Project Management in a RUP Environment: Driving Iterative Development With Use Cases

Ian Spence
IBM Software Group | Rational software
spenceia@uk.ibm.com
Agenda

- A Brief Introduction to the Rational Unified Process
- Planning the Overall Project
- A Walk Through The RUP Lifecycle
- Summary
Agenda

- A Brief Introduction to the Rational Unified Process
- Planning the Overall Project
- A Walk Through The RUP Lifecycle
- Summary
The Best Best Practice - Mature, Distilled, Accessible

Rational Unified Process®
Phases, Roles, Activities, Artifacts, Workflows
RUP – Software development: from idea to delivery…..

….from Business Modeling to Deployment
The Characteristics of the Rational Unified Process

1. Iterative and incremental
2. Risk focused
3. Architecture-centric
4. Use-case driven
5. Tailorable
6. Based on the Unified Modelling Language (UML)
What is an Iteration?

An iteration is a distinct sequence of activities based on an established plan and evaluation criteria, resulting in an executable release (internal or external).
Risk Profile: *Waterfall vs. RUP Projects*

- **Waterfall Risk**
- **Iterative Risk**

**Risk Reduction**

- **Risk**
- **Time**
Attack Significant Risks Early: Architecture First

**Benefits of an Architecture First approach**
- Early risk reduction, testability insight
- Demonstration based assessment
- Focus on the important 20%
  - Plans, requirements, use cases, designs, components, test cases, make/buy decisions
Evolving Levels of Detail

Get the architecture right first (the stuff that counts), then worry about completeness and precision.
The Progress Profile of an Iterative Project

Sequential iterations, but iterative activities

Prototypes ➔ Architecture ➔ Functional Releases ➔ Product Release

Development Progress (% coded)

Iterative Project Profile

Waterfall Project Profile

Sequential iterations, but iterative activities

Adapted from: Walker Royce, 1995
The RUP Defines a Risk Driven Lifecycle

<table>
<thead>
<tr>
<th>RUP Phase</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Confirm the scope and objectives of the project and <em>bring the business risks under control</em></td>
</tr>
<tr>
<td>Elaboration</td>
<td>Stabilize the product plans and <em>bring the architectural and technical risks under control</em></td>
</tr>
<tr>
<td>Construction</td>
<td>Build the product and <em>bring the logistical, project execution risks under control</em></td>
</tr>
<tr>
<td>Transition</td>
<td>Deliver the product and <em>bring the roll-out risks under control</em></td>
</tr>
</tbody>
</table>
The RUP is Controlled Iterative Development

Major Milestones: Planned (Business) Decision Points

- Lifecycle Objective Milestone
- Lifecycle Architecture Milestone
- Initial Operational Capability
- Product Release

- Project Viability Agreed
- Selected Approach Proven
- Usable Solution Available
- Project Completed

Iteration Assessments: Planned (Technical) Visibility Points
Major Artifacts
Use Cases Bind the Core Disciplines Together

- Project Management
- Test
- Requirements
- Implementation
- Analysis
- Design
What is a Use Case?

A use case describes a sequence of actions a system performs that yields an observable result of value to a particular actor.

- Use cases are shown in UML diagrams
- Use cases are described in text
  - They tell the story of the interactions between actors and the system

Bank Customer: Withdraw Cash
Looking inside the use case

**Basic Flow**
1. Insert Card
2. Validate Card
3. Select Cash Withdrawal
4. Select Amount
5. Confirm Availability of Funds
6. Return Card
7. Dispense Cash

**Alternative Flows**
A1 Invalid Card
A2 Non-Standard Amount
A3 Receipt Required
A4 Insufficient Funds in ATM
A5 Insufficient Funds in Acct
A6 Would Cause Overdraft
A7 Card Stuck
A8 Cash Left Behind
Etc…

**Scenarios**
1. Successful withdrawal of cash with a receipt
2. Attempt withdrawal when Bank System unavailable
3. Attempt withdrawal with a “hot card”
Etc…
Use Cases Drive Development

