

ISM

IT Solutions Management for Human Services

an affiliate of the American Public Human Services Association



2007 ISM Conference Boston

Innovative Approaches to
Software Development
Ohio SACWIS
6 August 2007



- Large Projects are
 - High Risk
 - 44% of software projects run over budget
 - 30% of software projects are canceled
 - 50% of software projects are not considered a success by the business
 - 90% of software projects are delivered late
 - Difficult to Effectively Manage
 - Expensive
 - Hard to Evaluate
 - Outcomes
 - Progress
 - Require Structure AND Agility (i.e., ability to adapt to change)



Ohio Department of Job and Family Services

- Develop and oversee programs that provide:
 - Healthcare
 - Employment and economic assistance
 - Child support and services to children and families
- Largest state agency in Ohio
- State Automated Child Welfare Information System (SACWIS) project
 - 6000+ State/County Workers
 - 120,000 Child Welfare Cases/Year
 - 54,000 Abuse
 - 44,000 Foster Care
 - 44,000 Adoption



- **Background Numbers**
 - 167 Use Cases
 - 650+ web pages
 - 176 on-line reports
 - 6000 Functions
 - 400+ Data Classes/Entities
 - 3.8M lines of code
- **6 System Interfaces (4 batch, 2 real-time)**
- **Replace Functionality/Convert Data from State and County Legacy Systems (7 systems)**
- **24x7x365 Availability**
- **Accomplish Technical Knowledge Transfer**



- **Business**
 - Provide appropriate, consistent, cost effective and useful services to children and families
 - Provide timely access to case information
 - Minimize data entry; maximize client face-time
 - Incorporate County Requirements in Statewide System
 - Enable smooth transition for end users
- **Technical**
 - Build a Reusable Framework and Services to Maximize Reuse
 - Incorporate Rational Unified Process (RUP)
 - Incorporate Agile Methods
 - Model Driven Architecture (MDA) Best
 - Make sure “-ilities” are maximized



Ohio SACWIS Approach

- Generate code from models
 - Import Rose Models/Use Cases/Business Rules
 - Employ Visual Modeling Techniques
 - Implement Service Oriented/Model Driven Architecture
- Combine a formal process with elements from Agile programming
 - Short Development Iterations
 - Continuous Business Feedback Loop
 - Small Teams with integrated vendor/state staff
 - Pair Programming
- Continuous Integrated Testing
 - Align agile testing with iterative development
- Use automated testing tools, techniques and metrics



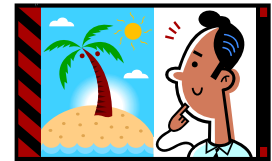
Why Use This Approach?

- Hundreds of existing methodologies out there
- Current “Biggies”
 - Waterfall
 - Iterative
 - eXtreme



Waterfall on “The Island”

- You are stranded on a deserted island with a co-survivor who has skills to build things
- You need a boat....
- Your co-survivor asks you to describe all the requirements for the boat, goes to the other side of the island for six months to build the boat, and returns six months later when it is done.
- You finally see your boat and it is a raft! It has a sail, and will do basically all that you asked for. But, it is not what you thought you'd communicated and it won't meet all your needs - in this case, to get to the mainland!
- Inherent risks with the Waterfall Methodology are unanticipated outcomes, resulting in acceptance or usability issues.





Iterative Development on “The Island”

- You are still stranded on a deserted island... and you still need a boat!
- Your co-survivor asks you all about how you will use the boat from start to finish. You draw pictures until you both are sure you understand.
- Your co-survivor goes off. In two weeks, he's back and shows you what he's built. It's a raft! But, only two weeks are lost, and you discuss it again until your full needs are understood.



