Extreme Programming from a CxOne Point of View

A Construx Software Whitepaper

11 October 2001

Construx

Delivering Software Project Success

For more information on Construx's products and services,
please see our website at www.construx.com.

Contents

Executive Summary

This paper explores Extreme Programming (nicknamed “XP”) and its relationship to Construx® Software’s CxOne™ software process framework. This paper discusses similarities and differences and describes how the two methodologies can be used together.

Extreme Programming is an emerging industry Methodology that is characterized by small releases; intensive, project-long customer involvement; simple design; test-first programming; refactoring; and pair programming. XP’s focus is in-house information systems projects with small development teams.

CxOne is Construx Software’s toolbox of materials that provide practitioner support for software development. CxOne is a flexible process framework based on industry best practices. This framework supports a diverse set of processes that are optimized to best meet the business needs of each organization and project.

CxOne can provide tools for implementing some XP practices, and also provide support for addressing necessary issues that may be outside the scope of XP.

Overview of Extreme Programming (XP)

Extreme Programming (nicknamed "XP") is a methodology for building software that was originally designed to work on in-house information systems projects with 2-10 developers. XP is based on longstanding industry best practices, including evolutionary prototyping[1], short release cycles[2], and active end-user involvement in requirements definition.[3] XP's contribution is that it has bundled and packaged a specific set of practices to form a methodology. The specific practices were chosen based on the belief that they are mutually supportive. XP is based on the work of Kent Beck, Ward Cunningham, Ron Jeffries, and others.

Keys to Success

Keys to success with Extreme Programming include active management, tool support for test-first programming, facilities support for pair programming, frequent access to an on-site customer, and adequate training for technical personnel in planning, test case design, and pair programming.

Main Benefits

XP introduces a structured software development methodology to project teams that have previously been exposed primarily to code-and-fix approaches. It provides a training ground that exposes teams to the benefits of more structured software development approaches.

When to Use

Extreme programming is best suited to small to medium-size projects with total staff of less than 10 people and total duration of 1-6 months. Because XP is based on the evolutionary prototyping approach, it is well-suited for in-house IS/MIS/IT development pro-

[1] Steve McConnell, *Rapid Development*, Microsoft Press, 1996 (Chapter 21).

[2] Tom Gilb, *Principles of Software Engineering Management*. Wokingham, England: Addison-Wesley, 1988.

[3] The Standish Group, "Charting the Seas of Information Technology," Dennis, MA: The Standish Group, 1994.

jects. It is a reasonable choice when the main risks a project faces are changing requirements, significant mismatch between project scope and available schedule, and technical staff members that are not currently using advanced software practices. In such cases, CxOne's materials can provide a springboard for effective deployment of XP.

XP is not well suited to large projects, long projects, or projects with high reliability requirements. It is not well suited to projects that face risks in areas other than requirements change, schedule, or staff inexperience in advanced software practices. It is not well suited to projects on which systems engineering is needed. Experienced technical managers, programmer, and testers may occasionally choose to use XP "by the book," but they will usually be able to achieve more efficient results through custom plans based on the unique needs of their specific project.

Main Risks

Active management is needed to ensure that Extreme Programming does not devolve into code and fix. Careful assessment of the business context into which XP will be deployed is needed to ensure that XP is not treated as a "one size fits all" methodology.

Interactions with Other Practices

Extreme programming excludes use of evolutionary delivery, staged delivery, and other more structured lifecycle models. It excludes use of formal review techniques such as Fagan inspections. It creates significant redundant effort in cases in which independent testing is required. XP excludes use of up-front requirements practices such as JAD sessions and User Interface Prototyping.

Further Information

The main sources of information on XP are:

- *Extreme Programming Explained*, Kent Beck
- *Planning Extreme Programming*, Kent Beck and Martin Fowler
- *Extreme Programming Installed*, Ron Jeffries, Ann Anderson, and Chet Hendrickson

For information on Extreme Programming, see the books listed above or the Extreme Programming website at http://www.extremeprogramming.org/.