Use Cases & Supplementary Requirements

Comp A  Comp B  Comp C  Comp D

Architecture and Design

Documentation and UI Design

Test Cases
Agenda

- What is RUP?
- Planning the Overall Project
- A Walk Through The RUP Lifecycle
- Summary
The Principles of Lifecycle Planning

- Understand the desired outcomes
- Identify and assess overall risks
- Create an achievement-based roadmap
- Set the management strategy
- Assess and estimate the work to be done
- Define the project plan(s)
- Delegate the execution of the plan(s)
- Plan to iteratively evolve and challenge the plan(s)
Plans Share Milestones

PRINCE2 Project Plan – Business Significant Milestones
coverage - all specialist projects / management stages

Business Significant Milestones Allocated to Specialist Project

Software Development Plan – S/W Significant Milestones
coverage – the development of a major release

Major Milestones

Phase Plan – Phase Significant Milestones
coverage – single phase

Iteration Objectives

Iteration Plan – Iteration Significant Milestones
coverage – single iteration
Planning the Project

- Map Project Delivery Dates on Phase and Iteration Milestones

- Use the iteration and phase-ends as progress checkpoints

- Determine how much “Inception” is already done
  - If business problem and funding are already defined, focus on the viability of potential solutions

- Work Top-Down and Bottom-Up
  - Map risks to iterations
  - Select use-case scenarios that will force “confrontation” of the risks
  - Plan related requirement, analysis, design, and test activities
# Release Planning

<table>
<thead>
<tr>
<th>Initiation</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Close</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inception</td>
<td>Elab</td>
<td>Con</td>
<td>Trans</td>
</tr>
<tr>
<td></td>
<td>Release 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inc</td>
<td>Elab</td>
<td>Con</td>
<td>Trans</td>
</tr>
<tr>
<td></td>
<td>Release 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inc</td>
<td>Elab</td>
<td>Con</td>
<td>Trans</td>
</tr>
<tr>
<td></td>
<td>Release 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inc</td>
<td>Elab</td>
<td>Con</td>
<td>Trans</td>
</tr>
</tbody>
</table>

Adapted from the PRINCE2 manual: Figure 16.6 page 237
Select an Iteration Pattern

Incremental (1)

Evolutionary (2)

Incremental delivery (3)

“Grand design” (4)
Risks Can Be Balanced Across Multiple Releases

Architectural risk could be brought forward making the initial projects Elaboration Phase harder and the second projects easier.

Production risks could be moved backwards reducing the business benefits of the earlier project.

Remember the iteration patterns. The initial project is likely to be “Evolutionary”. The later projects “Incremental”.
Adjusting Estimates Across the Lifecycle

Project Schedule

Completion Date

Original Target Date

Estimate of work to complete
Introduction to our Example

ACME Super ATM

- Objective: to provide a modular, expandable and customizable platform for ATMs, and ultimately general-purpose “dispenser” kiosks
  - Configurable as ATM, ticket dispenser, postage dispenser, etc.

- Design goals
  - To reduce cost of supporting kiosk dispensing systems used in different markets, and to achieve a shared architecture
  - To expand into new markets
Example Risks for the ACME Super ATM

---

**Business Risks**
- Market may be too crowded with competitors
- May not be able to price product aggressively enough to displace competitors
  - Cost to develop may be too high
  - Profit margin may provide insufficient return on investment
- Customers may not want a new or more flexible solution

---

**Technical Risks**
- PC/Windows-based platform may not be reliable enough for remote/unattended use
- System may not “fail gracefully” if power or network connection is lost in the middle of a transaction
- System may not be able to be made “tamper-proof”
- May not be able to deliver a sufficiently flexible, component-based system at low cost
- Underlying technology (J2EE) may not perform sufficiently well to meet needs
Use Cases for the ACME Super ATM

Customer:
- Withdraw Cash
- Deposit Funds
- Transfer Funds
- Manage Account

ATM Engineer:
- Refill and Service the Machine
- Configure the Machine
- Check the Machine is in Working Order
- Analyze System Performance

