CÚRAM SOFTWARE

Distributed Agile Development

Bapiraju Nandury
Product Development Manager
Bangalore Development Centre

Agenda

- Distributed / offshore Development
- Agile Methods
- Distributed Agile Development

Goals of this presentation

- **Assumption**
  - Knowledge of software development process
  - Open to ideas and emerging methodologies
- **At the end of this session, you should get an overview of**
  - Distributed agile development
  - Challenges posed by distributed agile projects
  - Possible remedies/guidelines

Overview

Distributed Development

Definition
Advantages

Agile Development

Agile Manifesto
Values
Principles
Development Process
Roles
Agile Umbrella

Distributed Agile Development

Why distributed agile?
Who likes it?
Key challenges
Critical Success Factors
Why does it work?
Questions to be answered

Context

- **Software Development Teams**
  - Collocated
  - Geographically distributed
- **Processes**
  - Conventional
  - Agile Methods
- **Focus of this presentation**
  - Distributed Development
    - *Conventional*
    - *Agile*

Distributed/offshore Development

What is Distributed Development?

- **A distributed development project is defined as a group of members actively collaborating on a common software/ systems project separated by**
  - Distance
  - Time zone
  - Culture
- **What makes distributed development interesting?**
  - Specialized Talent
  - Cost differentials
  - Advantage of “Follow-the-Sun” approach
  - Global Presence
  - Virtual Organization etc.

Agile Methodologies

Agile Manifesto

We* are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions*
- *Working software*
- *Customer collaboration*
- *Responding to change*

OVER

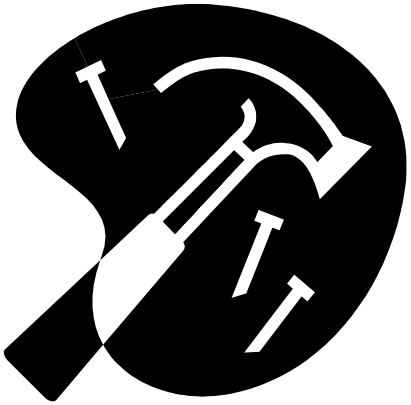
- *Processes and tools*
- *Comprehensive documentation*
- *Contract negotiation*
- *Following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

*Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, Andrew Hunt

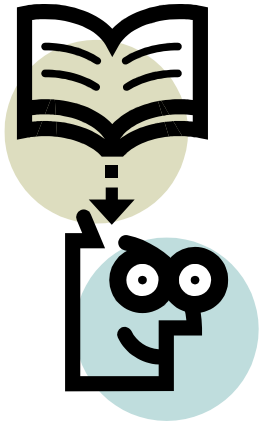
Value #1

- Individuals and interactions over processes and tools



Value #2

- **Working software over comprehensive documentation**

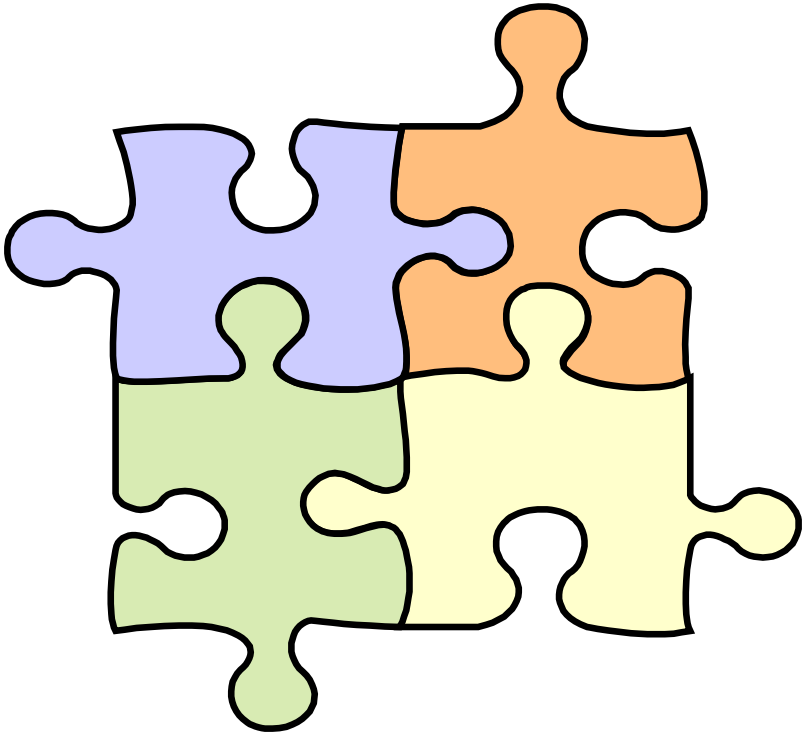


Value #3

- Customer collaboration over contract negotiation

Value #4

- Responding to change over following a plan



Agile Principles

Agile Principles

- 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

Agile Principles

- 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

Agile Principles

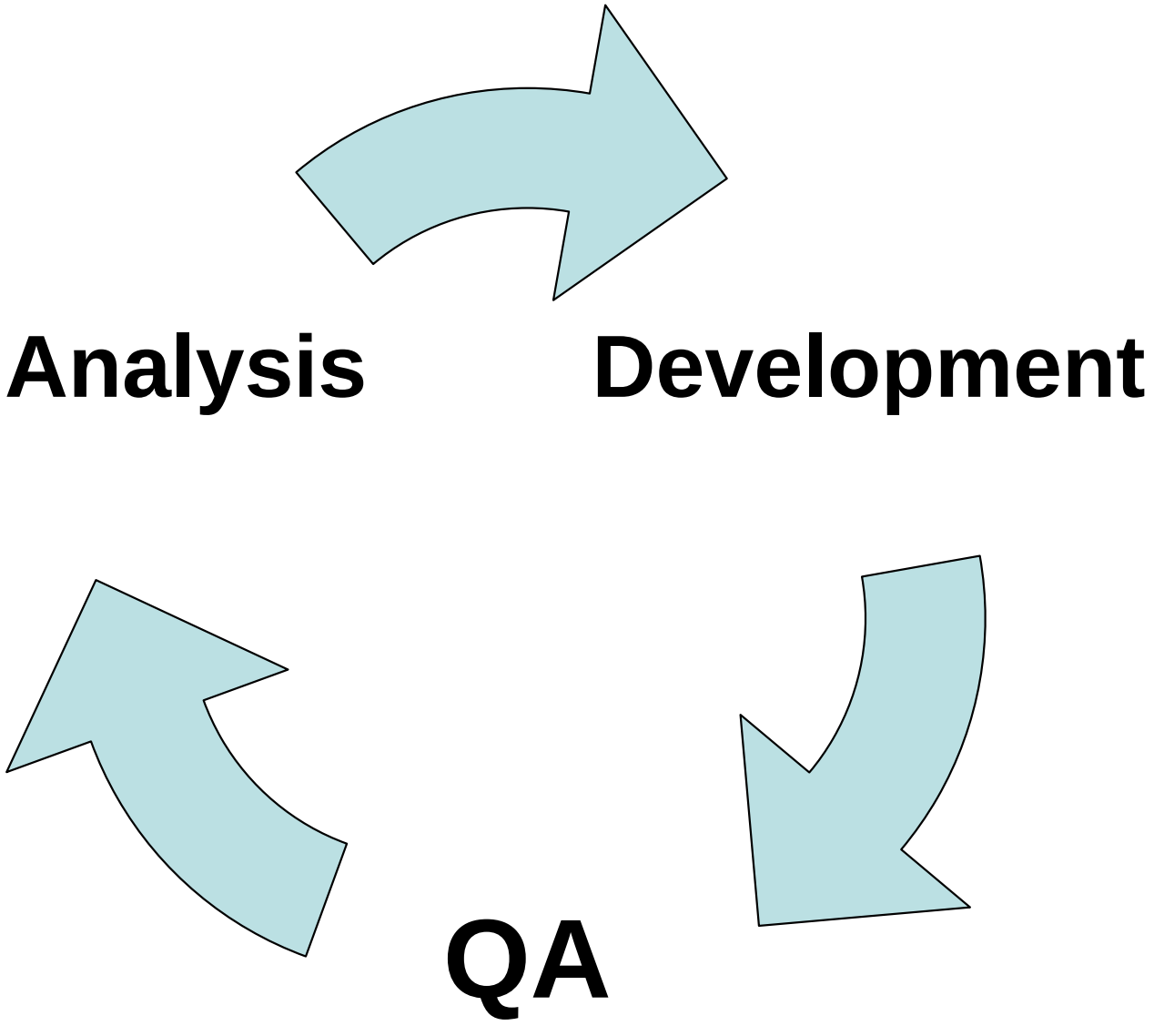
- Working software is the primary measure of progress
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- Continuous attention to technical excellence and good design enhances agility
- Simplicity – the art of maximizing the amount of work not done – is essential

