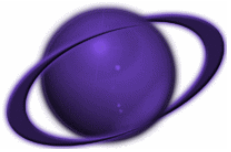
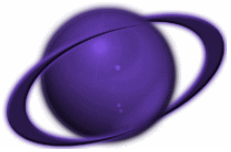


# Adopting The Unified Process: Can It Be Agile Or Is It Too Heavy?

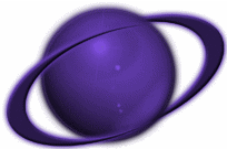
Presented by:
William F. Nazzaro
Principal, Nazzaro & Associates

bill@williamnazzaro.com
www.williamnazzaro.com



# Introduction

- The Rational Unified Process is being adopted by many Fortune 500 companies at an astounding pace
- However, many of these companies are unsure of which deliverables should be considered core and which deliverables may not be necessary
- Some view the Rational Unified Process as prescriptive and therefore all deliverables should be leveraged
- Others view the Rational Unified Process as a heavy-weight process that's burdensome, less than agile, and avoided at all costs
- Can RUP and Agile be used in the same sentence?



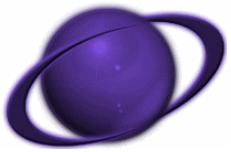
# What We Will Cover

- The Good and The Bad
- What is a Software Process?
- Tenets of a Good Software Process
- What is an Iteration and an Iterative Philosophy?
- Review a Process Map of Software Process
- Rational Unified Process (RUP) Overview
- Agile Manifesto and Agile Modeling  Principles
- RUP Artifacts to Maximize Agility
  - *Green Field*
  - *Maintenance*
  - *Hot Fix*
- Indicators of Not Being Iterative

# The Good And The Bad*

- Alan MacCormack (Harvard Business School) polled software IT executives to provide “good” and “bad” examples of software projects
- MacCormack rated the products based on market acceptance, expert quality rating, & productivity
- Executives considered good projects to be ones where
  - *Specifications were completed up front*
  - *Designs had been frozen*
  - *Projects were executed efficiently*
  - *Teams built what they set out to build*
- Executives considered bad projects to be ones where the final results were very different from the original goal
  - *"The people who were overseeing projects there assumed that the good projects were the ones that delivered to the spec. In fact, good projects are ones that deliver to the market.“*
- Interestingly, the "good" products were market failures and the “bad” products were a market success

* Google.com – www.stanford.edu Computer Science Course, CS194, 2005



# What is a Software Process?

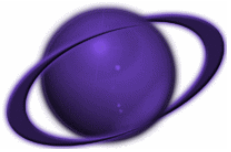
- "...a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget."

*Philippe Kruchten, The Rational Unified Process, 2000, pg. 17*

# Tenets of a "Good" Software Process?

- Driven by end-user requirements
- Emphasizes:
  - *Software as your primary goal*
  - *Risk mitigation (attack risk)*
  - *Architectural stability*
  - *Iterative and incremental development*
- Defines project roles and facilitates staff transitions between iterations & phases
- Accommodates our limited understanding at each iteration & phase
- Recommendations on:
  - *What to do*
  - *When to do it*
  - *Who should do it*
  - *How to do it*
  - *Why we do it*

- Provides a way to incrementally increase our understanding of the problem and our vision of the solution
- Adaptive, not prescriptive
- Lastly, it needs to be reasonable, pragmatic, and flexible