Shorter Cycles = Reduced Risk



- Project development is divided up into a number of short iterations, or cycles
- Every iteration has its own phases of requirements, design, implementation, etc.
- Measurement Metrics are easily identified/tracked
- The project grows from simple to complex
- Conversion aligns with iterative development
- Every iteration ends in:
 - The delivery of a stable build
 - A visual model of the emerging system
 - Lessons Learned to incorporate into the next iteration
 - Incremental sign off on implementation of requirements



Iteration Details (The Work)

Analysis & Design (Iteration X+1)

Kickoff	JAD Preparation	Perform JAD	Design High Level Design	Architecture Roundtable	Subject Matter Expert Business Review	Lessons Learned
					Update Business Models	

Construction (Iteration X)

Kickoff	Detailed Design	Author Unit Test	Customized Iteration Baseline			Business Demo	Lessons Learned
			Unit Test				
Update OJ Models							

Internal System Integration Test (Iteration X)

Kickoff	Run Test Scripts	Write Test Cases	Define Test Data	Lessons Learned
			Write Test Scripts	

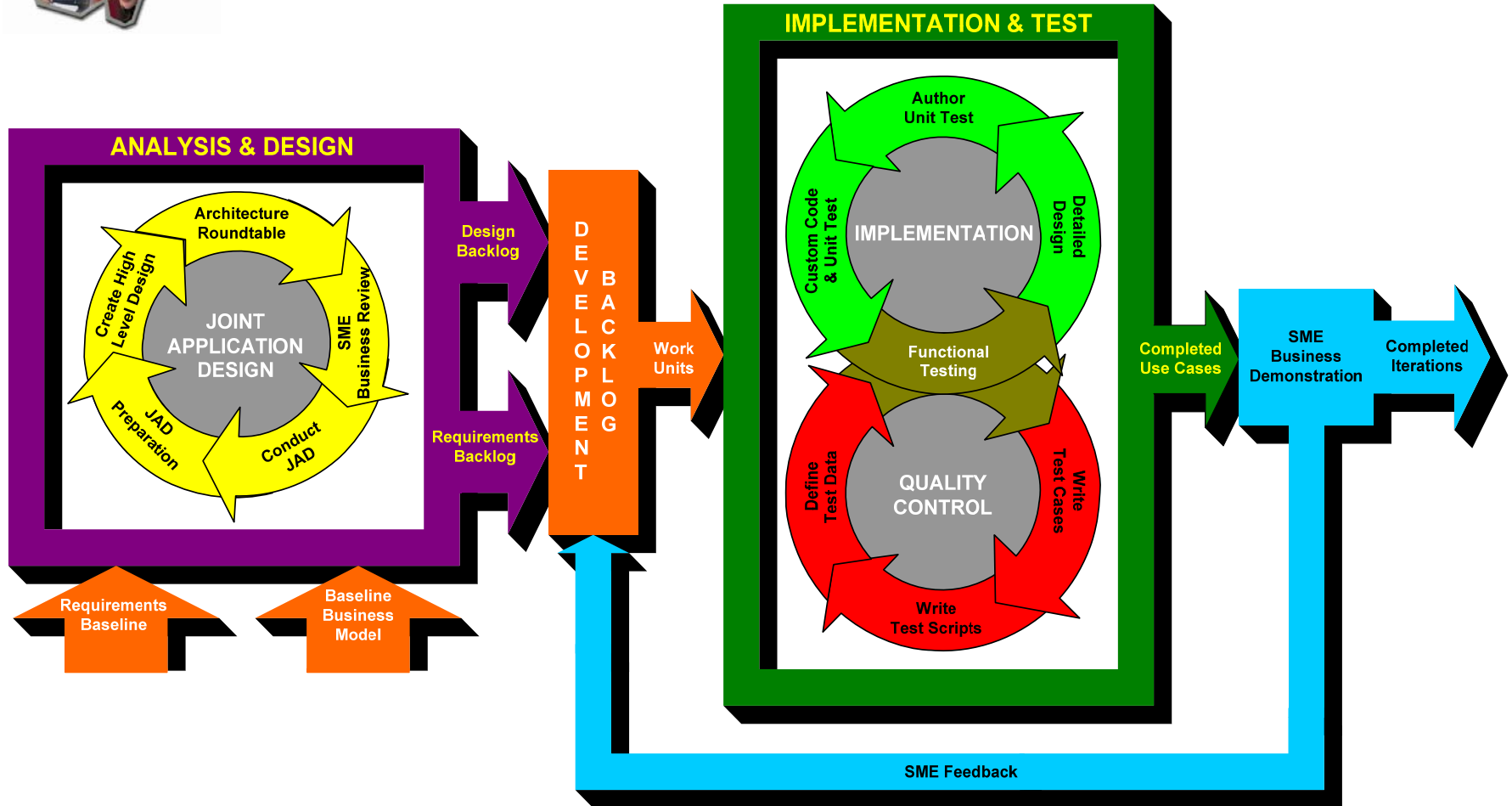
Project Management (Iteration X & X+1)