ATM Operator:
- Reconcile Transaction Logs
- Update System Configuration
- Run Advertising Campaign

Burglar:
- Break Into Machine
## Overall Release Plan for the ACME Super ATM

<table>
<thead>
<tr>
<th>Initiation</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Close</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="chart.png" alt="Diagram" /></td>
<td><img src="chart.png" alt="Diagram" /></td>
<td><img src="chart.png" alt="Diagram" /></td>
<td><img src="chart.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

### Release 1: Basic Withdrawal Facilities
- Inception
- Elab
- Con
- Trans

### Release 2: Account and Systems Management
- Inception
- Elab
- Con
- Trans

### Release 3: Full Account Management and Cross Selling
- Inception
- Elab
- Con
- Trans
Use Cases for the ACME Super ATM

- **Customer**
  - Withdraw Cash
  - Deposit Funds
  - Transfer Funds
  - Manage Account

- **Burglar**
  - Break Into Machine

- **ATM Operator**
  - Reconcile Transaction Logs
  - Run Advertizing Campaign
  - Update System Configuration

- **ATM Engineer**
  - Refill and Service the Machine
  - Configure the Machine
  - Check the Machine is in Working Order
  - Analyze System Performance

**Release 1:** Dark Blue, **Release 2:** Medium Blue, **Release 3:** Pale Blue, **Out of Scope:** Light Blue
Partial Project Plan for the ACME Super ATM Release 1

Inception Phase

- Risks Mitigated:
  - Crowded market
  - Product pricing and profitability
  - Customer demand for configurable transactions

- Scenarios:
  - Withdraw Cash
  - Configure Transaction

Elaboration Phase

- Risks Mitigated:
  - Reliability of OS platform
  - Scalability of J2EE infrastructure
  - Fault tolerance
  - Tamper-proofing
  - Remote operation

- Scenarios:
  - Break into Machine
  - Withdraw Cash + Bank System Stopping Responding
  - Withdraw Cash Basic Flow + Handle Transaction Log Failures + Remote Reconciliation
  - Withdraw Cash (scenarios in parallel)

Construction Phase

- Risks Mitigated:
  - Completing work on time, within budget

- All remaining use cases and scenarios
# Summary of Increments

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Resulting Incremental Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>I1</td>
<td><em>Proof of Concept: Basic cash dispensing</em></td>
</tr>
<tr>
<td></td>
<td>I2</td>
<td><em>Proof of Concept: Configurable cash dispensing</em></td>
</tr>
<tr>
<td>Elaboration</td>
<td>E1</td>
<td><em>Prototype: Withdraw Cash Basic Flow</em></td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td><em>Prototype: Withdraw Cash w. basic failure modes and protection + initial load testing</em></td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td><em>Prototype: Withdraw Cash w. correction and reconciliation + load and performance testing.</em></td>
</tr>
<tr>
<td>Construction</td>
<td>C1</td>
<td><em>Functional Release: Basic cash withdrawal</em></td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td><em>Use Cases: Withdraw Cash, Refill and Service</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Functional Release: Usable Cash Withdrawal</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Complete remaining use cases:</em></td>
</tr>
<tr>
<td>Transition</td>
<td>T1</td>
<td><em>Patch Release: Bug fixes</em></td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

- What is RUP?
- Planning the Overall Project
- A Walk Through The RUP Lifecycle
- Summary
Risks Addressed in the Inception Phase

In the Inception Phase we focus on “Business Risks”

<table>
<thead>
<tr>
<th>Risk</th>
<th>Questions to ask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Proposition</td>
<td>“Does it solve a meaningful problem that people are willing to pay to solve?”</td>
</tr>
<tr>
<td>Economic Feasibility of Solution</td>
<td>“Can we build it, profitably?”</td>
</tr>
<tr>
<td>Time to Market</td>
<td>“Can we deliver the solution within the window of market opportunity?”</td>
</tr>
<tr>
<td>Funding</td>
<td>“Given all the other things we need to do, can we fund this?”</td>
</tr>
</tbody>
</table>

