SOFTWARE PROJECT MANAGEMENT

LECTURE # 5

SOFTWARE DEVELOPMENT FUNDAMENTALS-III

13th October, 2011

Engr. Ali Javed

Contact Information

- Instructor: Engr. Ali Javed

  Lecturer

  Department of Software Engineering

  U.E.T Taxila

  - Email: ali.javed@uettaxila.edu.pk
  - Contact No: +92-51-9047592
  - Office hours:
    - **Monday, 11:00 - 01:00, Office # 7**

Course Information

- **Course Name: Software Project Management**
- **Course Code: SE-401**
- **CMS Link:** http://web.uettaxila.edu.pk/CMS/AUT2011/seSPMbs/index.asp

Software Models

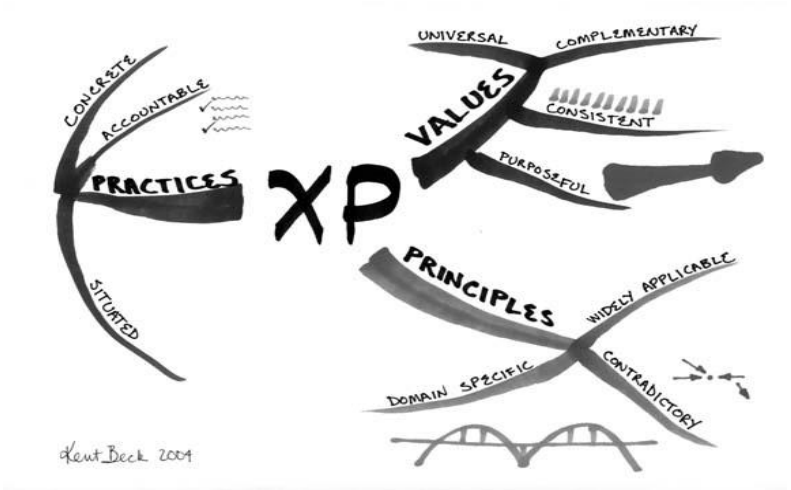
- Agile Development
  - ✓ Scrum
  - ✓ XP
  - ✓ FDD
  - ✓ ASD
  - ✓ DSDM
  - ✓ Lean