Agile Principles

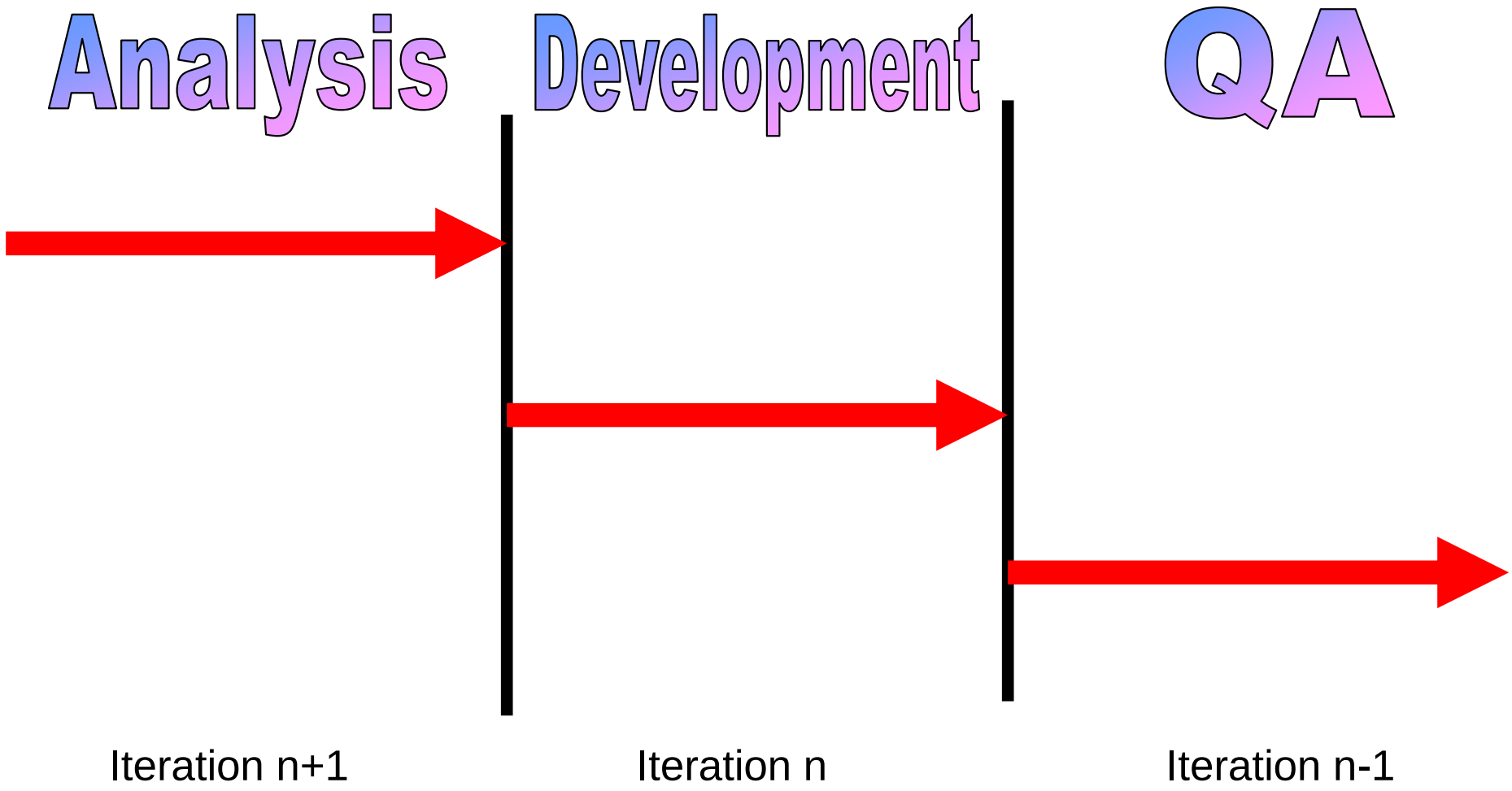
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Agile Development Process

Agile Development Process



Agile Development Process



Agile Development Process

- **Roles on Agile Teams**
  - Business Analyst aka Proxy Customer
  - Developer
  - Quality Analyst
  - Database Administrator
  - Build Master
  - Iteration Manager
  - Project Manager etc.

Agile Development Process

- **Agile Umbrella**
  - eXtreme Programming (XP)
  - SCRUM
  - Feature Driven Development
  - Crystal
  - DSDM (Dynamic System Development Method) etc

Distributed Agile Development

Distributed Agile Development

- **Who likes it?**
  - Business – sees value as the project progresses
  - Development teams – iterative development and continuous integration
- **May not be for all types of projects**
  - Identify what can be built offshore

What is it different?

