Agile Methodologies

Venkat Subramaniam

venkats@agiledeveloper.com

http://www.agiledeveloper.com/download.aspx

Abstract

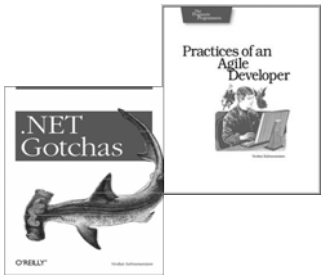
Abstract Agile development is picking up steam. You have most likely heard about eXtreme Programming(XP). What other Agile methodologies are you familiar with and what do they bring of interest or significant to the table of Agility? More important, why should you learn about these different methodologies instead of simply focusing on one? There is no one shoe that fits all. Any methodology that requires you to follow it in totality and not let you adapt is rather dogmatic, not pragmatic. To be effective we have to take the best of different approaches and apply to our projects base on our specific needs.

In this session, we will look at different methodologies that promote agility. We then will compare and contrast the features of each. You can take away from the presentation what makes the most sense for your project and team.

About the Speaker *Dr. Venkat Subramaniam*, founder of Agile Developer, Inc., has trained and mentored thousands of software developers in the US, Canada, and Europe. He has significant experience in architecture, design, and development of software applications. Venkat helps his clients effectively apply and succeed with agile practices on their software projects, and speaks frequently at conferences.

He is also an adjunct faculty at the University of Houston (where he received the 2004 CS department teaching excellence award) and teaches the professional software developer series at Rice University School of continuing studies.

Venkat has been a frequent speaker at No Fluff Just Stuff Software Symposium since Summer 2002.



Agile Methodologies

- **What's Agility?**
- Why Agility?
- Agile Manifesto and Principles
- What's Methodology and why?
- Methodologies that promote agility
- Conclusion

Agility

- What's Agility?
- Being agile
- What's Agile?
- "marked by ready ability to move with quick easy grace"
- "having a quick resourceful and adaptable character"
- What does that mean?
  - Process has to be lightweight and sufficient
  - Lightweight helps us adapt and move
  - Sufficient recognizes our ineffectiveness to be complete and relies on strong communication

Agile Methodologies

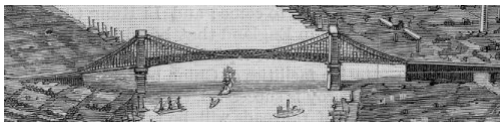
- What's Agility?
- **Why Agility?**
- Agile Manifesto and Principles
- What's Methodology and why?
- Methodologies that promote agility
- Conclusion

Evolution of Fields

- Bridge Construction
- Medicine
- Airplanes
- Software Development

Bridge Construction

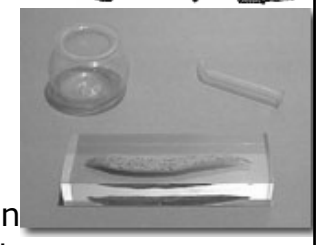
- Early Wood, Stone
- Then Iron, Steel
- Concrete Bridges



- Constructing a bridge is different from innovating a bridge (with new material for instance) for the first time
- Engineers use well established metrics to design bridges – they do not innovate at this stage

Medicine

- "Health was thought to be restored by purging, starving, vomiting or bloodletting"
  - Both surgeons and barbers were specializing in this bloody practice
  - Widely practiced in 18th and 19th century
  - Declared quackery by 1900
- Infection control
  - If patient survived surgery, he most likely died out of infection
  - Germ theory and sterility came only in late 1800s (Lister)
  - Current rate of infection < 2.5%



Airplanes

- 400 BC Chinese fly kite aspiring humans to fly
- For centuries, we tried to fly like birds… disastrous
- Steam powered, hot air
- Gliders, single man
- Engine powered
- 1903 Wright brothers' first flight – 12 seconds, 120 feet, 10 feet altitude

Software Development

- Relatively nascent field in comparison
- Machines are getting faster or more powerful
- Are we getting better in delivering software applications though

Success (or lack there of)

- How successful are we in developing software?
- Less than 10% of software projects succeed[1]
- Criteria for success?: On time, within budget, feature complete, works (failure free)
- Why is it so hard to get this right?

Software Engineering?