XP

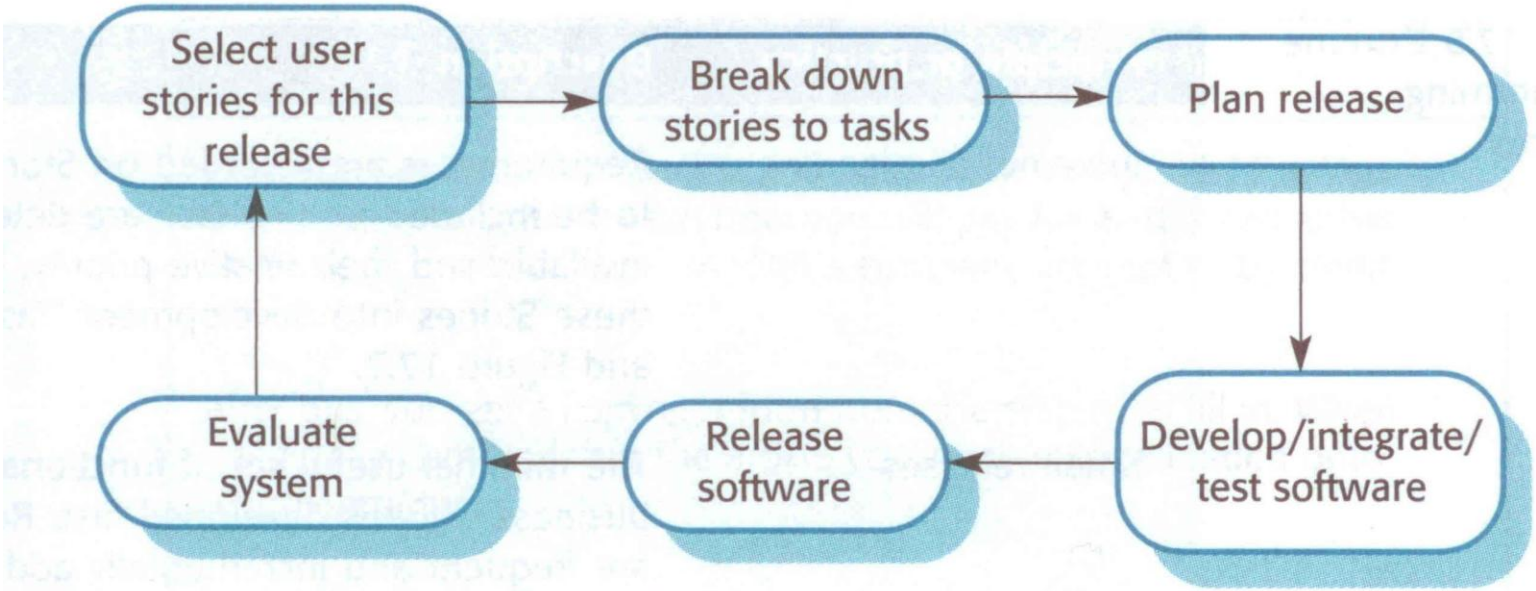
- XP
- XP Release Cycles
- XP Practices

Extreme programming [21]

- **Extreme Programming (XP)** is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles (time boxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted.
- Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure [22], simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers



The XP release cycle



Requirements scenarios

- In XP, user requirements are expressed as scenarios or user stories.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

Story card for document downloading

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it—this can either be through a subscription, though a company account or by credit card.

After this, you get a copyright form from the system to fill in. When you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system printing has been successful.

If the article is a print-only article, you can't keep the PDF version, so it is automatically deleted from your computer.

Testing in XP

- Test-first development.
- Incremental test development from scenarios.
- User involvement in test development and validation.
- Automated test harnesses [12] are used to run all component tests each time that a new release is built.

Task cards for document downloading

Task 1: Implement principal workflow

Task 2: Implement article catalog and selection

Task 3: Implement payment collection

Payment may be made in 3 different ways. The user selects which way they wish to pay. If the user has a library subscription, then they can input the subscriber key which should be checked by the system. Alternatively, they can input an organisational account number. If this is valid, a debit of the cost of the article is posted to this account. Finally, they may input a 16 digit credit card number and expiry date. This should be checked for validity and, if valid a debit is posted to that credit card account.

Pair programming

- In XP, programmers work in pairs, sitting together to develop code.
- This helps develop common ownership of code and spreads knowledge across the team.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- It encourages refactoring as the whole team can benefit from this. [14]
- **Measurements suggest that development productivity with pair programming is similar to that of two people working independently but …….**

Extreme programming practices 1

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figure 17.6 and Figure 17.7.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.

Extreme programming practices 2

Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers own all the code. Anyone can change anything.
Continuous integration	As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium-term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Adaptive Software Development

- Introduction
- ASD Cycle

Adaptive Software Development [15][16]

- **Adaptive Software Development** is a software development process that grew out of rapid application development work by Jim Highsmith and Sam Bayer.

- ASD replaces the traditional waterfall cycle with a repeating series of *speculate*, *collaborate*, and *learn* cycles.

- The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time boxed [13], risk driven, and change tolerant.

Adaptive Software Development [15][16]

- The word *speculate* refers to the paradox of planning. In speculation, user requirements are understood and an adaptive cycle planning is conducted.
- *Collaboration* refers to the efforts for balancing the work based on predictable parts of the environment and adapting to the uncertain surrounding mix of changes caused by various factors – technology, requirements, stakeholders, software vendors, etc.
- Effective collaboration with customer is very important. Communication, teamwork, individual creativity are a part of effective collaboration.

Adaptive Software Development [15][16]

- The *learning* cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During these iterations the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience and eventually mastery in the problem domain. ASD teams can learn through focus groups, formal technical reviews and postmortems.