- **Quality standards are harder to reach**
  - In a co-located team, you can reach quality standards by pair programming, coding standards
  - Knowledge diffuses easily by spontaneous communication, you can build shared values just by walking around and talking
- **In a distributed setting it's harder:**
  - less identification with the project
  - less identification with the employer
  - less responsibility, and may be less team work

Application of agile to offshore

- **How can agile make it better?**
  - Do collocated agile first, then distribute it. It takes time to learn this new practice, as it is different from the traditional model
- **Allow several weeks for getting used to self-organization**
  - If you're used to "command & control"
    - *you want detailed work lists*
    - *you don't estimate yourself*
    - *you want to be left alone for some time*
    - *you don't want to give others visibility into your way of working*
    - *Command and control hides a lot of things and fosters the "blame game". If something is not how you like it to be, you just do whatever you like and blame someone else*

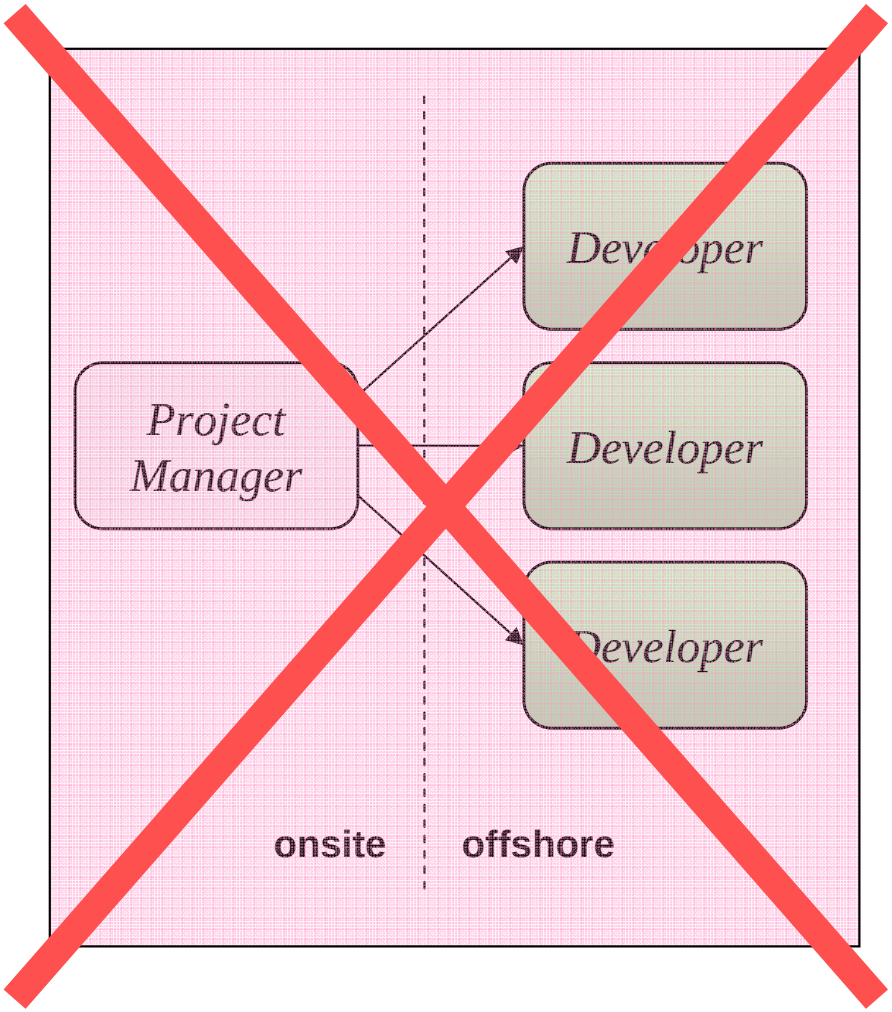
Faculties of Distribute Agile

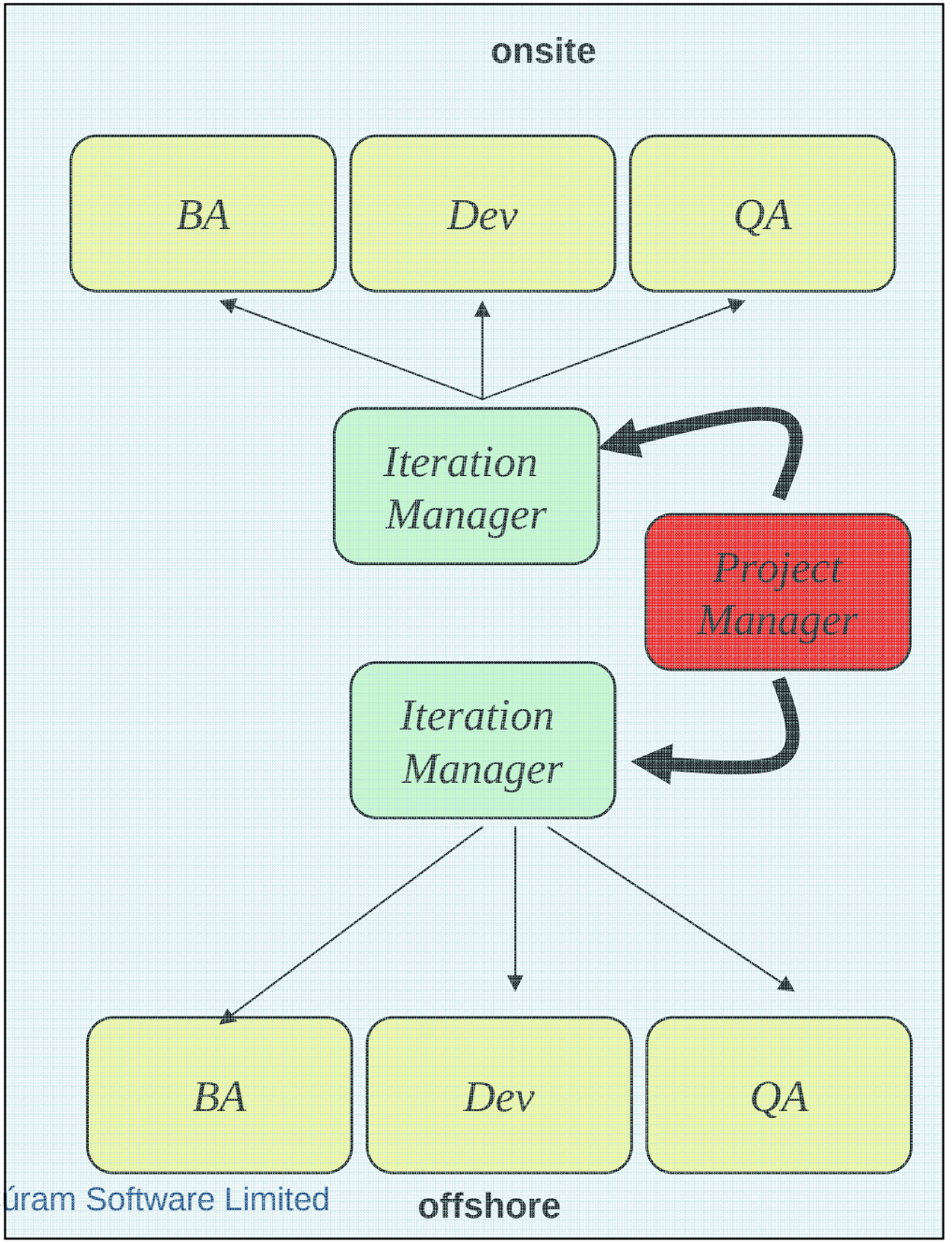
- Management
- Planning
- Analysis and Design
- Development
- Testing
- Configuration Management etc

Management

- **Role of Project Managers**
  - No Micro Management
  - Project schedules, task breakdowns, work assignments, and progress reports
  - Organize teams by functionality, but not by activity
- **Role of Iteration Managers**
  - Run iterations effectively
  - Live on the floor
  - Create excitement, enthusiasm in the team
  - Facilitate among various stakeholders
  - Communicate, communicate and over communicate

Management





Planning

- **Release Level Plans**
  - Master Story List
  - Split into iterations
  - Identify stakeholders
  - Release level estimation
  - Establish timelines and release dates
  - Short releases
- **Iteration level plans**
  - 2 week Iterations
  - Iteration Planning Meeting
  - Elaborated requirements / story cards
  - Estimates
  - Accommodate changes
  - Manage Hangover cards
  - Manage cards being pushed to subsequent iterations

Analysis and Design

- Requirements Management: formal requirement specifications, dependencies, responsibilities, and change history
- Involvement of development resources in Design to provide continuity into Development or disciplined empowerment on Design
- Business Analysts – Proxy Customers
- Pairing on Story Cards
- Functional Tests
- Feedback to Customers

Development

- Source Code Management: versioned file assets such as design documents, source code, and binary files
- Test-first approach
- Continuous integration across multiple locations
- Senior – Junior pairing
- Pairing, not just pair programming
- Explicit communication
- Progress check via daily stand ups / conference calls
- Refactoring