- What's Engineering?[2, 3]
  - "the application of science and mathematics by which the properties of matter and the sources of energy in nature are made useful to people"
  - "the design and manufacture of complex products <software *engineering*>"
- If software engineering like manufacturing or designing a manufacturing plant?
  - Is it like making another cell phone or making of cell phones (took 37 years for commercialization)?
- Manufacturing is predictive
  - You can measure and control quality, quantity
- Designing a manufacturing plant is creative/innovative
- Most software development is innovative process rather than predictive manufacturing
  - Requires great deal of innovation, interaction/communication

Why is it hard to communicate?

- Why not simply write good documents to describe requirements and hand them off to developers to create software?
- We have tried that, but we know it does not work
- 3 factors influence
  - What you are communicating
  - Who is communicating
  - With whom

- A Picture is worth a thousand words
- Let's take a look at this picture from Stephen Covey's "7 Habits of Highly Effective People"

Realizing what makes it hard

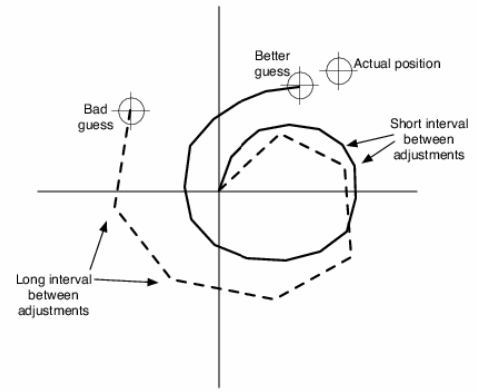
- Ceremony: Amount of method weight for documentation, formal steps, review, …
- Documents can't fully describe the requirements
- 3 types of people make up your team
  - Those with exceptional domain knowledge but little software development expertise
  - Those with exceptional software dev. experience, but little domain knowledge
  - Those with both domain and software development skills
  - (we will ignore that 4th category)
- Closer and frequent interaction is a necessity

Process

- Waterfall approach[4]
  - Actually specified iteration - largely ignored
- Customers' mind is not frozen after they give us the requirements
- We are not able to fully understand what is said
- Show me a long project duration, I will show you a project that is already doomed

Iterative and Incremental

- How to foster innovation and communication?
- Isolation does not help
- Interaction is key
  - among developers and with customers
- But will that not take more time?



The time/scheduling hypocrisy

- What can you tell me about the next project, you ask?
  - It is due on November 1st tells your manager
- We hold deadlines too dearly
- Of course, time to market is critical
- But what generally happens on projects when you hit that deadline?

Pick Two

- Ask your customers to pick two out of the following, you decide the third:

- Time
- Scope
- Quality

- Reality often ignored in project planning

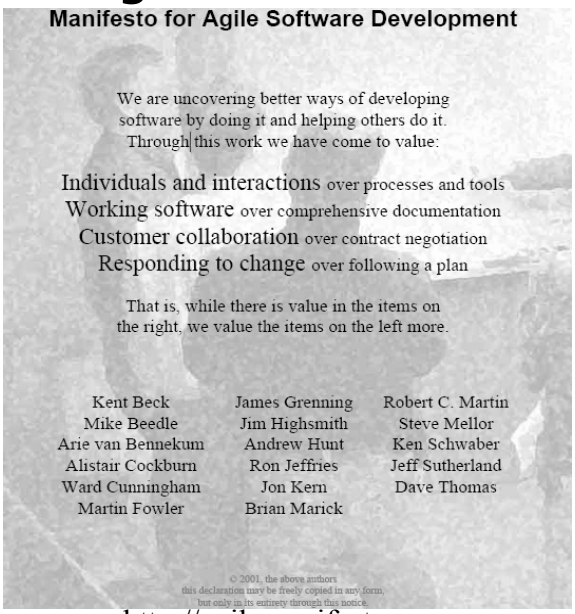
Agile Development Process

- Iterative and evolutionary development
- Timeboxing
  - Set amount of time for iteration
  - Adapt future iteration based on the realities
- Adaptive planning
- Incremental delivery
- Agility
- More focused on success than sticking with a plan
- Working software is valued and considered measure of progress

Agile Methodologies

- What's Agility?
- Why Agility?
- **Agile Manifesto and Principles**
- What's Methodology and why?
- Methodologies that promote agility
- Conclusion

Agile Manifesto



http://agilemanifesto.org

Principles behind the Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity–the art of maximizing the amount of work not done–is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Methodologies

- What's Agility?
- Why Agility?
- Agile Manifesto and Principles
- **What's Methodology and why?**
- Methodologies that promote agility
- Conclusion

Methodology

- Methodology
- It's what you do–whatever that is–to create software
- Series of related methods to coordinate people's activities on a team
- How work is done

Why Methodology?

