Extreme Methodologies for an Extreme World

Theron Leishman
Software Technology Support Center

The world we live in today demands greater availability of decision-making information. The use of cell phones, hand held computers, online banking, and internet stock trading are only a few examples of this demand for timely information. Information Technology organizations are feverishly scrambling in an attempt to gain ground on the ever increasing demand for their services. This article takes a glimpse into the software development methodologies that are being applied in an attempt to catch up in the rapidly changing world in which we live. A brief insight into traditional development methods will be followed by an analysis of the principles underlying the new, less traditional methods. These newer, extreme methodologies share many common characteristics, which will be identified and discussed. An analysis will also be made of the ability of extreme methodologies to meet the rigor demanded by the Software Capability Maturity Model.

While enjoying a day on the slopes at one of the local Utah ski resorts, I had the chance to ski on the course that has been developed for the downhill ski races in the upcoming 2002 Winter Olympic Games. As I spent the day in my feeble attempt to navigate the course, I could only image the adrenaline rush experienced by those who will be competing in a sport in which skiers race at speeds in excess of 70 miles per hour.

Another winter sport not as well known is speed skiing. Called the fastest non-motorized sport on earth, skiers plunge down a steep ramp of ice propelling themselves from 0 to 154 mph in about 10 seconds. Is this insanity? Why would anyone in his or her right mind participate in this sport?

Growing numbers of people are enjoying high risk, *extreme* activities. Everything in our world today seems to be following this extreme theme. Daily we are bombarded with the demand for things to be bigger, better, and faster! What was once viewed as adequate is now considered substandard and lack-luster.

Business and government organizations are not immune from this phenomenon. Developments in recent years have resulted in an environment where information is needed now! Decisions must be made quickly! Fortunes can be made, governments can fall, and wars can be won or lost based on the availability of information to make sound decisions.

This rapid pace has had substantial impact on computer resources. The boom in web technologies is an example of this demand for information. For companies to remain competitive, the amount of information available on the web has skyrocketed. Information technology organizations have had to scramble to respond to this demand. New techniques are evolving in an attempt to keep pace with growing demand. Into this environment has sprung *extreme software development methods*.

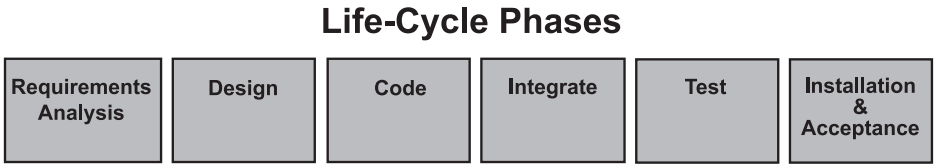


Figure 1: *Traditional Software Development Phases*

Is an extreme programmer a wild and crazy thrill seeker, pounding out code with no concern for his own safety or the security and stability of the organization? What about the project lead that leaps from tall buildings in a single bound with no thought of standards, best practices, or the future maintainability of the application? These are the visions that come to mind when thinking of extreme development methods. However, before validating or dispelling this stereotype, a review of traditional development methods comes first.

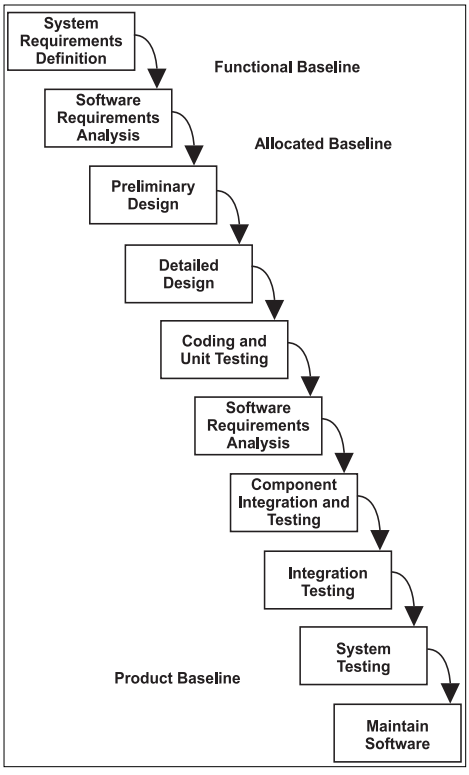
Traditional Development Methods

Traditional software development methodologies were developed to standardize the procedures used to develop and maintain software. These methodologies follow the general phases outlined in Figure 1. These methodologies are usually based on one of the following three basic models:

The Waterfall Model: The Waterfall model for software development is a step-by-step process. The requirements for one phase are completed prior to the next phase beginning. This is a onetime through approach. The phases build upon each other toward the completion of the development effort. Figure 2 illustrates the sequential nature of a typical waterfall methodology.

The Incremental Model: Using the incremental approach, user needs are determined and system requirements are defined. Then the rest of the development is performed in a sequence of builds. The first build incorporates part of the planned

Figure 2: *Waterfall Model*



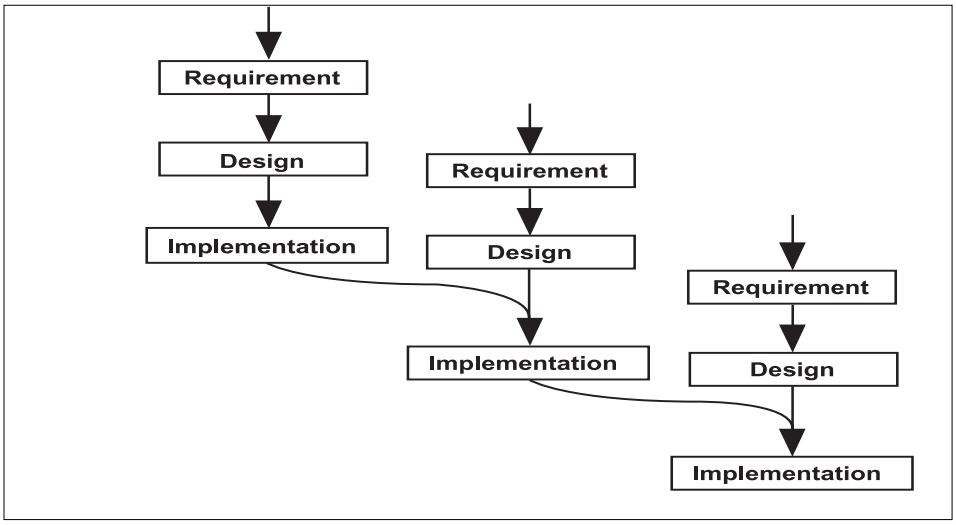


Figure 3: *Incremental Model*

capabilities. The next build adds additional capability. This incremental process continues until the system is complete. An example of the incremental model is presented in Figure 3.

The Evolutionary Model: An evolutionary, or spiral, model also uses a system of builds as in the incremental model. The variation comes in the acknowledgement up front that the user needs and requirements are not fully understood at the inception of the project. Using this strategy, the user needs and system requirements are partially defined at the inception of the project. They are then refined in each succeeding build. A pictorial representation of the evolutionary model is in Figure 4.

Extreme Development Methodologies

Extreme methodologies take a different approach from traditional software development methodologies. These methodologies accept the notion that, change happens during all phases of the development process. Further, it is anticipated that the system users may not know exactly what they want the final product of the development effort to be.

A number of non-traditional methodologies have evolved in recent years. Methodologies such as Extreme Programming (XP), Adaptive Software Development (ASD), SCRUM, and Crystal Light are a few of the methodologies attempting to achieve notoriety in these extreme times.

Extreme methodologies tend to be tailored after the evolutionary or spiral development model. They use phases like inception, elaboration, construction, and transition or speculation, collaboration, and learning to streamline the development approach of traditional methodologies. These methods attempt to satisfy customer requirements by speeding development and delivering useful products in rapid iterations. Full functionality is achieved over time, while partial functionality is achieved quickly. Figure 5 presents a comparison of the phases of traditional life-cycle methodologies to those of typical extreme methodologies.