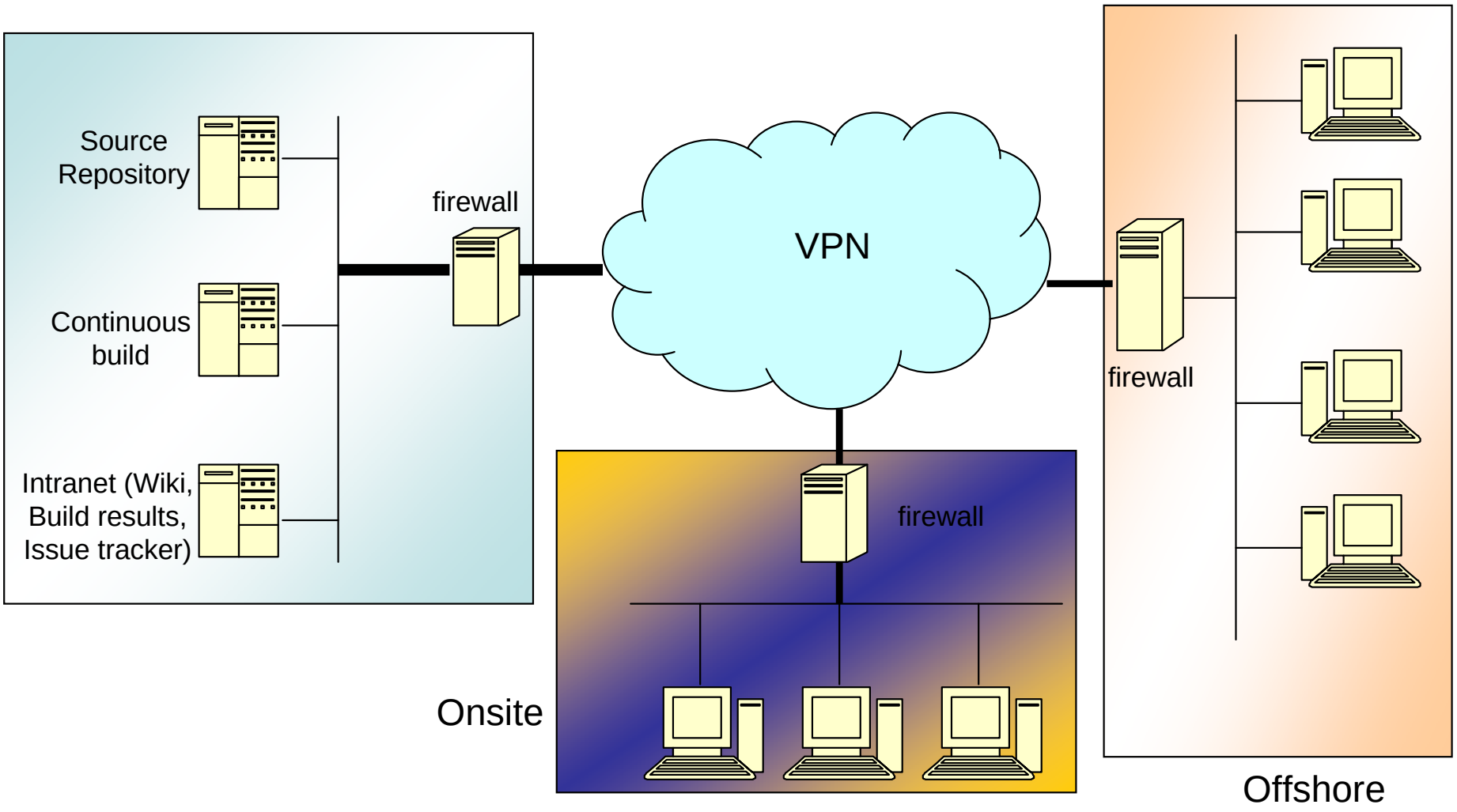
Testing

- **Use (automated) test cases to specify requirements**
  - Written specifications can hide details
  - Test cases make the details and the flow of events (dynamics) explicit.
- **Write functional test cases before development starts**
  - Helps transfer business knowledge
  - Easy way to know requirements have been understood
  - Can then be scripted/automated by testing team
- **Functional Testing**
  - FIT : define acceptance tests in HTML tables
  - FitNess : tests defined in a Wiki
  - Selenium : web testing inside the browser