Capacity Planning	Software Configuration Management						Future Activity Project Planning
	Activity Tracking & Oversight	Release Management			Performance Metrics		
		Defect Management					
Process Improvement							

Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------



Iterations in Motion





- Developers and Subject Matter experts often think completely differently, almost as if in a foreign language
- Use Cases are activity-based tools to translate business needs into application functionality to support those needs
 - Ensures the application meets functional requirements as well as matching your business flow and process.
- It to refocus the questions from:

What do you want? What components do you need?

to

What activity do you want to accomplish? What are the steps?

- Visual models are generated that illustrate the activity, all the way through design activity.
- Visual model is viewable in the emerging application



Visual Modeling on “The Island”

- You are stranded on a deserted island with a co-survivor who has skills to build things but...
- Your co-survivor speaks a different language. How would you try to communicate your needs for your raft? How about with pictures?
 - What are you going to use your raft for?
 - How will you go about the tasks on your raft?
- You draw a picture, or visual model, of what you will be doing with the boat. Your co-survivor then solves how the boat must work to support what you need it to do.
- Use cases are the tool for translating business needs into applications that support those business needs. Visual modeling extends that tool to assist in understanding abstract interactions and associations.



Model Driven Architecture

- Model Driven Architecture® (MDA) is an innovative way of developing applications and writing specifications to manage the challenge of rapid business and technology change.
- The application is first developed using a modeling tool. Once the model is complete, the modeling tool generates the majority of actual application code.
- Why is this important? Because as technical and functional changes are needed, they can be made in the model, not at the code level. Changes are quicker, more efficient, and of less risk and impact than with traditional coding techniques.
- Why is this faster? The model abstracts the more difficult tasks (e.g. common connections using a multiplicity of connections languages) and allows the developer to focus on application specific code.



Model Driven Architecture on “The Island”

- Your talented co-survivor, using a model first, has built a boat that is almost perfect, BUT... a test drive in the lagoon during a storm reveals a design problem. The boat is sinking.
- Without MDA:
 - You may wait an additional month while your co-survivor undoes half of the hull and rebuilds the boat re-tying every knot that keeps it together
 - You re-test the boat, and discover that something else was broken by the effort!
- With MDA:
 - Your co-survivor changes the critical part of the model and the model re-builds the boat. It takes far less time and everything still works!
 - OR, Your co-survivor realizes that the boat is made out of wood and should have been made out of fiberglass. Instead of having to restart from scratch, fiberglass is added as a base material and the boat re-builds itself with a fiberglass hull instead of a wooden one.



Lessons Learned (Keeping Change Constant)

- This mixture works, but unanticipated items can still extend the project schedule
 - Major Change to Child Welfare Practice regarding risk assessment identified during Requirements (CAPMIS)
 - Realization that what was discussed in theory did not work when designed
- Communication Across Multiple Teams is Difficult at best
 - Architectural Round Tables
 - Expressions of standards and guidelines
 - Communication of reusable objects and services



Lessons Learned (Internal Structure)

- Use Cases and Visual Modeling are Key to Customer Acceptance
- Iterative Approach Increases Comfort Level for:
 - End Users
 - Management
 - Business Analysts
- Incorporation of Industry Best Practices/Tools does NOT increase cost or expand schedule over the life of the project
- Structured Process Provides
 - Necessary Discipline to Effectively Manage Scope
 - Flexibility to Effectively Handle Change

Lessons Learned (Staying Agile)



- Deliver working software as frequently as possible
- Deliver what is necessary – not what someone may want in the future
- Small teams support the most effective method of conveying information to and within a development team (face-to-face conversation)
- Working Software is the Primary Measure of Progress
- The art of maximizing the amount of work NOT done is essential
- Resist the urge to “hunker down” – share what you are doing