Each of the extreme methods have developed it's own approach to conquering the brave new world of speedy development. Underlying each of these methods is a foundation of basic characteristics common to each. A summary of these similarities follows.

An Iterative Approach: These methods are designed for use in uncertain conditions. They accept the fact that change is inevitable, therefore the system evolves through various iterations. Requirements are identified and incorporated into each iteration of the system. The application is presented to the customer in releases providing increased value to the customer with each iteration until the desired system evolves.

Risk Driven: With each development iteration, risk is evaluated and acts as a driving factor in the evolution of the system. If additional requested functionality causes system risk, or increases the complexity of the application to the point of impacting developmental cost or schedule, the risk is identified and dealt with as it arises.

Studies of software development projects have shown that projects are often over budget, behind schedule, and do not provide the desired functionality. All too often these projects are never used in a production environment. The risk of this type of waste is greatly reduced by extreme methodologies. As iterations of the prod-

Figure 4: *Evolutionary/Spiral Model*

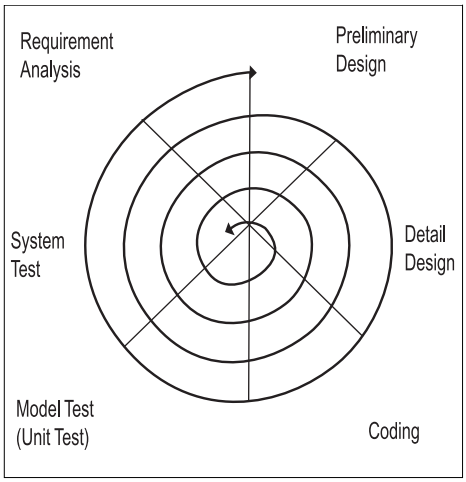
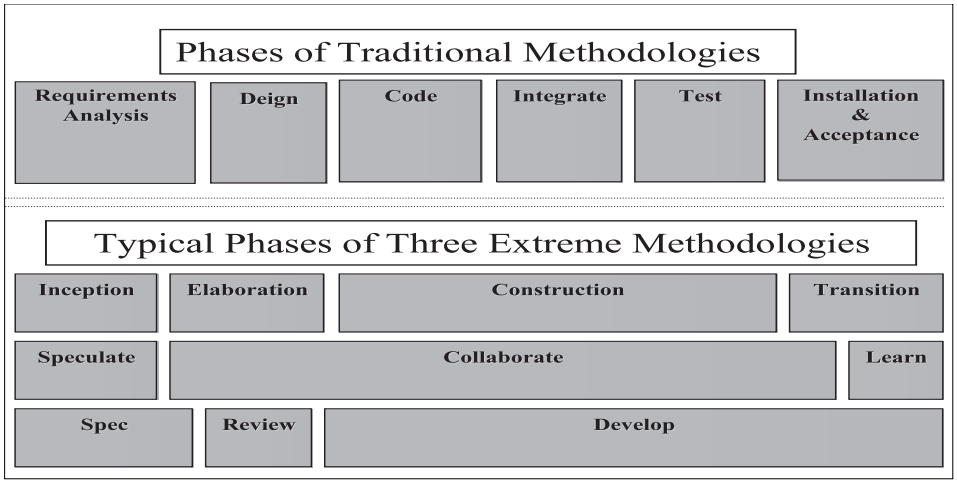


Figure 5: *Traditional Life-Cycle vs. Extreme Methodologies*



uct are completed, design errors and omissions are identified and corrected. Large investments in applications that provide little or no value can be eliminated. Systems evolve that either meet user requirements, or the development is terminated prior to large investments being made in applications that do not warrant the investment.

Results Oriented: Extreme methodologies are intended to achieve results that are often not possible using other approaches. These methodologies portray a *movers and shakers* attitude by focusing on results. These methods build in processes intended to clear the tracks for the express development train to move at extreme speeds. Extreme methods get results in these ways:

- Removing impediments to progress.
- Assuring prompt and timely decision making.
- Isolating the project and team members from irrelevant issues.
- Utilizing all resources and expertise required to achieve success.
- Rewarding individuals and teams involved with successful projects as the success occurs.
- Focusing the attention of team members on the extreme project and nothing else.

