



TCBAC
twin cities business analyst community

BA Meets Agile:
*The changing role of a
Business Analyst on an Agile project*

April 3rd, 2008

5:30pm Registration and Networking

6:00pm Welcome and Presentation

7:00pm Questions and Wrap-up

Sponsored by:

Solutia Consulting Inc. and Elite Software Architects Inc.

Please do not distribute this information

BA's meet Agile

*The changing role of a Business Analyst on
an Agile project*

John Dubchak

April 3, 2008



elite

SOFTWARE ARCHITECTS INC.

Excellence in IT Consulting, Outsourcing and Offshoring

About Me...

- Enterprise Application Architect
- Agile and TDD Trainer, Coach and Mentor
- 15 years development experience in Java, C++ and .NET
- Have worked in both Agile and Non-Agile projects
- Agile Manifesto Signatory

What we'll cover

- Assumptions I make about this audience
- Methodologies - Past, Present and Future
- The BA role
- Agile, What?
- Agile, How?
- Key success factors for Agile
- Wrap up and Questions

Assumptions

- Extract, collect, elicit, curate, report on requirements
- Work with Customers and/or end users or other stakeholders
- Drive consensus with stakeholders on the scope and features of the system

Assumptions, Part II

- Produce some sort of document describing what the system, ***ideally***, should do
- The product of the BA is a document is used as input by developers in order to construct a system that someone is interested in to solve a business problem

What's the point?

- The focus of any methodology, for a BA, is to produce something that developers can use in order to design and construct a software system