Adaptive Software development

DEMO

Feature Driven Development

- Introduction
- FDD Activities
- Mile Stones
- Practices in FDD

Feature-driven development [17]

- **Feature-driven development** (**FDD**) is an iterative and incremental software development process. It is one of a number of Agile methods for developing software and forms part of the Agile Alliance.
- Its main purpose is to deliver tangible, working software repeatedly in a timely manner.
- Activities
  - ✓ Develop Overall Model
  - ✓ Build Feature List
  - ✓ Plan By Feature
  - ✓ Design By Feature
  - ✓ Build By Feature

Feature-driven development [17]

- Develop Overall Model
  - ✓ The project starts with a high-level walkthrough of the scope of the system and its context. Next, detailed domain walkthroughs are held for each modeling area.

- Build Feature List
  - ✓ The knowledge that is gathered during the initial modeling is used to identify a list of features. This is done by functionally decomposing the domain into subject areas.
  - ✓ Subject areas each contain business activities, the steps within each business activity form the categorized feature list.
  - ✓ Features in this respect are small pieces of client-valued functions for example: ´Calculate the total of a sale´ or ´Validate the password of a user´.
  - ✓ Features should not take more than two weeks to complete, else they should be broken down into smaller pieces.

Feature-driven development [17]

- Plan By Feature
  - ✓ Now that the feature list is complete, the next step is to produce the development plan. Class ownership is done by ordering and assigning features (or feature sets) as classes to programmers.

- Design By Feature
  - ✓ A design package is produced for each feature. A chief programmer selects a small group of features that are to be developed within two weeks. Together with the corresponding class owners, the chief programmer works out detailed sequence diagrams for each feature and refines the overall model. Next, the class and method prologues are written and finally a design inspection is held.

- Build By Feature
  - ✓ After a successful design inspection a per feature activity to produce a completed client-valued function (feature) is being produced. The class owners develop the actual code for their classes. After a unit test and a successful code inspection, the completed feature is promoted to the main build.

Feature-driven development [17]

- **Milestones**
  - ✓ Since features are small, completing a feature is a relatively small task. For accurate state reporting and keeping track of the software development project it is however important to mark the progress made on each feature.
  - ✓ FDD therefore defines six milestones per feature that are to be completed sequentially.
  - ✓ The first three milestones are completed during the Design By Feature activity, the last three are completed during the Build By Feature activity.
  - ✓ To help with tracking progress, a percentage complete is assigned to each milestone. In the table below the milestones (and their completion percentage) are shown. A feature that is still being coded is 44% complete (Domain Walkthrough 1%, Design 40% and Design Inspection 3% = 44%).

Domain Walkthrough	Design	Design Inspection	Code	Code Inspection	Promote To Build
1%	40%	3%	45%	10%	1%

Feature-driven development [17]

- **Practices in FDD**
  - ✓ Domain Object Modeling
  - ✓ Developing by Feature
  - ✓ Individual Class Ownership
  - ✓ Feature Teams

Feature-driven development [17]

- **Practices in FDD**
  - ✓ Inspection

  - ✓ Configuration Management
  - ✓ Regular Builds

  - ✓ Visibility of Progress and Results

Feature-driven development

DEMO

Dynamic Systems Development Method

- Introduction
- DSDM Phases
- DSDM Principles

Dynamic systems development method

- **Dynamic systems development method** (**DSDM**) is an agile project delivery framework, primarily used as a software development method.
- DSDM is an iterative and incremental approach that embraces principles of Agile development, including continuous user/customer involvement.
- The most recent version of DSDM, launched in 2007, is called DSDM Atern.
- It is extremely useful in projects with low budgets and tight schedule

Dynamic systems development method

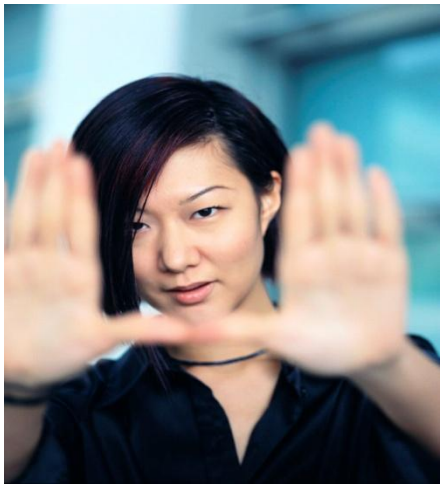
- **Dynamic systems development method** (**DSDM**) has 3 main phases

  - ✓ Phase 1 - The Pre-project
    - In the pre-project phase candidate projects are identified, project funding is realized and project commitment is ensured. Handling these issues at an early stage avoids problems at later stages of the project

  - ✓ Project Lifecycle Phase
    - It depicts the following mentioned stages a project will have to go through to create an implemented system. The first two stages, the Feasibility Study and Business Study are sequential phases that complement to each other. After these phases have been concluded, the system is developed iteratively and incrementally in the Functional Model Iteration, Design & Build Iteration and Implementation stages.