We must be able to answer these questions with reasonable certainty by the end of the Inception Phase.
Exploring the Value Proposition

- Understand the Customers and their problem(s)
  - Make sure you get at the root causes of the problems, not just symptoms

- Identify potential solutions to these problems
  - Often a solution involves both process and technology
  - A set of use cases can be used to define each potential solution
    - Identify and briefly describe key actors and key use cases
    - For a few use cases, brief outline of the basic flow is useful
      - Identify important alternative flows
    - Focus on the value provided to the customers/stakeholders
    - Also capture any key constraints or non-functional requirements
  - Small functional prototypes are often useful to illustrate key aspects of value proposition

- Understand the value the customer will receive from each potential solution
  - This places an upper bound on what you can charge for the solution
Exploring the Viability of Potential Solutions

- Can the solution be developed?
- How much will it cost?
- How much time will it take?
- What are the technical risks of the solution?
- Funding
- Time to Market
Example: The ACME Super ATM

Tasks for the Inception Phase

- Identify the problems and root causes. Develop a vision for the solution.
- Identify primary actors and use cases; briefly describe and outline basic flows.
- Develop a coarse-grained estimate of overall schedule and costs to develop proposed solution.
- Determine whether project should move ahead.

- Customer
  - Withdraw Cash
  - Deposit Funds
  - Transfer Funds
Concluding the Inception Phase

- Project Viability Agreed
- Alternatives evaluated and business case produced
- Vision established and shared
- Risks analysed and understood
- Overall Project Plan Updated
  - Releases, deadlines, estimates, constraints and objectives clarified
- Elaboration phase planned
- Primary / critical use cases and key non-functional requirements identified
  - A few key use cases outlined with alternatives identified
- Initial prototypes to prove the value proposition and address technical issues

---

**Inception**

**Elaboration**

**Construction**

**Transition**

* time

**Lifecycle Objective Milestone**

**Lifecycle Architecture Milestone**

**Initial Operational Capability Milestone**

**Product Release**
Risks Addressed in the Elaboration Phase

In the Elaboration Phase we focus on “Technical Risks”

<table>
<thead>
<tr>
<th>Risk</th>
<th>Questions to ask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement on Capabilities</td>
<td>“Do all parties share a common understanding of what will be built?”</td>
</tr>
<tr>
<td>Technical Feasibility of Solution</td>
<td>“Can the solution meet the requirements?”</td>
</tr>
<tr>
<td></td>
<td>“Can we build it in the time allotted, with funded resources?”</td>
</tr>
</tbody>
</table>

We cannot successfully exit the Elaboration Phase until we can resolve these questions
Gaining Agreement on Capabilities

- Identify all primary use cases
  - Don’t worry about *inclusion* or *extension* just yet

- Document architecturally significant use-case flows
  - Only describe flows that will force resolution of architectural choices
    - Identify all flows but describe only some at this time

- Document “supplementary” requirements
  - Performance, scalability, security, etc.

- Identify architecturally significant “scenarios”
  - A scenario consists of the basic flow plus zero or more alternative flows
  - Architecturally significant scenarios are those that will force the team to confront a set of technical issues that will shape the architecture
  - Example:
    - Withdraw Cash Basic Flow + A4.11 Handle Transaction Log Failures
Ensuring Technical Feasibility

For each architecturally-significant scenario:

- Create a use-case realization, identifying architecturally-significant components and allocating functionality to them.
- Develop Test Cases that can validate the scenario:
  - Black-box and White-box tests.
- Design and implement the use-case realization:
  - “Internals” of most of the components can be “stubbed-out”; the important thing is to work-out the interactions between, and responsibilities of, the components.
  - Implement “internals” selectively to evaluate key technical issues:
    - Algorithmic complexity
    - Performance/scalability
- Evaluate the results through testing:
  - “Paper reviews” of the architecture don’t count – the goal is to identify things you don’t know rather than confirming things that you do know.
Concluding the Elaboration Phase