Use Sound Development Processes

In an article published in *Software Testing & Quality Engineering*, Jim Highsmith stated that the Old World was one of optimization where efficiency, predictability, and control dominated. The New World is one of adaptation in which change, improvisation, and innovation rule. This dichotomy – optimization versus adaptation – provides a distinct way of viewing the future of software project management.

A study of extreme methodologies soon leads to the determination that these methods do not promote out-of-control, free-for-all programming. These methodologies recognize value in the lessons learned from the history of software development. Although they emphasize the need to rethink the traditional methods and eliminate processes that are not value added, they recognize the need for sound development processes. For example, the extreme methodologies studied encourage the following:

- Testing at the unit and functional levels.
- Project planning.
- Developing and adhering to sound development standards.
- Using pair programming.
- Developing and following coding standards.
- Releasing software in frequent intervals.
- Establishing configuration management process.
- Providing onsite dedicated customer project support.

Emphasize Collaboration

Extreme methodologies all share the idea of collaboration. The idea of empowered, high-performance teams is key to the success of projects using extreme methodologies. Team members must have complementary skills and be dedicated to the common purpose, performance goals, and approach to which they hold themselves mutually accountable. These empowered teams include all required technical disciplines and user involvement to allow decisions to be made as problems arise.

Having the right people dedicated to a project will ensure that stakeholder and developer alike have agreement on issues from primary requirements through final implementation. Issues of scope, cost, schedule, priorities, risk, and trade-off's between these are well understood by all parties. Surprises are eliminated and successful projects are the result.

A recent study on the cost and benefit of pair programming revealed that a pair of programmers working on a project took 15 percent more time to complete the programming effort than a single programmer working alone. This cost was easily offset by the resulting 15 percent reduction in coding defects. History has shown that the cost to correct defects is exponentially greater than the cost of original coding [1].

A team of programmers developing simulator software for the Israeli Air Force also realized results similar to those identified in the above study. By using pair programming, and being located in the same facility as their customer, the team has been able to greatly reduce the time between software releases. The team is seeing great benefits by using XP principles in their development efforts [2].

It has been said that the number one killer of software development projects is time. Collaboration allows extreme projects to remove barriers and eliminate delays that kill projects.

How Extreme is Extreme?

Upon consideration of the so-called extreme methodologies, the question arises, "What is so extreme?" The characteristics of the extreme methods that appear to make these approaches appealing and successful are not glitzy new tools or earth shaking new discoveries. They are things that contribute to the success of any project following any sound methodology.

Ward Cunningham, one of the fathers of XP has indicated that, "Extreme Programming is a lot of simple little things. It's lots of things that have been done before, some of which have even been discredited. We have reconstructed a collection of practices that support each other in a way that is startling [3]."

The Capability Maturity Model

The Software Engineering Institute developed the Capability Maturity Model® (CMM®) for software, which organizations can use as a base to measure their software development and maintenance processes. It provides a standard for software process improvement to assist organizations desiring to make conscious improvement in their software development efforts. The CMM lays out a path to lead an organization from an ad hoc, immature development organization to a mature, disciplined organization. The CMM contains five levels of maturity. It covers practices for planning, engineering, and managing software development and maintenance [4].

The Key Process Areas (KPAs) of CMM Level 2 and Level 3 cover many areas of sound software development methodologies. Below is a summary of these levels along with a list of the KPAs for each level.

Level II – The focus of CMM Level 2 is on software project management. Software project management processes are to be documented and followed.

Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark office.

Organizational policies guide projects in establishing management processes. Successful practices developed can be repeated. These are the key process areas of Level 2:

- Requirements Management.
- Software Project Planning.
- Software Project Tracking and Oversight.
- Software Subcontract Management.
- Software Quality Assurance.
- Software Configuration Management.