  - ✓ Post-Project Phase
    - The post-project phase ensures the system operates effectively and efficiently. This is realized by maintenance, enhancements and fixes according to DSDM principles. The maintenance can be viewed as continuing development based on the iterative and incremental nature of DSDM.

DSDM Principles



- **Focus on the business need**
  - ✓ The main criteria for acceptance of a "deliverable" is delivering a system that addresses the current business needs. Delivering a perfect system which addresses all possible business needs is less important than focusing on critical functionalities.

- **Deliver on time**
  - ✓ Timebox the work
  - ✓ Focus on business priorities
  - ✓ Always hit deadlines

DSDM Principles

- **Collaborate**
  - ✓ User involvement is the main key in running an efficient and effective project, where both users and developers share a workplace (either physical or via tools), so that the decisions can be made collaboratively and quickly.
    - Involve the right stakeholders, at the right time, throughout the project
    - Ensure that the members of the team are empowered to take decisions on behalf of those they represent without waiting for higher-level approval.
    - Actively involve the business representatives
    - Build a one-team culture

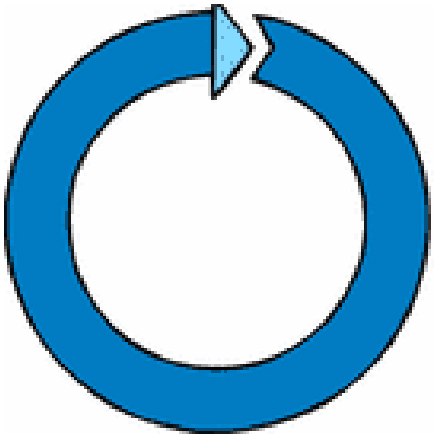
DSDM Principles

- **Never Compromise on Quality**
  - ✓ Set the level of quality at the outset
  - ✓ Ensure that quality does not become a variable
  - ✓ Design, document and test appropriately
  - ✓ Build in quality by constant review
  - ✓ Test early and continuously.

DSDM Principles

- **Build Incrementally from firm foundations**
  - ✓ Strive for early delivery of business benefit where possible
  - ✓ Continually confirm the correct solution is being built
  - ✓ Formally re-assess priorities and ongoing project viability with each delivered increment

- **Develop Iteratively**
  - ✓ A focus on frequent delivery of products, with assumption that to deliver something "good enough" earlier is always better than to deliver everything "perfectly" in the end.



DSDM Principles

- **Communicate Continuously and Clearly**
  - ✓ Communication and cooperation among all project stakeholders is required to be efficient and effective.
    - Run daily team stand-up sessions
    - Use facilitated workshops
    - Use rich communication techniques such as prototyping
    - Keep documentation lean and timely
    - Manage stakeholder expectations throughout the project
    - Encourage informal, face to face communication at all levels

DSDM Principles

- **Demonstrate control**
  - ✓ Use an appropriate level of formality for tracking and reporting
  - ✓ Make plans and progress visible to all
  - ✓ Measure progress through focus on delivery of products rather than completed activities
  - ✓ Manage proactively
  - ✓ Evaluate continuing project viability based on the business objectives