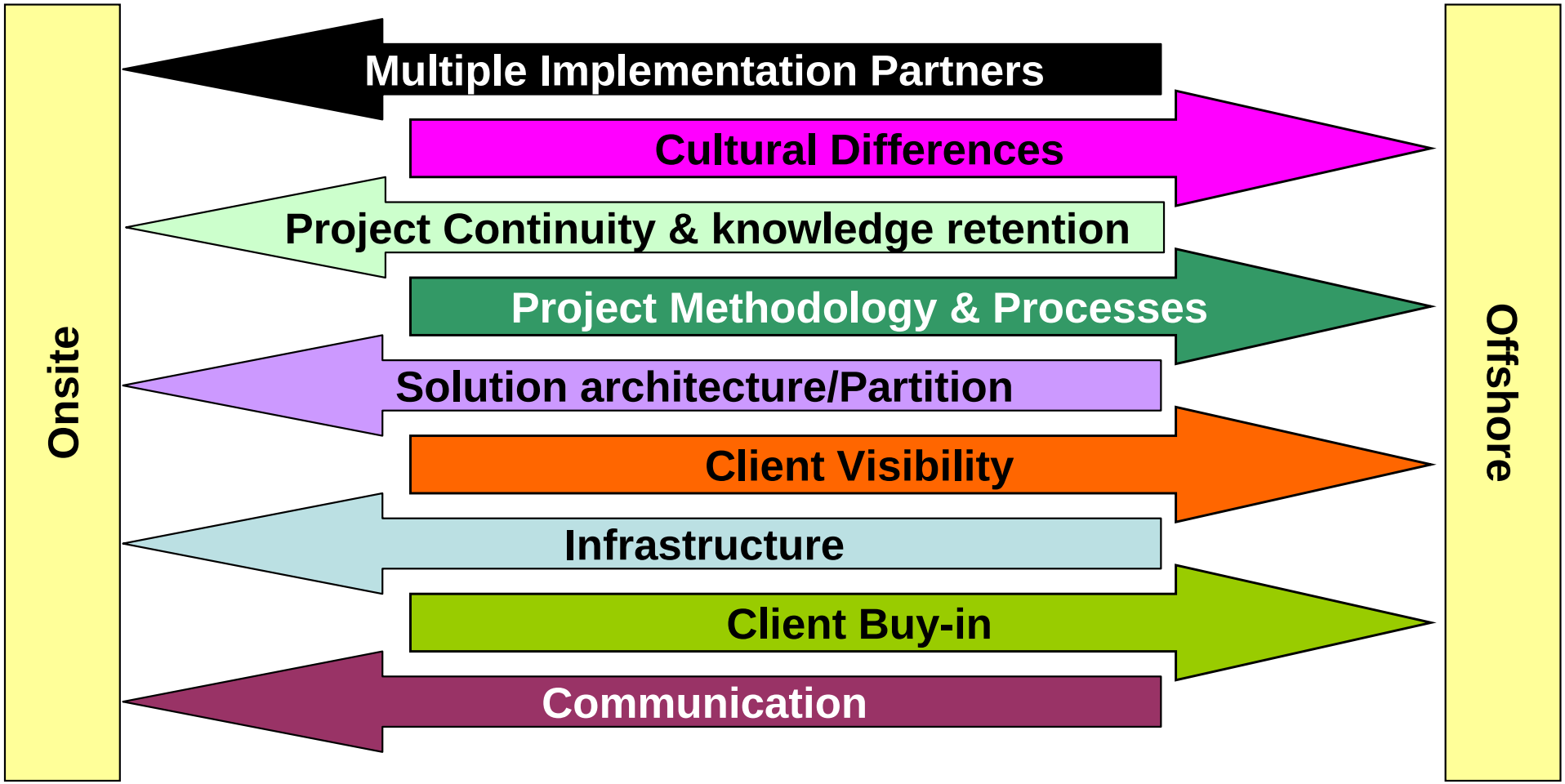
Configuration Management

- **Use common tools:**
  - Integrated development environment (e.g. eclipse)
  - Source Code Repository (e.g. CVS)
  - Build environment (e.g. Maven)
  - Unit test framework (e.g. Junit)
  - Acceptance test framework (e.g. FIT)
  - Centralized build server (e.g. CruiseControl)
  - Bug tracking system (e.g. Bugzilla)
- **Integrate early and often**
- **Automate builds**
- **Use Continuous Integration**
- **Environment identical at both places**

Infrastructure



Key Challenges



Key Challenges

- **Client Buy-in**
  - Developing a partnership with the client in delivering the geographically distributed projects
- **Project Continuity and knowledge retention**
  - Knowledge of the earlier phase (if any) or specifications
- **Solution architecture/Partition**
  - Selection of the appropriate multi-site model, for otherwise it could lead to a failure in distributed approach

Key Challenges

- **Defined Deliverables**
  - It is the most important factor for successful delivery. If the deliverables are not clear then the Project could very well go into a whirlpool
- **Formalized Tracking**
  - Accurate documentation and continuous tracking of issues/assumptions and communication threads