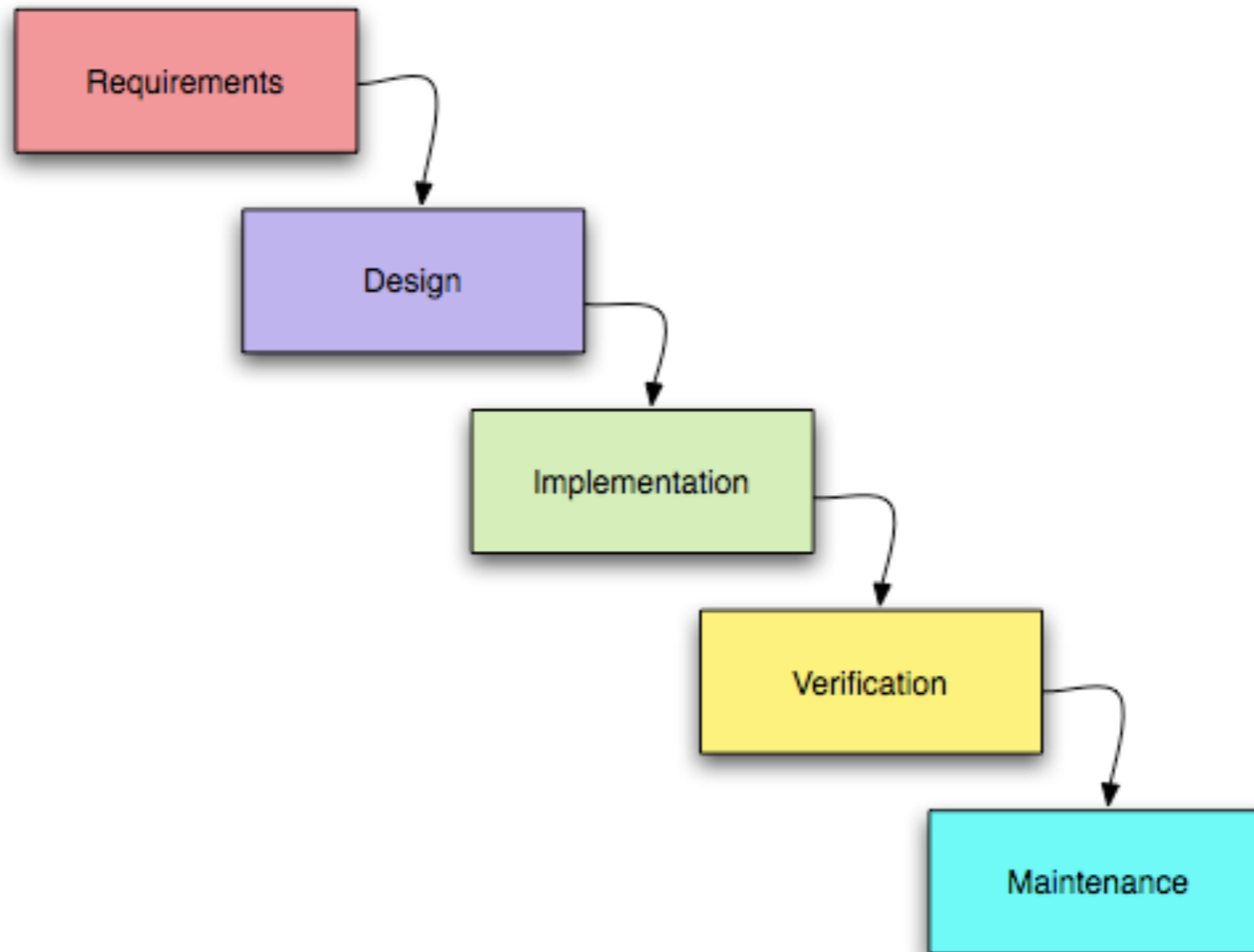
Taking a Step Back

A Quick overview of current methodologies

Software Development Lifecycle - SDLC

- Developed in 1970 by Dr Winston W. Royce
- Consists of the following phases that are expressly followed in order:
 - Requirements Specification
 - Design
 - Construction (Implementation or Coding)
 - Integration
 - Testing and Debugging

SDLC - A Visual



So what is the output
of a BA in the SDLC?

Functional Requirements Specification (FRS)

Outline of an FRS

I Overview

- 1.1 Purpose of this document
- 1.2 Scope of this document
- 1.3 Document Overview
- 1.4 Identification
- 1.5 Relationship to Other Plans
- 1.6 Related Documents
- 1.7 Key Stakeholders
- 1.8 Points of Contact
- 1.9 Traceability

Outline, Section 2

2 Current System

- 2.1 Background
- 2.2 Application Overview
- 2.3 System Objectives
- 2.4 Current Methods and Procedures
 - 2.4.1 Equipment
 - 2.4.2 Input and Output
 - 2.4.3 Provisions
 - 2.4.4 Deficiencies
- 2.5 Business Context
- 2.6 Organization Profile
- 2.7 Business Functions
- 2.8 Component Description

Outline, Section 3

3 Requirements Specifications

- 3.1 Introduction
 - 3.1.1 Goals
 - 3.1.2 System Users
 - 3.1.3 Assumptions
- 3.2 System Description
 - 3.2.1 System Overview and Environment
 - 3.2.2 Functional Structure and Inter-relationships

Outline, Section 3

- 3.3 Functional Requirements
- 3.4 User Roles
- 3.5 System Operational Requirements
- 3.6 Input and Output Requirements
- 3.7 Performance Requirements
- 3.8 Communication Requirements
- 3.9 Communications Requirements
 - 3.9.1 Communications Overview
 - 3.9.2 Communications Hardware
 - 3.9.3 Communications Software

Outline, Section 3

3.10 Security Requirements

3.11 Hardware Requirements

- 3.11.1 Hardware Functionality
- 3.11.2 Hardware Characteristics

3.12 Software Requirements

- 3.12.1 Software Functionality
- 3.12.2 Software Characteristics

Outline, Section 3

3.13 Usability Requirements

3.14 Data Requirements

- 3.14.1 Data Structures and Relationships
- 3.14.2 Data Framework & Relationships
- 3.14.3 Data Inputs
- 3.14.4 Data Outputs
- 3.14.5 Inter-functional Data Definitions
- 3.14.6 Component Cross Reference

3.15 Functional Component Specifications

Outline, Section 4

4 **Proposed Methods and Procedures**

4.1 Improvements

- 4.1.1 Functional Improvements
- 4.1.2 Improvements to Existing Capabilities
- 4.1.3 Timeliness

4.2 Impacts

- 4.2.1 User Organizational Impacts
- 4.2.2 User Operational Impacts
- 4.2.3 User Developmental Impacts