- Selected Approach Proven
- Stable, proven, executable architecture
- Technical risks mitigated
- Business case confirmed
- Accurate estimates available
- Requirements stable
  - All use cases identified and outlined, including alternative flows
  - Architecturally significant scenarios detailed
- A partially completed Design Model, Test Cases, executing Code

---

Project Viability Agreed

Life Cycle Objective Milestone

Life Cycle Architecture Milestone

Initial Operational Capability Milestone

Product Release
Risks Addressed in the Construction Phase

- In the Construction Phase we focus on “Logistical Risks”

<table>
<thead>
<tr>
<th>Risk</th>
<th>Questions to ask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to do the work</td>
<td>“Are the resources in place?”</td>
</tr>
<tr>
<td></td>
<td>“Is progress sufficient to meet the project commitments?”</td>
</tr>
<tr>
<td>Quality and completeness of the solution</td>
<td>“Is the product of sufficient quality to release?”</td>
</tr>
<tr>
<td></td>
<td>“Is the product of sufficient functionality to be worth releasing?”</td>
</tr>
<tr>
<td>Readiness of the user community</td>
<td>“Is the user community ready to receive the new release?”</td>
</tr>
<tr>
<td></td>
<td>“Are the roll-out plans in place?”</td>
</tr>
</tbody>
</table>

- We cannot successfully exit the Construction Phase until we can resolve these questions
Organizing the Work

- Now that the architecture is stabilized, much of the work can be done in parallel
  - Components can be developed in parallel
  - Descriptions for the remaining use case flows can be written, designed and developed in parallel
  - As a result, project staffing peaks in the Construction Phase
- Development of components can be outsourced if desired
- The ability to develop in parallel based on a stable architecture is the main tactic for ensuring success
Common Problems in the Construction Phase

- Architectural Instability
- Requirements Instability
- Too much work, not enough time
- Not enough testing
- Too much rework
- Not enough rework
- The snow-plow effect
Managing Scope and Coping With Change

- Scope management is key to directing the project
  - Often at this point the work has taken longer than hoped for (architectural issues usually do) and some scope reduction is in order
  - Some alternative flows (representing “optional” behavior) may not need to be implemented in the first release
  - Not all use case flows may need to be fully described
    - “Simple” behavior can be prototyped and then implemented directly
    - Simple “data management” (CRUD) behavior
    - Simple menus and navigation

- Change is inevitable but disruptive:
  - Force ‘big changes’ early to reduce disruption
  - Requirements change ripples through all products
  - Use traceability to assess the impact of change
  - Form a framework for making decisions about change
Concluding the Construction Phase

- Usable solution available
- User community ready for receipt of release
- Transition planned
- Requirements complete and match product release
  - All use cases in scope completed and satisfied
  - All supplementary requirements met
- Completed System Test Results
- A completed product
  - (Design, Implementation, Documentation etc)
Risks Addressed in the Transition Phase

In the Transition Phase we focus on “Roll-out Risks”

<table>
<thead>
<tr>
<th>Risk</th>
<th>Questions to ask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery and installation</td>
<td>“Are the delivery mechanisms in place?”</td>
</tr>
<tr>
<td></td>
<td>“Can the product be installed and configured to meet the needs of the customer and users?”</td>
</tr>
<tr>
<td>Training and mentoring</td>
<td>“Can the users of the solution make use of the system capabilities?”</td>
</tr>
<tr>
<td>Operations and support</td>
<td>“Can the solution be supported in a “production” environment (IT operations)?”</td>
</tr>
<tr>
<td></td>
<td>“Can users of the solution get questions answered and problems resolved?”</td>
</tr>
</tbody>
</table>

We cannot successfully exit the Transition Phase until we can resolve these questions
Typical Issues Related to Transition

- Installation and Configuration are usually “architecturally significant” issues that require consideration in the Elaboration and Construction phases.
- In many systems, data conversion and production “cut-over” are complex problems that require significant attention.
- Few systems are so intuitively easy to use that they do not require some training in their use.
- Leading change is a complex area – there are many emotions triggered by induced change.
- Like any other machinery, systems must be maintainable.
- A process for maintaining the system must be put in place.
Activities in the Transition Phase