Key Challenges

- **Project Methodology & Processes**
  - Issue resolution
  - Communication
  - Change management
  - Quality measures
  - Standards
  - Working times
  - Team roles and responsibilities
  - Project KPI's
  - Project time reporting
  - Expenses and similar other logistical activities
  - Development & testing approach

Key Challenges

- **Client Visibility**
  - Use short iterations
  - Show prototypes to user
  - Deliver software frequently
  - Build up common understanding of what is necessary for shipping software
    - Kinds of tests
    - Levels of quality
  - Have customer participate in Daily Status Meetings
  - Show Progress Charts publicly
  - Use Test-Driven Development At the Acceptance-Test Level

Key Challenges

- **Communication**
  - Communication issues because of varied time zones, dispersed teams and cultural issues
  - Expect a lot of difficulty understanding each other
    - English may be nobody's native tongue
    - We usually don't say "no", and politeness can hide misunderstanding
  - Use direct communication and instant messaging rather than emails
    - Avoid the misunderstanding
    - Reach consensus, check whether the other parties really have understood what you mean, check again and again
    - Avoid the time lag

Success Lessons

- **Require good mediators/coordinators**
  - To solve communication issues due to language and cultural differences
  - To suggest improvements continuously
  - To act as “neutralizers”
- **Lots of discipline**
  - The Build Must Pass! (requires developing a build culture)
  - Weekly meetings
  - Issues trackers and training
- **Dedicated offshore support persons in each team to minimize question round trips**

Success Lessons

- **Share activities between onsite/offshore**
  - Not just development, but also Business Conception, Detailed Design, Testing, etc
  - Improve team spirit/job satisfaction
  - Allow offshore people to progress
  - Share the global knowledge and thus improve efficiency
- **Have Project Leads do the business conception**
- **Have Project Leads do the detailed design**
- **Ideally, send functional tests along with use cases**
  - Have a separate team script and automate them

Success Lessons

- **Cross Pollination**
  - Plan in place
  - Both ways – works better
  - Long term projects, plan rotation
- **Lots of travels/exchanges**
  - Hard to work with someone you haven't seen and leads to misunderstandings
  - Help share knowledge at all levels
    - Managers, Project Leads, Developers

Success Lessons

- **Estimates**
  - Have the developers select and estimate their tasks.
  - Have the developers update their estimates regularly (e.g. every day at Stand up).
  - Post charts of status / effort remaining publicly.
- **Communication**
  - Use Collaborative Tools
    - Phone, Wiki, Email, Chat
    - Threaded discussions, Web Conferencing etc
  - Have Daily Status Meetings (“Daily Stand-ups”)

Success Lessons

- **Seeing is Believing**
  - Release Plan Walls
  - Story Card Walls
  - Iteration Management Walls
  - Retrospectives
  - Colorful Post Cards
  - Build Tree etc

Why does it work?

- **Today the main driver is cost…**
- **Exchange of knowledge in both directions**
  - On how to develop software
  - On cultures / languages
- **Broadening of vision for all participants**
- **Key skill for participants: working in an international team**
- **Organization are more and more distributed**
  - Will require practices for working from a distance
    - Ex: Mars/Earth
- **Ability to find talents/skills**
  - Ex: Open Source (pushed to the extreme)

Questions to be answered

- What is the best cross pollination strategy?
- What is the best induction program for new comers on large and complex projects?
- How best to implement a light weight issue tracking system?
- How to keep the story cards up-to-date when stories change?
- Pairing between onsite – offshore teams
- How can we make client sit with developers at the offshore location?

blah blah blah ….

Summary

- Distributed Development has its own challenges
- Agile principles, practices, tools can help to address the issues of distributed development.
- There are a lot of lessons and experiences with Distributed Development and Offshore Development, so you don‘t have to take all the pitfalls yourself.

References

- Bailey, P. M. (2004). A formula for successful peer reviews. *Better Software*
- Bailey, P. M. (2005). Creative license. *Better Software*
- Bossavit, L. (2003) Information radiator. http://www.ayeconference.com/wiki/scribble.cgi?read=InformationRadiator
- Brown, J. S. & Gray, E. S. (1995). The people are the company. *Fast Company*
- Clark, M. (2004). Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Apps. Raliegh, NC: Pragmatic Bookshelf
- Clark, M. (2004b). Bubble, bubble, build's in trouble. http://www.pragmaticautomation.com/cgi-bin/pragauto.cgi/Monitor/Devices/BubbleBubbleBuildsInTrouble.rdoc

References

- http://www.agilemanifesto.org/
- Distributed Agile, Dr. Christoph Steindl
- http://www.agilealliance.org/articles
- http://www.martinfowler.com/articles/newMethodology.html
- And various other sources from internet ☺

Please write to me:

bnandury@curamsoftware.com