4.3 Product Functions

4.4 Similar System Information

4.5 User Characteristics

4.6 User Problem Statement

4.7 User Objectives

Outline, Section 5

5 Design Constraints

5.1 Software Design Constraints

- 5.1.1 Software Interfaces
- 5.1.2 Software Packages
- 5.1.3 Database
- 5.1.4 Operating System
- 5.1.5 Tolerance, Margins and Contingency

5.2 Hardware Design Constraints

- 5.2.1 Hardware Requirements and Environment
- 5.2.2 Hardware Standards
- 5.2.3 Hardware Interfaces
- 5.2.4 Capacity

5.3 User Interface Constraints

- 5.3.1 User Characteristics
- 5.3.2 Environment/Operational Constraints

Outline, Section 6

6 Detailed Characteristics

6.1 System Description

6.2 System Functions

6.3 Flexibility

6.4 Performance Requirements

- 6.4.1 Accuracy
- 6.4.2 Timing
- 6.4.3 Capacity Limits

6.5 Functional Area System Functions

6.6 Input and Output

6.7 Failure Contingencies

OK already!

Outline, the rest

7 Functional Requirement [x]

8 Resources

- 8.1 Personnel Requirements

9 Appendixes

- 9.1 Support Material
- 9.2 Glossary of Terms
- 9.3 Acronyms and Abbreviations

Index of Tables

- Table 1 — Functional Requirements Matrix
- Table 2 — User Roles
- Table 3 — Resources
- Table 4 — Glossary of Terms
- Table 5 — Acronyms and Abbreviations

SDLC - Wrap up

- A highly structured and prescriptive approach to building a software system
- Projects undertaken with this process failed far more often than not, any idea why?

Unified Process

- An adaptation of the SDLC
- Tackles architecture and addresses riskiest parts first
- A highly compressed cycle with upfront stakeholder involvement
- Abbreviated as Iterative and Incremental Development

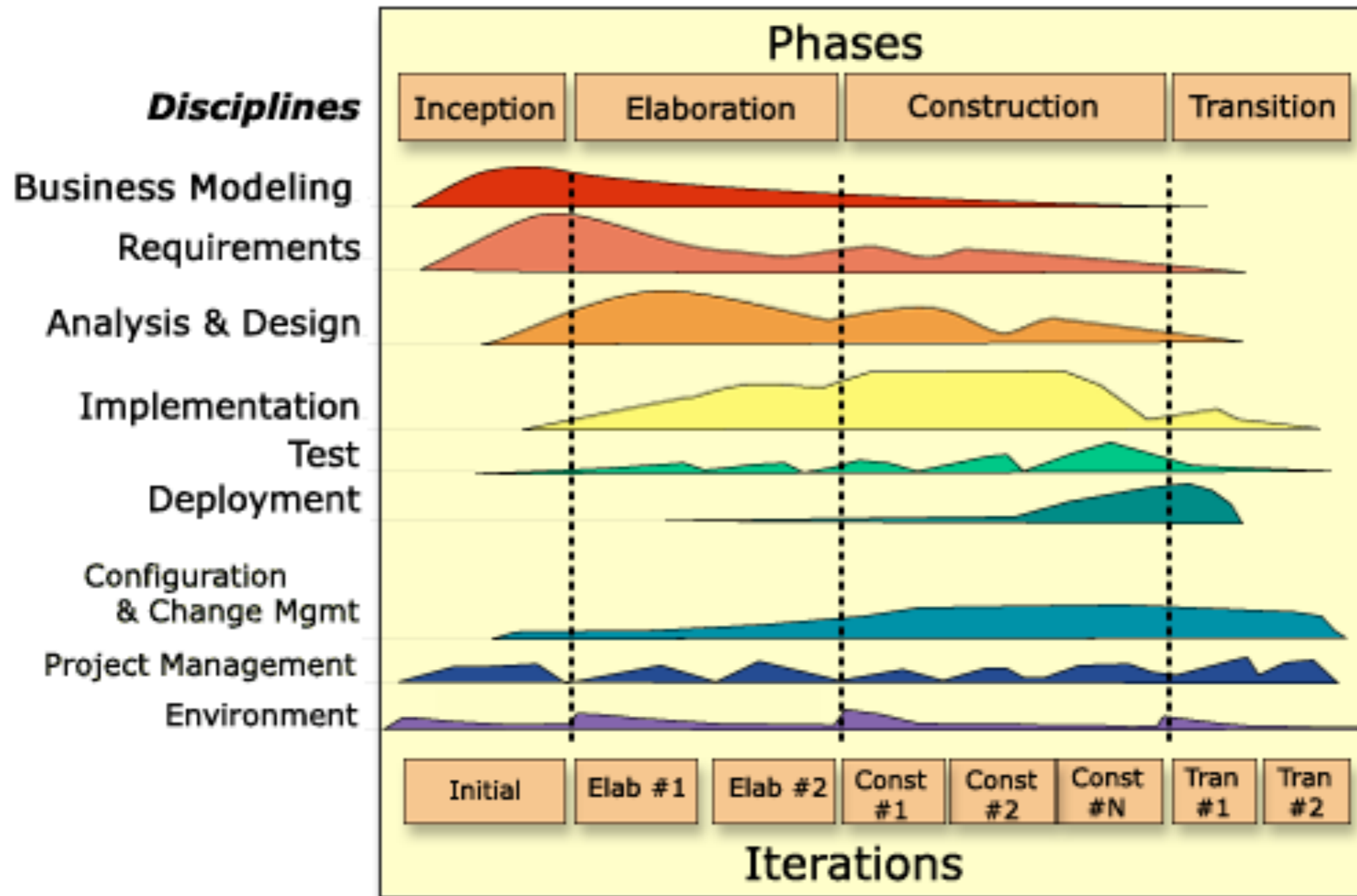
Unified Process, cont...