- The end of the Construction Phase marks the Initial Operational Capability (IOC) milestone
  - At this point the solution becomes available for deployment
  - In the case of simple applications, this marks the beginning of a “beta test”

- Most business-critical applications require quite a lot of work to deploy an application
  - Training of users and support staff in both system use and new or changed business processes
  - Data conversion
  - Transition to a “production” environment, often at multiple sites

- Now is not the time to “let down one’s guard”
  - If anything, precise attention to detail and the ability to successfully carry out a carefully planned transition is essential
Example Transition Phase Activities for the ACME Super ATM

- Installation of ATMs in “parallel” environments to assess suitability to deploy
- Installation of ATMs at “live test locations” to assess suitability for wide-scale deployment
- Training of sales people on new product capabilities
- Production of Marketing materials
  - Preparation of materials would have occurred earlier, in parallel with development activities in the Construction Phase
- Training of service and support staff in new ACME ATM capabilities
  - Technician “Boot Camps” in which they experience real-world problems
  - Certification of knowledge levels at the end of the “boot camps”
- Phased-in deployment of ATMs to “real world” locations, working closely with “early adopter” customers
- Monitoring of early deployments to allow fine-tuning of product roll-out
Concluding the Transition Phase

- Project completed
- Solution is in use
- Lessons learnt
- Responsibilities discharged
  - handed over to production support or the next project
- Next release planned

Project Viability Agreed

Selected Approach Proven

Usable Solution Available

Lifecycle Objective Milestone

Lifecycle Architecture Milestone

Initial Operational Capability Milestone

Product Release
Agenda

- What is RUP?
- Planning the Overall Project
- A Walk Through The RUP Lifecyle
- Summary
Using RUP in a PRINCE2 Environment

Adapted from Implementing RUP Within a PRINCE2 Environment: Laurence Archer, Oak IT, 2001
PRINCE2: The Four Management Levels

- Corporate or Programme Management
- Directing a Project
- Managing a Project
- Managing Product Delivery
- Project Board
- Project Manager
- Team Manager

Left: Adapted from PRINCE2. Figure 3.1 page 21
Right: Adapted from PRINCE2. Figure 14.1 page 196
RUP and PRINCE2: The Four Management Levels

Corporate or Programme Management

Directing a Project

Managing a Project

Managing Software Delivery

Corporate or Programme Management

Project Board

Project Manager

RUP Project Manager

Left: Adapted from PRINCE2. Figure 3.1 page 21
Right: Adapted from PRINCE2. Figure 14.1 page 196
RUP & PRINCE2: Take a holistic approach

- For software development the primary product is software
  - Not requirements, design, code or tests but working, verified software

- Treat the software as the product in the PRINCE2 product breakdown structure

- The RUP is a specialist management process that produces a single technical product: the software.

- Typically the software is only part of a larger business solution
## Understanding the Phases and Milestones

<table>
<thead>
<tr>
<th>Phase / Milestone</th>
<th>Incorrect Interpretation</th>
<th>Correct Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>High Level Requirements</td>
<td>Business Risks</td>
</tr>
<tr>
<td>Lifecycle Objectives (LCO)</td>
<td>Planning completed</td>
<td>Project Viability Agreed</td>
</tr>
<tr>
<td>Elaboration</td>
<td>Detailed requirements and / or design</td>
<td>Architectural / technical risks</td>
</tr>
<tr>
<td>Lifecycle Architecture (LCA)</td>
<td>Specification completed</td>
<td>Selected Approach Proven</td>
</tr>
<tr>
<td>Construction</td>
<td>Implementation and development; team testing</td>
<td>Logistical risks (the risk of not getting all the work done)</td>
</tr>
<tr>
<td>Initial Operational Capability (IOC)</td>
<td>Coding completed</td>
<td>Usable Solution Available</td>
</tr>
<tr>
<td>Transition</td>
<td>Acceptance Testing</td>
<td>Solution roll-out (delivery) risks</td>
</tr>
<tr>
<td>Product Release (PR)</td>
<td>Product available/deployed</td>
<td>Project Completed</td>
</tr>
</tbody>
</table>
The Lifecycle of a Use Case