Level III – The focus of CMM Level 3 is the software engineering process. At Level 3 emphasis moves toward the organization. The organization has processes in place that empower the individuals doing software development. Processes are defined, documented, and understood by individuals within the organization. These are the key process areas of Level 3:

- Organization Process Focus.
- Organization Process Definition.
- Training Program.
- Integrated Software Management.
- Software Product Engineering.
- Intergroup Coordination.
- Peer Reviews.

CMM and Extreme Methodologies

The CMM is intentionally not prescriptive in nature. An organization choosing to follow the framework of the CMM should develop policies and procedures that make sense for its environment. The CMM should be used as a model of the best practice in software development and maintenance.

An established software development methodology is an essential element of a mature software development organization. In addition, standardized development methodologies span many of the KPAs identified within the CMM. However, the CMM does not advocate one development methodology over another. The model does recommend that a sound methodology be followed.

An extreme development methodology can provide the elements necessary for a development organization to be successful in their development efforts. As noted earlier in this article, extreme methodologies advocate sound practices that are defined in the CMM. Principles of planning, requirements management, quality assurance, and project tracking and oversight are all essential elements of a sound extreme methodology. These are all KPAs within the CMM. They are practices essential to the ongoing success of a development and maintenance organization.

Conclusion

The demands of the world we live in require more information to be available faster than any time in history. To remain competitive, an organization must be able to respond to this demand. Extreme software development methodologies have demonstrated their ability to help information technology organizations respond to this growing demand for their services.

These methods are not simple solutions to an age-old problem. Just as the participants of extreme sports have tools and equipment to allow them to reduce the risk of participation in their chosen sport, so also should an extreme methodology reduce risk to the organization. A mature development organization will have sound policies and procedures in place to guide the selection and tailoring of appropriate methodologies.

It is important to remember that not all projects are extreme projects. However, when a project is determined to be extreme, a sound extreme methodology will allow the project to accomplish it's objective while not exposing the organization to undue risk.◆

References

1. Byer, Dr. Sam and Highsmith, Jim, RADical Software Development, *American Prog rammer Magazine*, June 1994.
2. Cockburn, Alistair and Williams, Laurie, The Costs and Benefits of Pair Programming, Humans and Technology Technical Report 2000.01, January 2000.
3. Waters, John K., Extreme Method Simplifies Development Puzzle, *Application Development Trends*, July 2000.
4. Software Engineering Institute, *The Capability Maturity Model Guidelines for Improving the Software Process*, Boston, Addison Wesley, 1994.

Additional Reading

1. Beck, Kent, *Extreme Programming Explained: Embrace Change,* Boston, Addison Wesley, 2000.
2. Experience With XP, c2.com/cgi/wiki?ExperienceWithXP
3. Highsmith, Jim, Retiring Lifecycle Dinosaurs, *Software Testing & Quality Engineering*, July/August 2000.
4. Koerner, Brendan I., Extreme, *U.S. News*, June 30, 1997.
5. Rational Software Corporation, Rational Unified Process: Best Practices for Software Development Teams, White Paper TP-026A Rev 11/98.

About the Author

Theron R. Leishman is a consultant currently under contract in the Software Technology Support Center at Hill AFB, Utah, which provides consulting services to Air Force and other Department of Defense (DoD) and government agencies. He began his career as an Electronic Data Processing auditor assessing software development, software project management, computer security and compliance to government standards and regulations. Leishman has a wide range of experience in various aspects of software design, development, project management, and process improvement. He has successfully managed projects of various size and complexity for the DoD, aerospace, manufacturing, health care, gas and oil, higher education, and various other industries. Leishman is currently employed by TRW.

Software Technology Support Center
7278 4th Street
Bldg. 100 G19
Hill AFB, Utah 84056
Voice:801-775-5738
Fax:801-777-8069
E-mail:Theron.Leishman@hill.af.mil

"Computers are to computing as instruments are to music. Software is the score whose interpretations amplifies our reach and lifts our spirits. Leonardo da Vinci called music the shaping of the invisible, and his phrase is even more apt as a description of software"

Alan C. Kay