- Rather than a single construction phase, there are a number of construction phases
- A prescriptive process - each step must be followed (more or less) to ensure success
- A BA's role on a UP project consists of eliciting, organizing, and documenting required functionality - very similar to SDLC

Unified Process, cont...

- Requirements are generally fully documented well advance of a hand-off to developers
- Requirements generally must be signed-off before a hand-off
- BA Product Output: Use Case Diagram(s), Use Cases and/or UML Activity Diagram(s)

Unified Process



Use Cases - A Retrospective

- Names a goal, e.g. “Register for Courses”
- Consists of scenarios, which consist of action steps
- Each action step is phrased as a goal
- Possibly one or more Alternate Flows, or Exception flows

Use Cases - A Retrospective

- Has a specific scope
- Documents system interaction between a Primary Actor (and/or one or more Secondary Actors) and the System
- Contains a Basic Flow - main success scenario

Sample Use Case

1. Use Case Name: Register for Course

1.1 Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selection if changes are made within the the add/drop period at the beginning of of the semester. The Course Catalog System provides a list of all the current course offerings for the current semester.

The main actor of this use case is the Student. The Course Catalog System is an actor within the use case.

2. Flow of Events:

The use case begins when the Student selects the “maintain schedule” activity from the Main Form.

2.1 Basic Flow

2.1.1 Create a Schedule

2.1.1.1 The Student selects “create schedule”

2.1.1.2 The system displays a blank schedule form.

2.1.1.3 The system retrieves a list of available course offerings

2.1.1.4 The Student selects 4 primary course offerings and 2 alternate course offerings from the list of available offerings.

2.1.1.5 The Add Course Offering subflow is performed at this step for each selected course offerings.

2.2 Alternate Flows

2.2.1 Modify a schedule...

Q: What do these two process methodologies have in common?

A: A very high level of ceremony.

What is Ceremony?

The amount of effort and/or rigor that is applied to a process to define it in an absolute series of steps or sequence of operations so as to impose external constraints on an individual or team.

Examples of Ceremony

- A detailed project plan before starting the project
- A detailed full-spec'd out requirements document before construction
- A detailed and documented design document before construction can begin
- A detailed set of test plans that are documented before testing can begin

- How will Agile thinking and processes improve upon these previous methodologies?
- How can it provide any benefit to make the likelihood of success for a project more probable?
- For that we need to understand what “doing Agile” really means

The Agile Movement - A Primer

- Two days in February
- 17 Individuals from the “alternative” methodologies movement: eXtreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal Clear, Feature Driven Development etc. created the Agile Alliance
- Developed the Agile Manifesto: their values to creating quality software

Agile Manifesto

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, the Agile Alliance values the items on the left more.

