing*> a(n) <*object*>

e.g. *Making a product sale*

4) Scrum


Scrum
24 hours
30 days
List of items in sprint backlog
Sprint backlog: features assigned to sprint
Sprint
New functionality increment demonstrated at the end of sprint
User
Product backlog: prioritized product features

Scrum product backlog

Priority	Feature #	Description	Cost	By
Very high	1	Admin: add user	12	PS
High	2	Authentication via LDAP	20	TK
High	3	Admin: remote login	16	PS
...	...	.....	...	...

5) Extreme Programming

- The four values of XP
  - Communication
  - Simplicity
  - Feedback
  - Courage
- *"A deeper value that lies below the surface of the other four is respect"*

XP basic principles

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

“Less central principles”

- Teach change
- Small initial investment
- Play to win
- Concrete experiments
- Open, honest communication
- Trust people’s instincts
- Accepted responsibility
- Local adaptation
- Travel light
- Honest measurement

XP practices - I

- The planning game (scoping)
  - from business priorities and technical estimates
- Small releases (production quality increments)
  - quick short increments
- Metaphor (guiding idea from customer domain)
  - concept of how the system works

XP practices -2

- Simple design
- Testing (verification and validation)
  - customers write tests too
- Refactoring (to simplify)
  - emphasis on communication
- Pair programming (verification)

XP practices -3

- Collective ownership (changes by anyone)
- Continuous integration (multiple times a day)
- 40-hour week
- On-site customer
- Coding standards

XP “process” - I

- Planning
  - set of user stories written by customers
  - describes required functionality
  - prioritized by customer, estimated by team
  - grouped and implemented
  - project velocity is calculated for next increment

Example of XP user story

Customer Story and Task Card B/W Development \ COLA

DATE: 3/19/98 TYPE OF ACTIVITY: NEW: X FIX: ___ ENHANCE: ___ FUNC. TEST

STORY NUMBER: ~~___~~ 1275 PRIORITY: USER: ___ TECH: ___

PRIOR REFERENCE: ___ RISK : ___ TECH ESTIMATE: ___

TASK DESCRIPTION:
SPLIT COLA: When the COLA rate chgs. in the middle of the B/W Pay Period we will want to pay the 1ST week of the pay period at the OLD COLA rate and the 2ND week of the Pay Period at the NEW COLA rate. Should occur "automatically based on system design.

NOTES:
For the OT, we will run a m/f ramo program that will pay or calc the COLA on the 2ND week of OT. The plant currently retransmits the hours data for the 2ND week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA

TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEEntExcessCOLA BIN.

Date	Status	To Do	Comments

XP “process” -2

- Design
  - start with test cases
  - implement enough to run tests
  - simplify code if possible
  - run tests again
  - repeat

XP “process” -3

- Coding
  - get a partner
  - code while partner thinks strategy
- Testing
  - automated tests ever few hours
  - customer acceptance test after each release

end