Overview of CxOne

CxOne is an adaptable, full-lifecycle software engineering process framework that is lightweight and document based. It is designed to be used both off the shelf as a basic project methodology, and as a toolbox that is tailored to the unique business needs of specific organizations. CxOne identifies pragmatic artifacts created by projects and organizations (e.g. project plans, requirements, designs, code, technology management plans, etc.) and provides materials to support the creation of those artifacts including checklists, templates, best practices, patterns, standards, and guides.

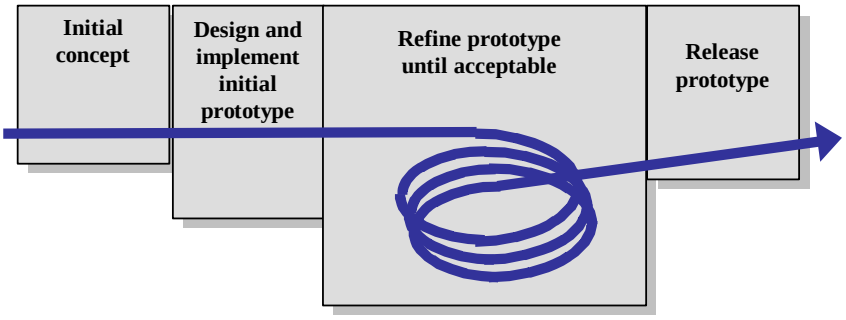
CxOne is synthesized from industry sources including Construx's experience from consulting, training, and its own software projects; Steve McConnell's books and articles; IEEE software engineering standards; and the Software Engineering Body of Knowledge (SWEBOK).

CxOne is offered in Basic and Enterprise versions. CxOne Basic is provided free of change and supports most mainstream projects and the evaluation of CxOne Enterprise. CxOne Enterprise provides organizational support and full project support through customization capabilities and a broad range of materials.

For more information on CxOne, see the CxOne web site at www.construx.com/cxone.

CxOne Support for Extreme Programming

In CxOne terminology, Extreme Programming is based on the evolutionary prototyping lifecycle model. In this lifecycle model, a system is developed in increments so that it can quickly be modified in response to end-user and customer feedback. Most evolutionary-prototyping efforts begin by prototyping the user interface and evolving the completed system from that, but prototyping can start with any high-risk area.



In XP's version of evolutionary prototyping, the project team prototypes business requirements in order of priority. XP spells out numerous requirements for how activity in each prototyping iteration should be conducted. Specifically, XP requires that projects contain 12 specific practices:

- Planning Game
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- 40 Hour Week
- Onsite Customer
- Coding Standard

Each of these practices is described in more detail in the following sections.

XP's Engineering Practices

Each of XP's 12 practices are variations of well-established industry best practices or common practices. The table below summarizes the XP practices, describes the industry

best practice on which they are based, and describes CxOne support for each specific XP practice and for related practices.

Description of XP Engineering Practices

XP Practice	Summary	Industry Practice	CxOne Support
Planning Game	Determines the scope for the next release by selecting *stories* on which the release is based	Project Planning *Fundamental*	Lite Project Plan
Small Releases	A release containing some new business value occurs approximately every two weeks.	Evolutionary Prototyping *Best Practice*	Incremental Lifecycles Miniature Milestones
Metaphor	The high level vision of the entire system is used to drive the detailed design and construction	Requirements Engineering *Fundamental*	JAD Use Cases Requirements Modeling
Simple Design	The design of the system should be as simple as possible to support the current functionality needs	Simple Design *Fundamental*	Design Checklists Design for Change
Testing	Developer's create unit tests prior to construction; referred to as *Test First*. Customers create functional tests. Tests are reused during the project	Testing Test Cases *Fundamental*	Test Plan Construction Testing Testing Checklists
Refactoring	The system is restructured and reworked throughout the project to decrease complexity, increase flexibility, etc.	Refactoring Incremental Development *Fundamental*	Evolutionary Lifecycles Refactoring Maintenance