Values? For real?

Ok, values are one thing, but how does that translate to actions which lead to a higher quality (i.e. successful) project and/or product?

Enter the 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.

- How does this translate to actions?
- “Cogito, ergo sum”
 - “I think, therefore I am” *Rene Descartes*
- Buddhism’s Right Intention leads to Right Action
- We’re setting the mind proactively to produce a desired action
- The values and principles are about setting out a mindset, an intention

Distilling the Values and Principles

A Summary

The Actions

- Simplicity
- Communication
- Upfront Business value
- Quality above quantity
- Customer is foremost involved
- Self-organizing and self-managing teams
- Introspecting teams - the ability to critically analyze our selves and improve upon our weaknesses
- Working software above all else

Agile Methodologies

- eXtreme Programming (XP)
- SCRUM
- Crystal Clear
- Lean Software Development

Principles of XP

- Involves analysis, design, coding, testing and deployment
- These activities occur always and simultaneously
- XP is about iterating
- Iterations are from 1 week to 4 weeks in duration with preference for the shorter time frame
- Planning occurs with on-site customer - the customer, or customer proxy is a member of the development team and works directly with developers

Principles of XP, 2

- XP calls this the planning game
- No upfront Analysis phase - customer sits with the developers to create software based on prioritized user stories (more on these later)
- Customers provide requirements using their experience and traditional techniques - an area where a good BA is definitely needed.
- BA facilitates/participates in interactions between Customers and Developers

Principles of XP, 3

- Coding is done in pairs (generally) using Test Driven Development (TDD)
- Note, architecture is evolutionary allow it to evolve as understanding of both requirements and adapting to changes influence system behavior
- Achieved through a process called refactoring: changing the design of an existing system through a series of small adaptive code changes
- Continuous Integration, frequent code committals to an SCM and an automated build system

Principles in Summary

- Pair Programming
- Test Driven Development
- Informative workspace
- We sit together
- Sustainable pace - 40 hour work week with no overtime
- Stand-up meetings
- Version Control Repository

Principles in Summary, 2

- Incremental Requirements - JIT!
- Continuous Integration
- Collective Code ownership
- Automated Build Process
- Automated Testing
- Frequent deliveries for customer feedback

The BA on an XP Project

- Augments on-site Customers involvement in system understanding
- Does not replace this involvement
- Creates use cases and/or user stories
- Help clarify and facilitate discussion among high-level stories
- Assists in Release and Iteration planning with Story overview and prioritization