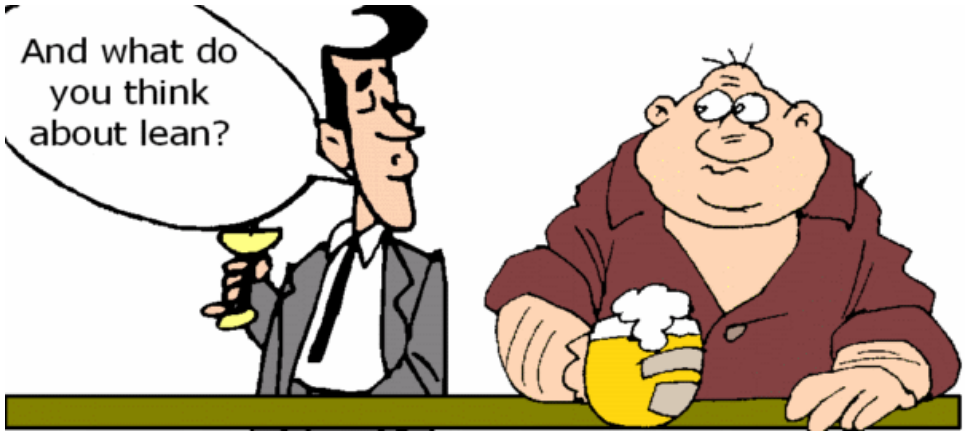
Dynamic systems development method

Lean Software Development

- Introduction
- Lean Principles

Lean software development [18]

- **Lean software development** is a translation of Lean manufacturing and Lean IT principles and practices to the software development domain. Adapted from the Toyota Production System, a pro-lean subculture is emerging from within the Agile community.
- Lean is an Agile methodology which can also be seen as a philosophy
- The core idea is to maximize **customer value** while minimizing waste. Simply, lean means creating more value for customers with fewer resources. [19]
- Eliminating waste along entire value streams, instead of at isolated points, creates processes that need less human effort, less space, less capital, and less time to make products and services at far less costs and with much fewer defects, compared with traditional business systems. [19]



Lean principles [18]

- Eliminating Waste

✓ Everything not adding value to the customer is considered to be waste. This includes:
  - unnecessary code and functionality
  - delay in the software development process
  - unclear requirements
  - insufficient testing, leading to avoidable process repetition
  - slow internal communication

✓ In order to be able to eliminate waste, one should be able to recognize and see it.
  - If some activity could be bypassed or the result could be achieved without it, it is waste.
  - Partially done coding eventually abandoned during the development process is waste.
  - Extra processes and features not often used by customers are waste.
  - Defects and lower quality are waste.

✓ The second step is to point out sources of waste and eliminate them.

Lean principles [18]

- Amplify Learning

- ✓ Software development is a continuous learning process. The best approach for improving a software development environment is to amplify learning.
- ✓ The accumulation of defects should be prevented by running tests as soon as the code is written.
- ✓ The process of user requirements gathering could be simplified by presenting screens to the end-users and getting their input.
- ✓ Increasing feedback via short feedback sessions with customers helps when determining the current phase of development and adjusting efforts for future improvements. During those short sessions both customer representatives and the development team learn more about the domain problem and figure out possible solutions for further development.

Lean principles [18]

- Decide as late as possible

- ✓ As software development is always associated with some uncertainty, better results should be achieved with an options-based approach, delaying decisions as much as possible until they can be made based on facts and not on uncertain assumptions and predictions.
- ✓ An agile software development approach can move the building of options earlier for customers, thus delaying certain crucial decisions until customers have realized their needs better. This also allows later adaptation to changes

Lean principles [18]

- Deliver as fast as possible

- ✓ In the era of rapid technology evolution, it is not the biggest that survives, but the fastest.
- ✓ The sooner the end product is delivered without considerable defect, the sooner feedback can be received, and incorporated into the next iteration.
- ✓ The shorter the iterations, the better the learning and communication within the team.
- ✓ This gives them the opportunity to delay making up their minds about what they really require until they gain better knowledge. Customers value rapid delivery of a quality product.

Lean principles [18]

- Empower the Team

- ✓ There has been a traditional belief in most businesses about the decision-making in the organization – the managers tell the workers how to do their own job.
- ✓ The lean approach favors this: "find good people and let them do their own job," encouraging progress, catching errors, and removing problems, but not micro-managing.
- ✓ People need motivation and a higher purpose to work for – purpose with the assurance that the team might choose its own commitments. The developers should be given access to customer; the team leader should provide support and help in difficult situations.

