If you have further interest, see [Hig011, [Hig02al, and [DeM02] for an enter- taining summary of other important technical and political issues.

CE

5.4 EXTREME PROGRAMMING

Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation. XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.

5.4.1 The XP Process

CXP is built on four main activities:

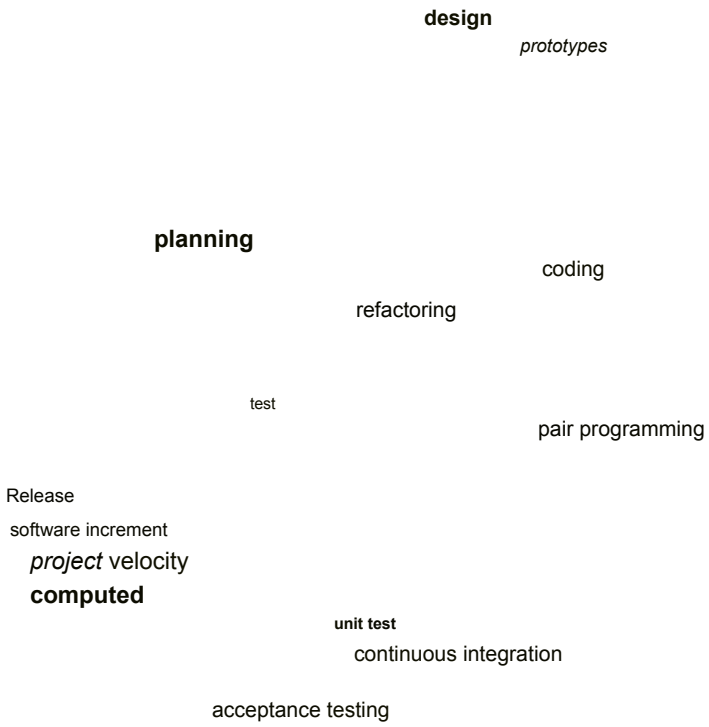
1. Planning
2. Design
3. Coding
4. Testing

Planning. The planning activity (also called *the planning* game) begins with *listening*-a requirements gathering activity that enables the technical members

FIGURE 5.2

The Extreme Programming process

user stories
values
acceptance test *criteria*
iteration plan
simple design spike solutions
CRC cards



WebRef

A worthwhile XP "planning game" can be found at: http://csis.pace.edu/bergin/xp/planninggame.html

of the XP team to understand the business context for the software and to get a broad feel for required output and major features and functionality. Listening leads to the creation of a set of "stories" (also called *user* stories) that describe required output, features, and functionality for software to be built. Each *story* (similar to use cases described in Chapter 8) is written by the customer and is placed on an index card. The customer assigns a *value* (i.e., a priority) to the story based on the overall business value of the feature or function.2 Members of the XP team then assess each story and assign a *cost*—measured in development weeks to it. If the story is estimated to require more than three development weeks, the customer is asked to split the story into smaller stories and the assignment of value and cost occurs again. It is important to note that new stories can be written at any time.

Customers and developers work together to decide how to group stories into the next release (the next software increment) to be developed by the XP team. Once a basic *commitment* (agreement on stories to be included, delivery date, and other project matters) is made for a release, the XP team orders the stories that will be developed in one of three ways: (1) all stories will be implemented Super immediately (within a few weeks), (2) the stories with highest value will be moved up in the schedule and implemented first, or (3) the riskiest stories will be moved up in the schedule and implemented first.

POINT

Project velocity is a subtle measure of team productivity.

ADVICE

XP deemphasizes the importance of design. Not everyone agrees. In fact, there are times when design should be emphasized.

After the first project release (also called a software increment) has been delivered, the XP team computes project velocity. Stated simply, project velocity is the number of customer stories implemented during the first release. Project velocity can then be used to (1) help estimate delivery dates and schedule for subsequent releases and (2) determine whether an overcommitment has been made for all stories across the entire development project. If an overcommitment occurs, the content of releases is modified or end delivery dates are changed.

As development work proceeds, the customer can add stories, change the value of an existing story, split stories, or eliminate them. The XP team then reconsiders all remaining releases and modifies its plans accordingly.

Design. XP design rigorously follows the KIS (keep it simple) principle. A simple design is always preferred over a more complex representation. In addition, the design provides implementation guidance for a story as it is written—nothing less, nothing more. The design of extra functionality (because the developer assumes it will be required later) is discouraged.[2]

XP encourages the use of CRC cards (Chapter 10) as an effective mechanism for thinking about the software in an object-oriented context. CRC

[2] The value of a story may also be dependent on the presence of another story.