- Helps to explain how your team works
- Helps us understand responsibilities and priorities
- Helps measure progress and show progress
- Serves as a framework to learn from

Agile Methodologies

- What's Agility?
- Why Agility?
- Agile Manifesto and Principles
- What's Methodology and why?
- **Methodologies that promote agility**
- Conclusion

Methodologies

Methodologies share common principles,
but differ in practices

- eXtreme Programming (XP)
- Scrum
- Evolutionary Project Management (Evo)
- Unified Process (UP)
- Crystal
- Lean Development (LD)
- Adaptive Software Development (ASD)
- Dynamic System Development Method (DSDM)
- Feature Driven Development (FDD)

eXtreme Programming (XP)

http://www.xprogramming.com

XP

- Kent Beck, Ward Cunningham, Ron Jeffries based on experience from C3 project
- XP has nothing new, yet it has something new!
- Four values, Twelve practices
  - Based on what has worked on projects, taking them to extreme
    - If something is good why not do it all the time?
- Small teams (under 20)
- Onsite customer presence
- Planning game
  - Negotiate requirements in form of stories captured on index cards
- 2 to 3 weeks iteration
- Scales well for problem size within limits, but does not scale well for team size
  - But, a competent smaller team is better than a large team following heavier methodologies
- Deemphasizes documentation
  - Accelerates development, but may be a problem for transition later on

Control Variables

- Cost
  - Too little, does not solve problems
  - Too much, some times more of a problem
- Time
  - More time can improve quality and increase scope.
  - Too much time hurts as well
    - Feedback from system in production is imperative
- Quality
  - Sacrificing this may result in short term gains
  - Over the long haul, lost is enormous
- Scope
  - Lesser the scope, better the quality
  - You can deliver sooner as well
  - Assuming it meets the business needs

Set of Values

- Communication
  - Communicate critical change in requirements, design, etc.
  - Put in place practices that will enhance communication
- Simplicity
  - Find simplest thing that will work
  - Build some thing simple today and pay a little to change tomorrow than build some thing complicated today that may never be used
- Feedback
  - Unit tests provide feedback
  - Corrected in minutes and days, not weeks
  - System that stays out of the hands of users is trouble waiting to happen
- Courage
  - Don't hesitate to throw code away if you find better simpler way
  - Do not hesitate to call attention to problems if they are significant and will benefit from reworking

Taking it to extreme

- It takes good commonsense principles and practices to extreme levels
  - If code review is good, we'll review code all the time
    - Pair programming
  - If testing is good, every body will test all the time
    - Unit testing by developers, functional testing by customers
  - If design is good, we'll make it part of everybody's daily business
    - Refactoring
  - If simplicity is good, we'll make it part of the system with simplest design that supports its current functionality
  - If architecture is important, everybody will work defining and refining the architecture all the time
    - metaphor
  - If integration testing is important, then we'll integrate and test several times a day
    - Continuous integration
  - If short iterations are good, we'll make the iterations really, really short – seconds and minutes and minutes and hours, not weeks and months and years
    - The planning game

XP Principles

- The Planning Game
  - Scope next release with business priorities, technical estimates
  - Update the plan based on reality
- Small Releases
  - Put simply system into production quickly
  - Release new version in short cycle
- Metaphor
  - Guide development with simple shared story of how the whole system works
- Simple Design
  - Design as simple as possible at any given moment.
- Testing
  - Continually write and run unit tests

XP Principles

- Refactoring
  - Restructure system without changing its behavior to remove duplication, improve communication, add flexibility and simplify
- Pair Programming
  - Two programmers, one machine, four eyes are better than two
- Collective Ownership
  - Anyone can change code anywhere in the system at any time
- Continuous Integration
  - Integrate and build the system many times a day, every time a talk is completed.
- 40-hour Week
  - Never work overtime a second week in a row
- On-site Customer
  - Real, live user on the team, available full-time to answer questions
- Coding Standards
  - All code written accordance with rules emphasizing communication through the code

Where does XP work?