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Development</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identified</td>
<td>Analyzed</td>
<td>Verified</td>
</tr>
<tr>
<td>Authored</td>
<td>Designed</td>
<td>Accepted</td>
</tr>
<tr>
<td>Agreed</td>
<td>Implemented</td>
<td></td>
</tr>
</tbody>
</table>
## The Evolution of the Use Case Model

<table>
<thead>
<tr>
<th>Use-Case State</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identified</td>
<td>60%</td>
<td>&gt; 80%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Outlined</td>
<td>50%</td>
<td>20% - 60%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Authored</td>
<td>10%</td>
<td>40% - 80%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Analysed</td>
<td>&lt; 10% *</td>
<td>20% - 40 %</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Designed, Implemented and Verified</td>
<td>&lt; 5% *</td>
<td>&lt; 10%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

* A small percentage may be addressed for proof of concept purposes.

Source: Adapted from The Unified Software Development Process, Jacobson et al (page 358).
Measuring Progress with Use Cases

Understanding how a project's use cases are evolving across the project lifecycle allows a project's progress to be measured.
Summary

ñ Each Phase is driven by a different set of forces
   ß Inception – Business risks
   ß Elaboration – Technical risks
   ß Construction – Project logistical risks
   ß Transition – Solution roll-out risks

ñ Each phase needs to be managed a little differently
   ß Different estimation approaches are used for each phase
   ß Each phased requires a different mixture of skills and levels of resources; it is not unreasonable to expect that different teams may staff each phase so long as there is a continuity of vision and expertise across phases

ñ Be rigorous about phase-end milestones
   ß Do not move on to the next phase until you have met the milestone objectives
   ß Don’t be pressured by the schedule into “declaring victory” – you will pay for it later!
RATIONAL USER CONFERENCE 2003

FROM ideas TO results

QUESTIONS
Additional Resources

- **Use Case Modeling** by Kurt Bittner and Ian Spence, Addison-Wesley 2002
  - Provides supporting information on working with use cases, as well as example use-case model surveys and use-case descriptions

- **Software Project Management** by Walker Royce, Addison-Wesley 1998
  - Provides background information on conventional and iterative project management strategies

- “Managing Iterative Software Development with Use Cases; Part I: The Lifecycle of an Iterative Project” by Kurt Bittner and Ian Spence, article on The Rational Edge
  - [http://www.therationaledge.com/content/mar_03/f_iterativeUse_kb.jsp](http://www.therationaledge.com/content/mar_03/f_iterativeUse_kb.jsp)
  - Outlines the purpose, perspective and objectives for each phase in the RUP
Estimation Resources

- Information about Function Point Analysis can be found at [http://www.cosmicon.com](http://www.cosmicon.com) and in information about ISO 19761 at [http://www.lrgl.ugam.ca/cosmic-ffp/](http://www.lrgl.ugam.ca/cosmic-ffp/)

- Information about COCOMO II can be found at [http://sunset.usc.edu/research/COCOMOII/cocomo_main.html](http://sunset.usc.edu/research/COCOMOII/cocomo_main.html)
RATIONAL USER CONFERENCE 2003

FROM ideas TO results

Thank You

Ian Spence
ispence@uk.ibm.com
## Comparing Waterfall To Iterative

<table>
<thead>
<tr>
<th>Waterfall</th>
<th>Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk averse</td>
<td>Actively attacks risk</td>
</tr>
<tr>
<td>Subjective measurement of progress</td>
<td>Objective measurement of progress</td>
</tr>
<tr>
<td>Delays integration and testing</td>
<td>Continuous integration and testing</td>
</tr>
<tr>
<td>Nothing runs until the end</td>
<td>Something “runnable” produced every iteration</td>
</tr>
<tr>
<td>Difficulties at the end of the project</td>
<td>Difficulties at the start of the project</td>
</tr>
</tbody>
</table>