[3] These design guidelines should be followed in every software engineering method, although there are times when sophisticated design notation and terminology may get in the way of simplicity.

WebRef

Refactoring techniques and tools can be found at: www.refactoring.com.

POINT

Refactoring improves the internal structure of

a design (or source code) without changing its external functionality or behavior.

WebRef

Useful information on XP can be obtained www.xprogram-ming.com.

class-responsibility-collaborator) cards identify and organize the object- oriented classes that are relevant to the current software increment. The XP team conducts the design exercise using a process similar to the one described in Chapter 10. The CRC cards are the only design work product produced as part of the XP process.

If a difficult design problem is encountered as part of the design of a story, **XP** recommends the immediate creation of an operational prototype of that portion of the design. Called a spike *solution*, the design prototype is implemented and evaluated. The intent is to lower risk when true implementation starts and to validate the original estimates for the story containing the design problem.

XP encourages *refactoring*-a construction technique that is also a design technique.

Fowler [Fow00l describes refactoring in the following manner:

> Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves the internal structure. It is a disciplined way to clean up code land modify/simplify the internal designl that minimizes the chances of introducing bugs. In essence, when you refactor you are improving the design of the code after it has been written.

Because XP design uses virtually no notation and produces few, if any, work products other than CRC cards and spike solutions, design is viewed as a tran- sient artifact that can and should be continually modified as construction pro- ceeds. The intent of refactoring is to control these modifications by suggesting small design changes that "can radically improve the design" [Fow00]. It should be noted, however, that the effort required for refactoring can grow dramatically as the size of an application grows.

A central notion in XP is that design occurs both before *and after* coding com- mences. Refactoring means that design occurs continuously as the system is con- structed. In fact, the construction activity itself will provide the XP team with guidance on how to improve the design.

Coding. After stories are developed and preliminary design work is done, the team does not move to

code, but rather develops a series of unit tests that will exercise each of the stories that is to be included in the current release (software increment). Once the unit test has been created, the developer is better able to focus on what must be implemented to pass the test, Nothing extraneous is added

4 Object-oriented classes are discussed in Appendix 2, in Chapter10, and throughout Part 2 of this book.

5

6 This approach is analogous to knowing the exam questions before you begin to study. It makes studying much easier by focusing attention only on the questions that will be asked.

Unit testing, discussed in detail in Chapter 22. focuses on an individual software component, exercising the component's interface, data structures, and functionality in an effort to uncover errors that are local to the component.

What is pair programming ?

ADVICE

Many software *teams are populated by individualists. You'll have to* work to *change that culture if pair programming is to* work *effectively*.

How are unit tests used in **XP**?

POINT

XP acceptance tests are derived from user stories

(KIS). Once the code is complete, it can be unit-tested immediately, thereby pro- viding instantaneous feedback to the developers.

A key concept during the coding activity (and one of the most talked-about as- pects of XP) is *pair programming*. XP recommends that two people work together at one computer workstation to create code for a story. This provides a mechanism for real-time problem solving (two heads are often better than one) and real-time quality assurance (the code is reviewed as it is created) It also keeps the develop- ers focused on the problem at hand. In practice, each person takes on a slightly different role. For example, one person might think about the coding details of a particular portion of the design while the other ensures that coding standards (a required part of XP) are being followed or that the code for the story will satisfy the unit test that has been developed to validate the code against the story.?

As pair programmers complete their work, the code they develop is inte- grated with the work of others. In some cases this is performed on a daily basis by an integration team. In other cases, the pair programmers have integration responsibility. This "continuous integration" strategy helps to avoid compatibility and interfacing problems and provides a "smoke testing" environment (Chap- ter 22) that helps to uncover errors early.

Testing. The unit tests that are created should be implemented using a frame- work that enables them to be automated (hence, they can be executed easily and repeatedly). This encourages a regression testing strategy (Chapter 22) when- ever code is modified (which is often, given the XP refactoring philosophy).

As the individual unit tests are organized into a "universal testing suite" [We1991, integration and validation testing of the system can occur on a daily basis This provides the XP team with a continual indication of progress and also can raise warning flags early if things go awry. Wells [We199] states: "Fixing small problems every few hours takes less time than fixing huge problems just before the deadline."

XP *acceptance tests*, also called *customer tests*, are specified by the customer and focus on overall system features and functionality that are visible and re- r stories that viewable by the customer. Acceptance tests are derived from user have been implemented as part of a software release.