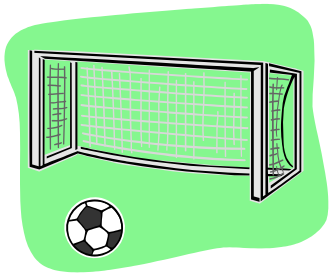


# What Is An Iteration?

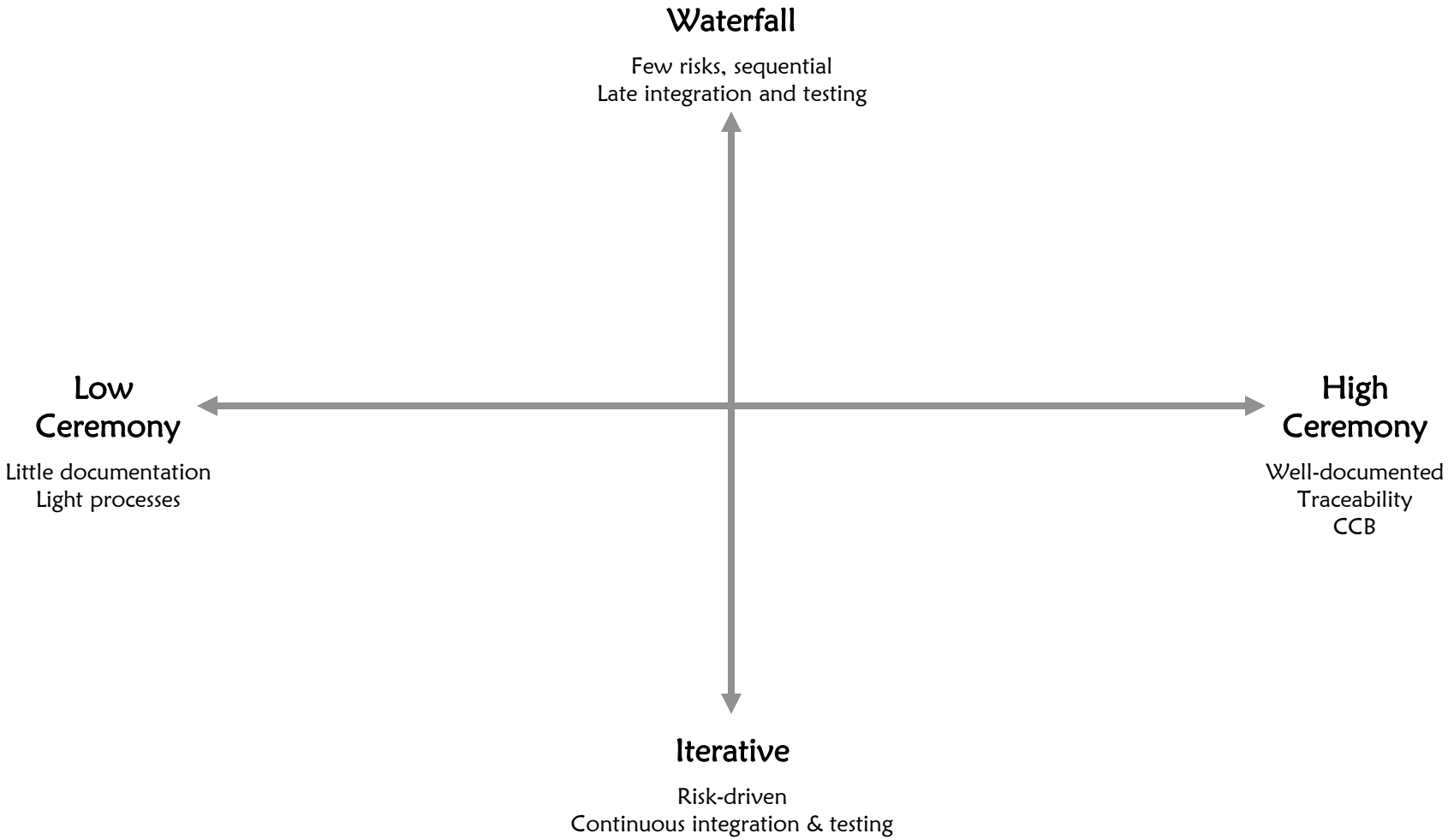
- A planned set of activities for a portion of the system under development
  - *Contains all lifecycle activities to some extent*
- Organized to
  - *Reduce risk*
  - *Meet business objectives*
- Produces a working piece of the system
  - *Work from multiple iterations results in a production release*

# Iterative Philosophy

- The nature of the iterative approach is:
  - *Map the problem, and the solution, into manageable "bites"*
  - *Build the system in "bites"*
  - *Always focus on the goal:*
    *delivering the proper, executable software*
  - *Always know where you are, where you are going, and when you will get there*
  - *Don't try to define everything up-front (admit it's impossible)*
  - *Don't try to answer everything before beginning*
  - *Admit that some answers cannot be found until you make some mistakes*
  - *Move forward so you can discover (quickly) what you overlooked*