SCRUM

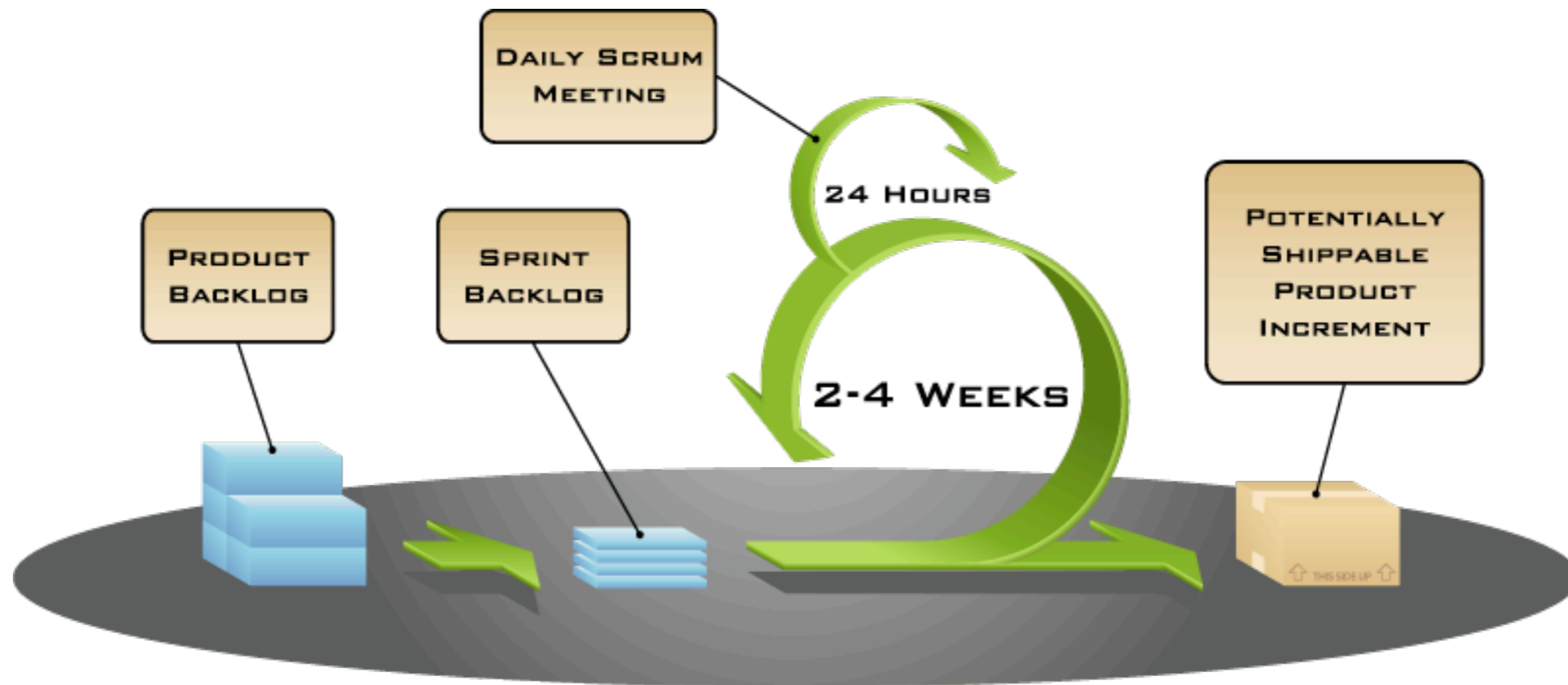
Principles of SCRUM

- Self-directed and self-organizing team
- Product Manager, manages Product Backlog
- Iterations are called Sprints
- Sprints last from 2 to 4 weeks
- Developers implement Sprint backlog
- No additional extra work added to an in-progress iteration, once chosen

Characteristics of SCRUM

- Daily stand-up meetings with 3 specific questions
- Sprints end with a Sprint Demo
- Each iteration is based on client-driven adaptive planning
- Each SCRUM team has a SCRUM Master (teaches the process and makes sure it fits within the Corporate culture)

A Picture is worth a Thousand Words



Being the BA on a SCRUM Project

- Responsible for capturing all items required for the Product Backlog and Release Backlog
- Requirements still need to be elicited
- The focus is on JIT requirements gathering
- Capture features, use cases/stories, enhancements, defects and technologies

Being the BA on a SCRUM Project

- The challenge? Reduce unnecessary documentation and unnecessary ceremony
- Act as Customer-proxy for Development in addition to working with on-site customers
- An addition to, not a replacement for, the on-site customer

Crystal Clear

Principles of Crystal

- Frequent Delivery - Focused Feedback
- Reflective Improvement
- Osmotic Communication
- Personal Safety
- Focus - knowing what to work on and time to work on it
- Easy Access to Expert Users
- Technical Environment with Automated Tests, Configuration Management and Frequent Integration

Lean Software Development

Principles of LSD

- Eliminate Waste
- Build Quality In
- Create Knowledge
- Defer Commitment
- Deliver Fast
- Respect People
- Optimize the Whole

Ok, so which one do I pick?

All of them, yet none of them!

Agile is Descriptive, it's not Prescriptive!

Key Practices when “doing Agile”

- Collective Code Ownership
- A work environment that fosters communication and collaboration
- Everyone has a voice on the team and a stake in its success
- Self-organizing teams
- TDD for developers work
- Automated Build Process

Key Practices when “doing Agile”

- Automated Testing
- Retrospectives
- Continuous Learning
- Continuous Self-Improvement
- Less documentation/ceremony for the BA
- UML Activity diagrams created where they add value

Key Practices when “doing Agile”

- Diagrams are to facilitate communication and understanding, nothing more
- Documentation generally not kept at the end of the iteration
- The code becomes the living documentation of the system

The need to shift Perspectives

Project Manager: “Sure, we don’t apply waterfall - everyone knows it doesn’t work. We’ve adopted <our Agile method> and are into our first project. We’ve been at it for two months and have the use case analysis nearly finished and the plan and schedule of what we’ll be doing in each iteration. After review and approval of the final requirements set and iteration schedule, we’ll start programming.”