- Culture
  - Business culture
    - How change is accepted? Need to work long hours? Goal oriented? Heavy on paper work?
- Size
  - Team size of around 10 is ideal
- Technology
  - Must be able to make change quickly and get feedback
- Work environment
  - Should promote closer interaction and communication

Scrum

http://en.wikipedia.org/wiki/Image:Rugby_union_scrummage.jpg

http://www.controlchaos.com

Scrum

- Developed by Ken Schwaber and Jeff Sutherland
- Name derived from Rugby
  - Groups effort to move quickly to counter the opposite team, adjusting the move along progress
- Scrum Master – coach for the team
  - Looks outward keeping distractions out
  - Trusts the self-managed team to get work done
- Sprint – 30 days iteration cycle with pre-sprint and post-sprint activities
- Scrum meeting – Short standup meeting to communicate and monitor progress
- Backlog – used for planning
  - Features and estimate of duration for each task
  - Task for sprint picked from the pool of tasks
  - Used to decide features for sprint and plan out the work
- Sprint Goal – minimum success criterion to steer and keep focus

Scrum…

- Leaves documentation depth to specifics of projects – may need more or less
  - aim for as little as possible
- Self-directed and self-organized team
  - Competent focused people
- Demo to stakeholder at iteration end
- Client-driven adaptive planning
- No work added during iteration
- Lends itself to experimenting on certain parts of the application development

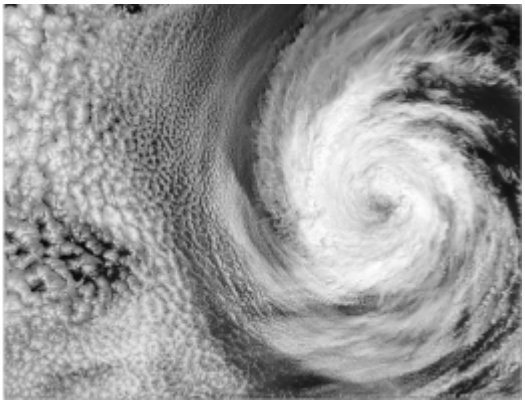
Scrum Lifecycle

- Planning
  - Vision, expectations, funding
- Staging
  - Identify requirements, prioritize iteration
- Development
  - Implement system ready for release in each sprint
- Release
  - Operational deployment

Scrum Values

- Commitment
  - Team takes responsibility to complete the Sprint. To avoid things that will stand in its way
- Focus
  - Team's focus is maintained. Distractions, interruptions are fielded
- Openness
  - Overall and individual status and commitments kept open.
- Respect
  - Team responsibility rather than scapegoating.
- Courage
  - Management and team have the courage to take responsibility to do what is necessary

Evolutionary Project Management (Evo)



http://www.gilb.com

http://www.gilb.com

Evo

- Oldest iterative and incremental method
  - introduced in 1960s by Tom Gilb, published 1976
- Short (5 days) iteration
- Evolutionary requirements and design
- Recommends measurable short list of project objectives
- Avoids big up-front specification
  - Evolving requirements
- Recommends use of a Planguage – a specification language that could make it ceremonial
- Emphasizes measurable progress
- Frequent delivery to stakeholders

Unified Process (UP)

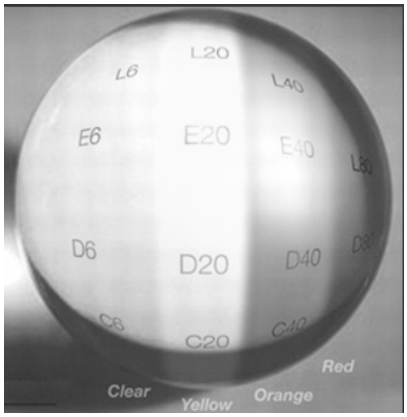
Rational Unified Process

http://www-306.ibm.com/software/awdtools/rup

UP

- Developed by Rational Software Corp (now part of IBM), lead by 3 amigos
  - Grady Booch, James Rumbaugh, Ivar Jacobson
- Derived from several methodologies at that time
- Micro and Macro development process
- Micro deals with tactical issues (daily activities)
- Macro process has inception, elaboration, construction, and transition
- Generally viewed as heavy weight process
- Agile in sprit, but can get very ceremonial
  - Emphasizes iterative cycles, constant feedback
  - Developed along with UML which provides for several forms of documentation
- Comes from disciplined process oriented angle
- Not easy to tailor for small projects