XP Practice	Summary	Industry Practice	CxOne Support
Pair Programming	Code is written by two people on the same machine at the same time.	*Extreme Practice*	Informal and Formal Reviews Collaborative Construction
Collective Ownership	Any code in the system can be changed by anyone at any-time.	Unselfish Teams / Egoless Programming *Good Practice*	Informal and Formal Reviews Collaborative Construction
Continuous Integration	Code changes or additions are integrated into the production build every time a small task is completed and no less than one a day.	Integration *Best Practice*	Daily Build and Smoke Test
40 Hour Week	Overtime is required no more than one week at a time.	Quality of Life *Good Practice*	Productivity Environments
On-site Customer	A customer or the selected customer representative is always available to provide and select stories and answer questions.	Requirements Engineering *Fundamental*	JAD Use Cases Requirements Modeling
Coding Standard	Code is written according to a set of rules to help improve communication via code.	Coding Standard *Best Practice*	Coding Standards

The “Industry Practice” column in the table defines what existing industry practices the XP practice is based on or equates to. The italicized words in this column are used to classify these practices based on effectiveness and maturity:

Fundamental identifies activities that all software projects should always do to allow basic success. Fundamentals include items such as planning, determining requirements, doing design, performing reviews and testing, good construction practices, and so on.

Best Practice is a common term used to identify practices proven through experience in the software industry to consistently increase the likelihood of success on a broad range or projects.

Good Practice covers practices that have not been proven to provide excellent results on a consistent basis.

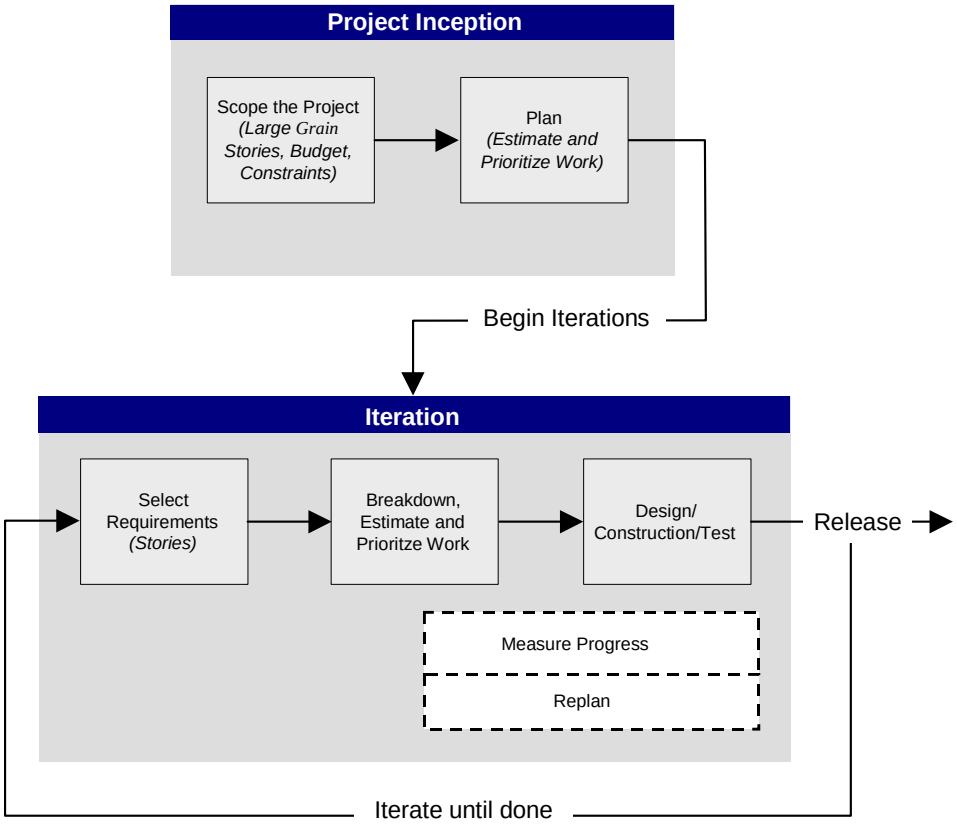
Extreme Practice is used to identify practices identified exclusively with Extreme Programming.