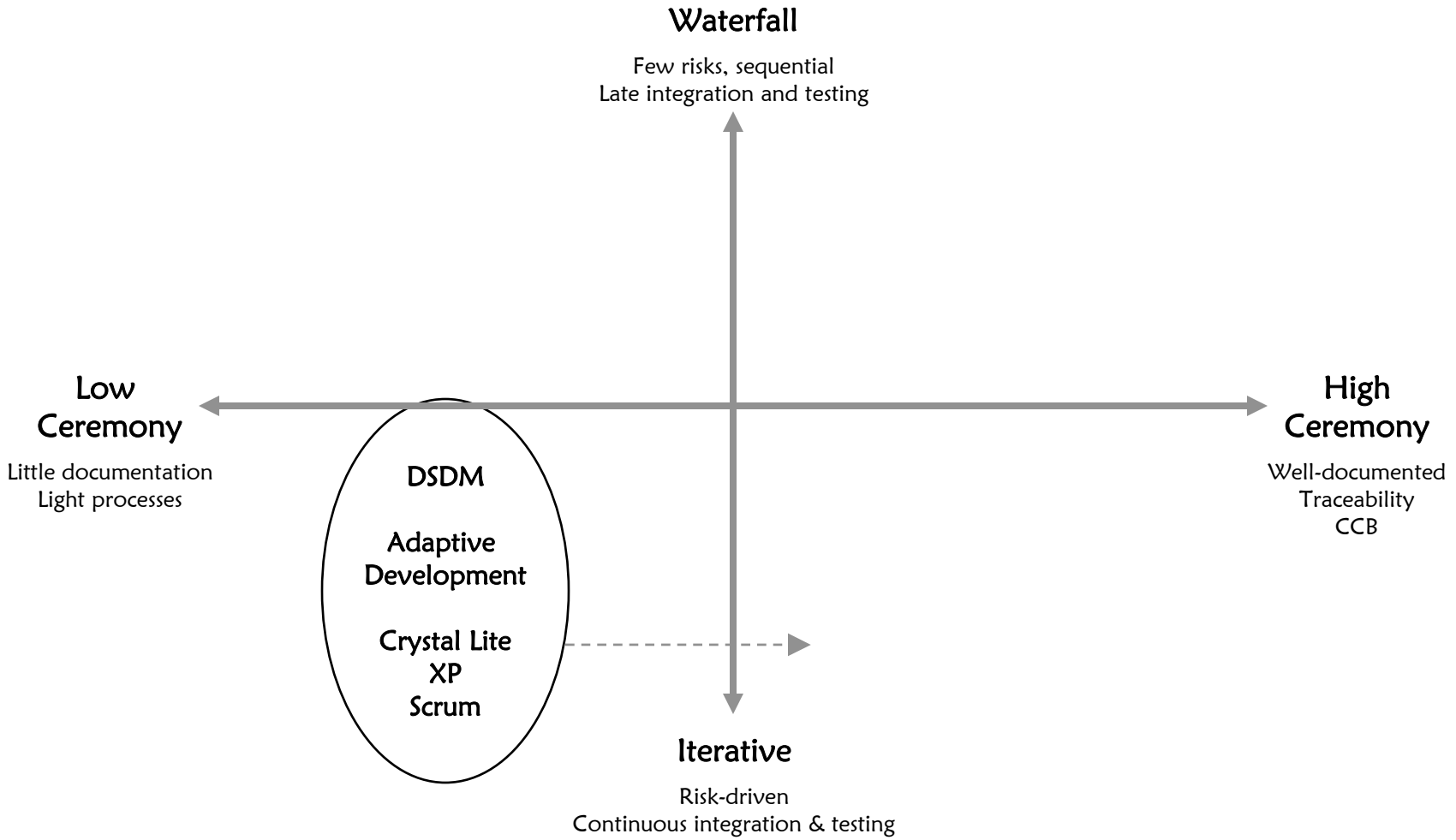


# Process Map



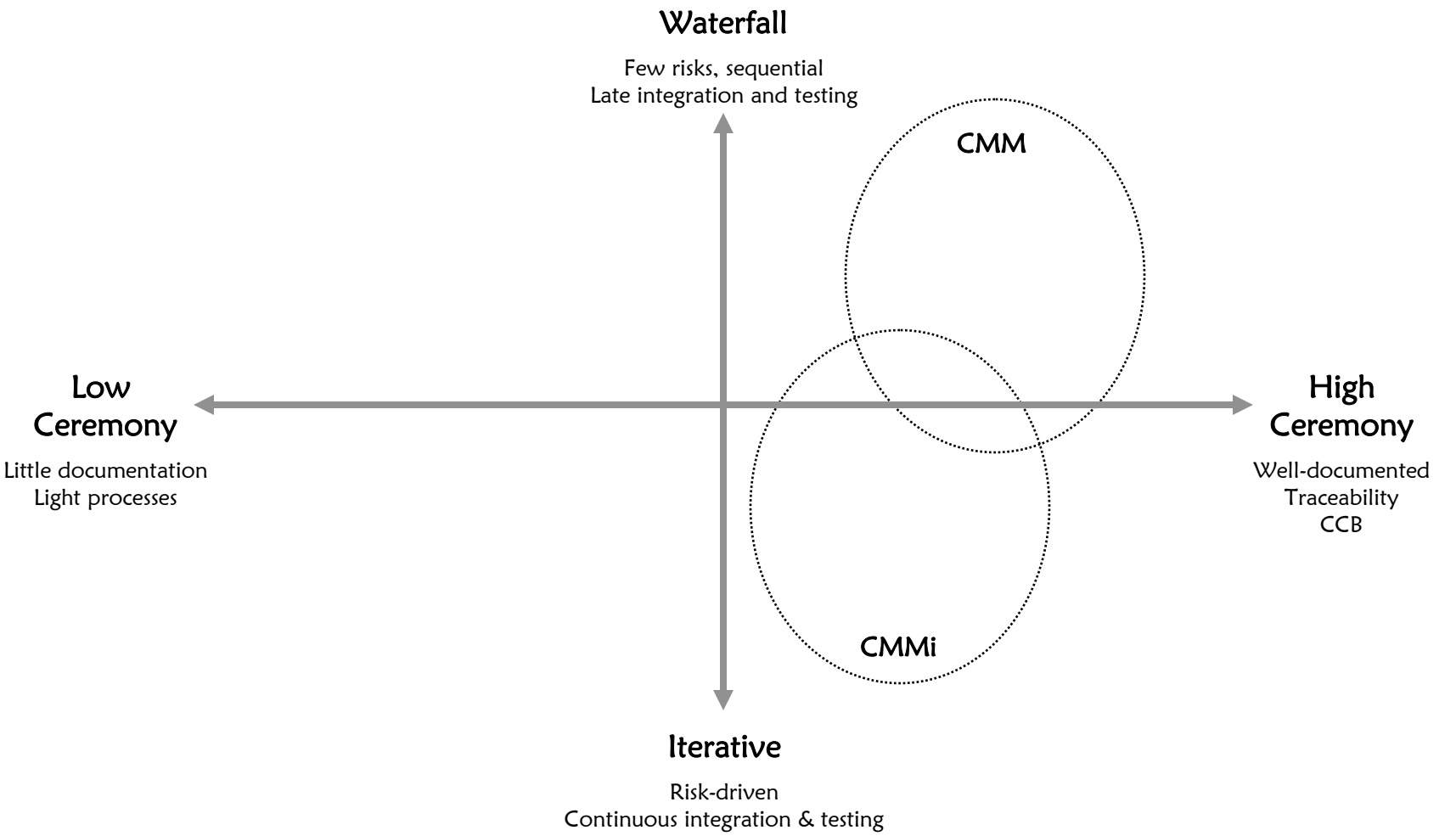