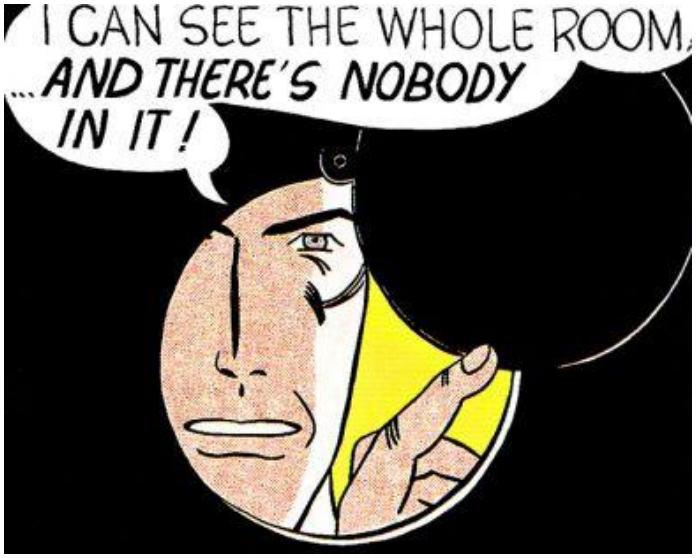
Lean principles [18]

- Build Integrity in

- ✓ The totality of the product achieves a balance of function, usability, reliability and economy that delight customers – this is the so called perceived integrity [23]
- ✓ Conceptual integrity is the principle that anywhere you look in your system, you can tell that the design is part of the same overall design. [20]
- ✓ One of the healthy ways towards integral architecture is refactoring. Repetitions in the code are signs for bad code designs and should be avoided.
- ✓ At the end the integrity should be verified with thorough testing, thus ensuring the System does what the customer expects it to. Automated tests are also considered part of the production process, and therefore if they do not add value they should be considered waste. Automated testing should not be a goal, but rather a means to an end, specifically the reduction of defects.

Lean principles [18]

- See the Whole
  - ✓ Software systems nowadays are not simply the sum of their parts, but also the product of their interactions. Defects in software tend to accumulate during the development process – by decomposing the big tasks into smaller tasks, and by standardizing different stages of development, the root causes of defects should be found and eliminated.



Lean Software Development

DEMO

References

1. Software Engineering by Roger Pressman
2. Software Engineering by Ian Sommerville
3. http://3back.com/scrum/certified-scrummaster-training/
4. http://scrummethodology.com/the-scrum-team-role/
5. YouTube - Scrum Master in Under 10 Minutes (HD) by @hamids.wmv
6. http://scrummethodology.com/scrum-product-owner/
7. http://www.codeproject.com/KB/architecture/scrum.aspx
8. http://abrachan.net/scrum/scrum/sprint-retrospective-meeting/
9. http://epf.eclipse.org/wikis/scrum/Scrum/workproducts/sprint_burndown_chart_F647C347.html
10. http://en.wikipedia.org/wiki/Scrum_%28development%29
11. http://msdn.microsoft.com/en-us/library/dd997796.aspx
12. http://en.wikipedia.org/wiki/Test_harness
13. http://en.wikipedia.org/wiki/Timeboxing
14. http://en.wikipedia.org/wiki/Code_refactoring
15. http://en.wikipedia.org/wiki/Adaptive_Software_Development
16. http://productdevelop.blogspot.com/2011/07/what-is-adaptive-software-development.html

References

17. http://en.wikipedia.org/wiki/Feature-driven_development
18. http://en.wikipedia.org/wiki/Lean_software_development
19. http://www.lean.org/whatslean/
20. http://cseweb.ucsd.edu/users/wgg/CSE131B/Design/node6.html
21. http://en.wikipedia.org/wiki/Extreme_Programming
22. http://www.learnmanagement2.com/flat%20structure.htm
23. http://books.google.com.pk/books?id=8o1eom6ifIMC&pg=PA17&lpg=PA17&dq=difference+between+perceived+integrity+and+conceptual+integrity&source=bl&ots=m1VuEK93KM&sig=5TcbxyjCdO57OJ7VIvC4TLk3ELI&hl=en&ei=VXmWTruiLITKhAeng62EBA&sa=X&oi=book_result&ct=result&resnum=3&ved=0CC0Q6AEwAg#v=onepage&q=difference%20between%20perceived%20integrity%20and%20conceptual%20integrity&f=false

For any query Feel Free to ask