Crystal



http://alistair.cockburn.us

http://alistair.cockburn.us

Crystal Family

- Alistair Cockburn
- Framework of related methods addressing variability of environment and specific characteristics of projects
  - Size of development team
  - Project criticality
    - Loss - due to defect - of comfort, essential money, discretionary money, life
- Crystal a metaphor for color and hardness
  - Clear, yellow, orange, red
- People and Communication centric
- Lighter (color) is better as long as it lasts
  - See/show significant consequence or risk before implementing a harder/darker version
- Project specific methodologies

Core Properties

- Frequent delivery/integration using time-boxed iterations
- Reflect and improve, criticize and fix
- Osmotic (passive) knowledge acquisition and communication through office organization and open channels
- Personal Safety, safe to be honest, confidence to court criticism
- Stay focused, clear tasks, priorities on work, limit the workload
- Access to expert users, fast, quality feedback
- The usual agile stuff: automated testing, CM, continuous integration

Lean Development (LD)

http://www.itabhi.com/ld.htm

http://www.itabhi.com/ld.htm

LD

- Developed by Robert Charette based on lean manufacturing - proprietary
- Risk entrepreneurship turn risk into opportunity
- Phases:
  - Startup
    - Planning, business cases, feasibility studies
  - steady state
    - Series of short spirals
  - transition-renewal
    - Doc developed and delivered
- More business strategies and project management approach
  - Involves everyone, not just developers
  - Focused on accessing and achieving business value
- LD is "strategic, business-down approach whereas most agile approaches are tactical, program team-oriented in nature."

LD Principles

- Satisfying the customer is the highest priority of the organization
- Always provide the best value for money
- Success depends on active customer participation
- Every lean development is a team effort
- Everything is changeable
- Domain, not point solutions
- Complete, don't construct
- Minimalism is essential
- Needs determine technology
- Product growth is feature growth, not size growth
- Never push lean development beyond its limits

Adaptive Software Development

http://www.adaptivesd.com

ASD

- Developed by Jim Highsmith and Sam Bayer based on rapid application development (RAD)
- Emphasizes continuous adaptation of the process
- Speculate, Collaborate, and learn cycles
- Continuous learning and adaptation as project emerges
- Mission focused, feature based, iterative, timeboxed, risk driven, and change tolerant
- Non prescriptive in nature – not much on how to do things, more of opportunities to take to meet the goal

Dynamic Systems Development Method (DSDM)

http://www.dsdm.org

DSDM

- Developed by DSDM consortium
  - Mostly European
- Five phases: feasibility, business study, functional model iteration, design and build iteration, implementation
- Strong emphasis for project management activities
- 10 project roles (a person may play more than one role)
- Plans evolve based on increments
- Timeboxing means for planning, monitoring, controlling
- Prioritized using MoSCow
  - Must have, Should have, Could have, Want
- Designed for small teams, but scales up

Feature Driven Development (FDD)

http://www.nebulon.com/fdd

http://www.nebulon.com/fdd

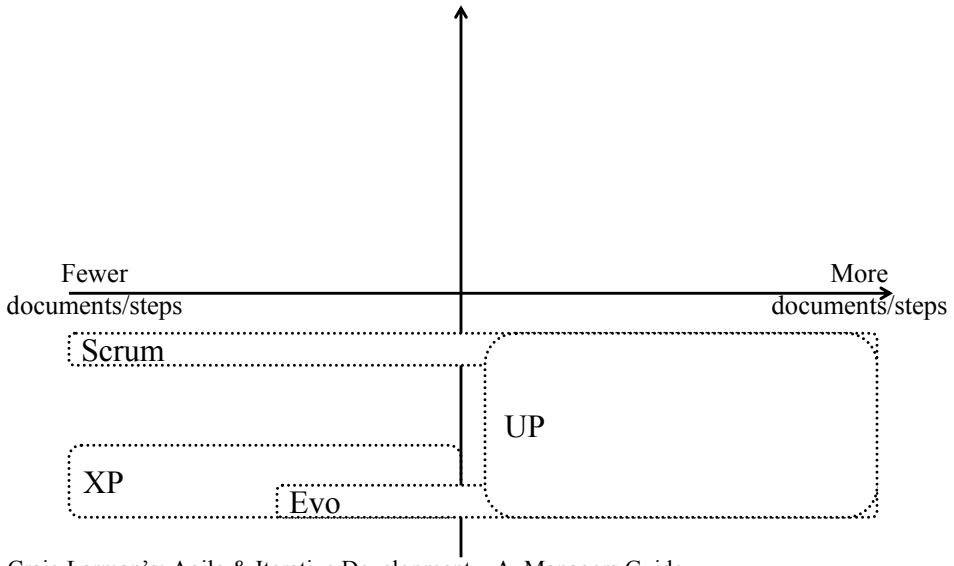
FDD

- Jeff DeLuca and Peter Coad
- Simple process, modeling, short iteration cycle
- Good people for domain knowledge, design, and development
- Expects requirements to be well captured and understood
- Expects classes to be assigned to individuals
- Emphases on getting the architecture right
- Suitable for stable systems with predictable evolution