For a summary of fundamentals, refer to Steve McConnell's *Rapid Development*. For information on other practices, see a general software-engineering textbook such as Roger Pressman's *Software Engineering: A Practitioner's Approach* or Ian Sommerville's *Software Engineering*.

CxOne's Support for XP Throughout the Project Lifecycle

CxOne checklists, templates, and guides may be used to support projects using Extreme Programming. This section describes how CxOne documents map to each of the activities within Extreme Programming.

The figure below outlines the workflow for a software project run using Extreme Programming.



Extreme Programming Workflow

The following sections list the CxOne materials that map to each Extreme Programming workflow. For more details on each CxOne material, see the CxOne website at www.construx.com/cxone.

Project Inception

Project Management

- Lite Project Plan and Project Charter
- Scheduling and Estimation checklists

Requirements

- Use Case Templates and Checklists
- Best Practice – Joint Application Development

Iterations

Requirements

- Best Practice – Requirements Scrubbing

Project Planning and Management

- Scheduling and Estimation checklists

Design

- Design Checklists
- Best Practice - Design for Change

Construction

- Construction Templates, Checklists, and Standards
- Best Practice – Collaborative Construction
- Best Practice – Daily Build and Smoke Test

Test

- Test Case Checklists
- Best Practice – Construction Testing

Software Release

- Release Sign-Off Template
- Release Checklist

Process Options

A project could use CxOne materials to support an Extreme Programming project, as described in the preceding section. If each of XP's practices is considered to be a tool in a software engineering toolbox, however, an effective project planner should always ask whether each specific practice is the right tool for the specific project. The following table lists tools that should be considered as alternatives to the standard XP tools before a practice is finally selected for a specific project.

Extensions and Alternatives to XP Practices supported by CxOne

XP Practice	Alternative Practices to Consider
Planning Game	Project Chartering
Small Releases	Evolutionary Delivery
	Evolutionary Prototyping
	Staged Delivery
	Design to Schedule
	Miniature Milestones
Metaphor	User Interface Prototyping
	Requirements Modeling
	Requirements Scrubbing
Simple Design	Design for Change
Testing	Construction Testing
	Requirements Fit Criteria
Refactoring	-

XP Practice	Alternative Practices to Consider
Pair Programming	Tester/Developer Buddy System
	Formal Reviews
	Informal Reviews
	Collaborative Construction
Collective Ownership	Formal Reviews
	Informal Reviews
	Collaborative Construction
Continuous Integration	Daily Build and Smoke Test
40 Hour Week	Productivity Environments
Onsite Customer	Joint Application Development
	Use Cases
	User Interface Prototyping
	Requirements Modeling
Coding Standard	-

Conclusion

Extreme Programming consists of a predefined, fixed set of practices based on an evolutionary prototyping lifecycle model. XP is intended to increase a project's ability to handle changing requirements, improve progress visibility, and enhance end-user and customer feedback. XP is well-supported through many books, articles, conference papers, and XP coaches.

XP is best suited for projects that have high requirements uncertainty and a need to maximize visibility into progress. XP is well-suited to small- to medium-sized in-house development projects that have moderate reliability requirements, require moderate economic efficiency, and that are intended to be conducted by staff with moderate experience levels. CxOne's toolbox nature allows its materials to be used to support such XP projects.

A fundamental difference between XP and CxOne is that CxOne allows different lifecycle models and practices to be used for different projects and organizations. The lifecycle model chosen impacts the organization's ability to improve development speed, improve quality, track and control projects, minimize overhead, minimize risk exposure, and satisfy client expectations. CxOne supports choosing lifecycles and practices that best meet each specific project's needs.

CxOne tailoring must be performed by highly trained and experienced staff, which will craft an implementation of CxOne suited to the organization's size, reliability needs, development efficiency needs, and staff skill level. Once tailored, CxOne can be used by staff at any skill level.

Because CxOne draws from established industry best practices, it is supported by dozens of books and articles published over the past few decades, in addition to Construx's CxOne website and other Construx materials.

Using CxOne's toolbox metaphor, XP can be seen as one specialized set of tools that are designed for a specific purpose. CxOne provides a comprehensive set of tools that can be applied to a broad range of projects.