From Use Cases to User Stories

- User stories are one to two paragraphs of text for some feature that provides business value
- A note for a future communication
- A high-level user goal or system feature

User Stories

- Estimable
- Implementable
- Testable

Example User Stories

- “A student can register for a course”
- “A student can drop a course”
- “Users can update their membership information”
- “The administrative user can update the event details”
- “A user can add a new Event to the calendar of Events”

Making the Shift

- Details, schmetails!
- It's about the software
- No one person is more important than any other
- You have knowledge that developers and testers can benefit from - offer that
- No one is competing on an Agile project: We either all succeed or we all fail
- Let your Manager's play the blame game!

Key Success Factors

- Remember the downstream consumer of your work: Developers
- Spare us the details! We'd rather have you sit with us at our Computer
- A high-level description, but not verbose
- Must be an implementable feature providing Business Value
- How do we test it?
- What could possibly go wrong with this feature?

An Agile Project from the BA's Perspective

BA - Release Planning

- Come to the release planning with a detailed understanding of the Product Backlog and key risk areas
- Be prepared to detail and discuss any or all stories
- Help establish both priority and commonality for stories or features that may fit together into releases
- Assist key stakeholders in understanding feature value to the business

BA - Iteration Planning

- Come to the iteration planning meeting with your user stories
- Be prepared to detail and discuss these stories
- Provide additional testing scenarios for developers (alternate flows and exception flows in traditional Use Cases)

Iteration Planning, 2

- Assist with establishing customer priorities
- Bring your detailed system understanding to help mitigate risks to the implementation of the stories
- Assisting with Developer estimates based on priorities and complexity

BA - The Iteration

- Proactive requirements for next iteration/sprint
- Working with developers (sitting at their desks)
- Providing Test team support of requirements understanding
- Working the Sprint backlog to maintain project velocity
- Assisting with testing and testing automation

Wrapping Up

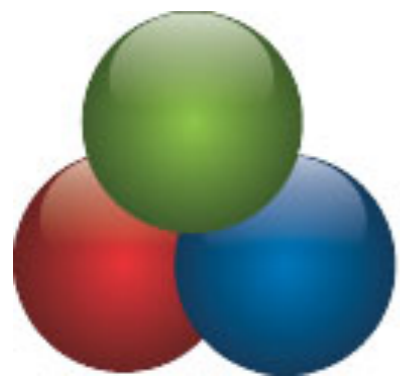
- Remember the intent of Agile is to keep the focus on the software
- If it doesn't add measurable value, question why you're doing it
- Ask yourself: "Is this the simplest way to do this?"
- "Embrace Change"
- Change happens, adapt your processes to accommodate changes sooner, rather than later

About Elite Software Architects Inc

- Small, Agile organization that is about upfront value
- Primary goal: Successful software projects, happy people
- How? Always looking for team players and adaptable participants who are willing to learn and grow

References

- [1] - Writing Effective Use Cases, *Alistair Cockburn*
- [2] - The Art of Agile Development,
- [3] - User Stories Applied, *Mike Cohn*
- [4] - Agile Estimating and Planning, *Mike Cohn*
- [5] - Agile Project Management, *Jim Highsmith*
- [6] - Agile Software Development, Principles, Patterns and Practices, *Robert C. Martin*
- [7] - Agile Software Development, A Manager's Guide - *Craig Larman*
- [8] - Waterfall Model, *Wikipedia*
- [9] - IBM Rational Unified Process, *Wikipedia*
- [10] - SCRUM Image, www.mountangoatsoftware.com/scrum



TCBAC
twin cities business analyst community

Questions and Wrap-up

Please fill out the feedback cards on the table

For more information please contact

TCBAC@solutiaconsulting.com

or refer to our website at

www.TCBAC.org

Next Meeting:
October 9th, 2008
Topic TBD