Quick Comparison

- Ceremony and Iteration
- Competency Level Expectations
- Emphasis

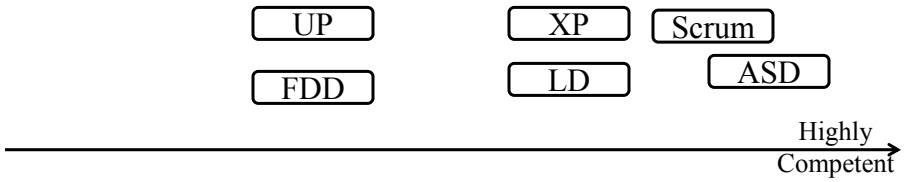
Ceremony and Iteration



Craig Larman's: Agile & Iterative Development – A Managers Guide

Competency Level Expectations

- Cockburn Level's: following (level1), detaching (level2), fluent (level3)
- The Dreyfus Model of Skills Acquisition (level 1 to 5) [Herding racehorses and racing sheep–Dave Thomas]



	XP	Scrum	Evo	UP	Crystal
Iteration	2/3 weeks	30 days	5 days	Short	Short
Customer Participation	Strong	Strong	Strong	Reco	Strong
TDD	Strong	Strong			Reco
Business Process	Medium	Medium		High	Med/High
Team Size	Small	Small		Med/large	Varies
Ceremonial	Low	Flexible	Med/high	High	Varies
Feature *Emphasis*	Pair Prog, Collective ownership, strong customer participation, constant refactoring, 40 hours work week	Scrum meeting, Timeboxing, Backlog, Self directed team, open communi-cation	Customer driven, frequent delivery, short iteration, one of the first		Varies based on criticality, high focus, domain expert presence

Quiz Time

Agile Methodologies

- What's Agility?
- Why Agility?
- Agile Manifesto and Principles
- What's Methodology and why?
- Methodologies that promote agility
- **Conclusion**

References

1. "Software Project Management Practices: Failure Versus Success," Capers Jones (http://www.stsc.hill.af.mil/crosstalk/2004/10/0410Jones.html)
2. "Agile Software Development," Alister Cockburn, Addison-Wesley.
3. "Agile and Iterative Development: A Manager's Guide," Craig Larman, Addison-Wesley.
4. "Iterative and Incremental Development: A Brief History," Craig Larman, IEEE Computer, June 2003.
5. "Planning Extreme Programming," Kent Beck, Martin Fowler, Addison-Wesley.
6. "Agile Software Development, Principles, Patterns, and Practices," by Robert C. Martin, Prentice Hall.
7. "Agile Software Development with SCRUM," Ken Schwaber, Mike Beedle, Prentice Hall.
8. "Information Radiator," http://c2.com/cgi-bin/wiki?InformationRadiator.
9. "Test Driven Development: By Example," Kent Beck, Addison-Wesley.
10. "Pragmatic Unit Testing in Java with JUnit," Andy Hunt, Dave Thomas, Pragmatic Programmers.
11. "Refactoring: Improving the Design of Existing Code," Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts, Addison-Wesley.
12. "Continuous Integration," Martin Fowler, Matthew Foemmel, http://www.martinfowler.com/articles/continuousIntegration.html.
13. "Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Apps," Mike Clark, Pragmatic Programmers.
14. "Continuous Integration Server Feature Matrix," http://docs.codehaus.org/display/DAMAGECONTROL/Continuous+Integration+Server+Feature+Matrix.
15. "The Pragmatic Programmer: From Journeyman to Master," Andrew Hunt, David Thomas, Addison-Wesley.
16. Some interesting articles to read - http://tinyurl.com/drnor

Download slides and More…

Download examples/slides from

http://www.agiledeveloper.com/download.aspx

Please fill out your evaluations!