Per Kroll, *The Rational Unified Process Made Easy*, 2003, pg. 51

# Agile Processes



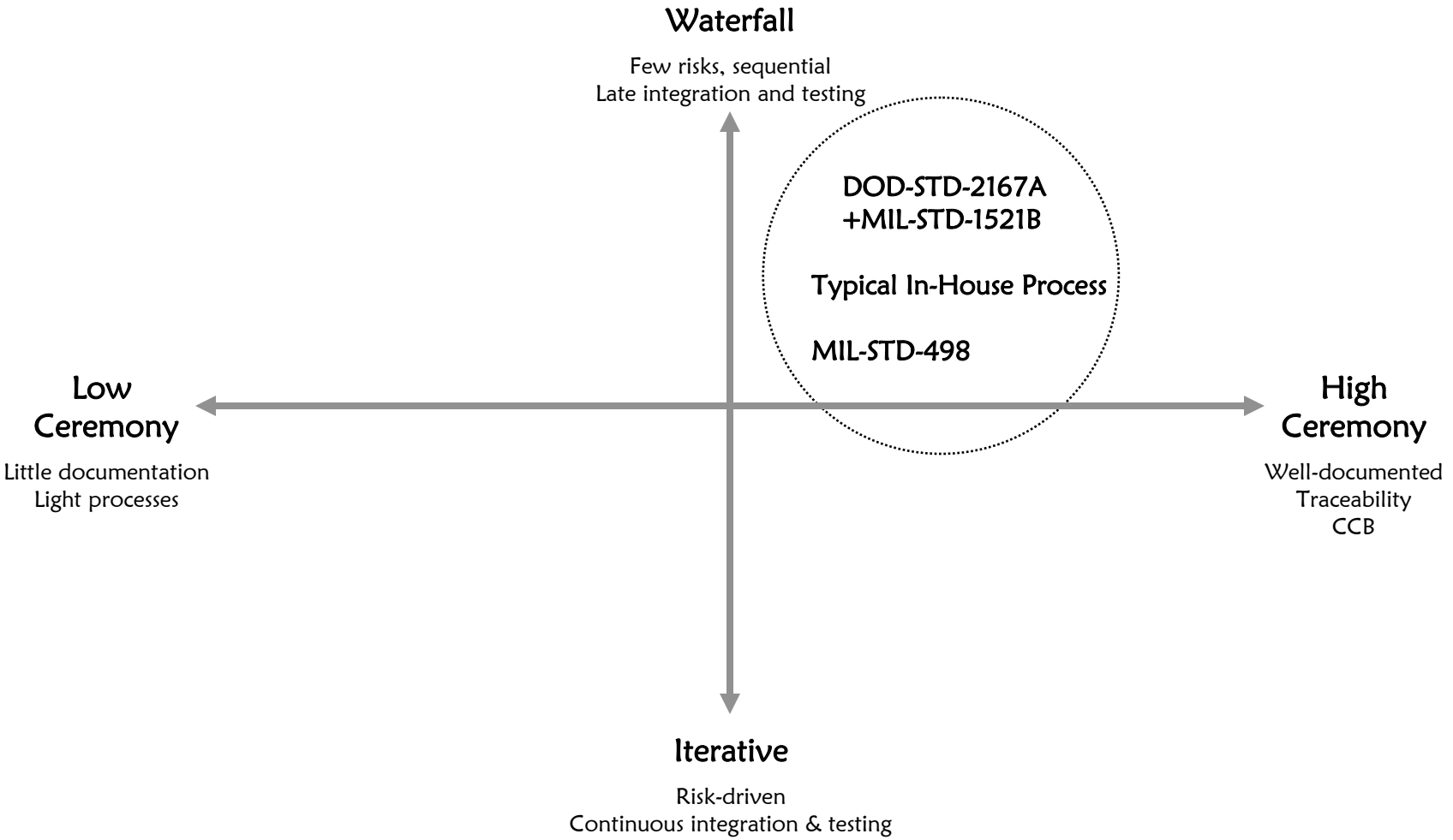

Per Kroll, *The Rational Unified Process Made Easy*, 2003, pg. 53

# SEI CMM and SEI CMMi



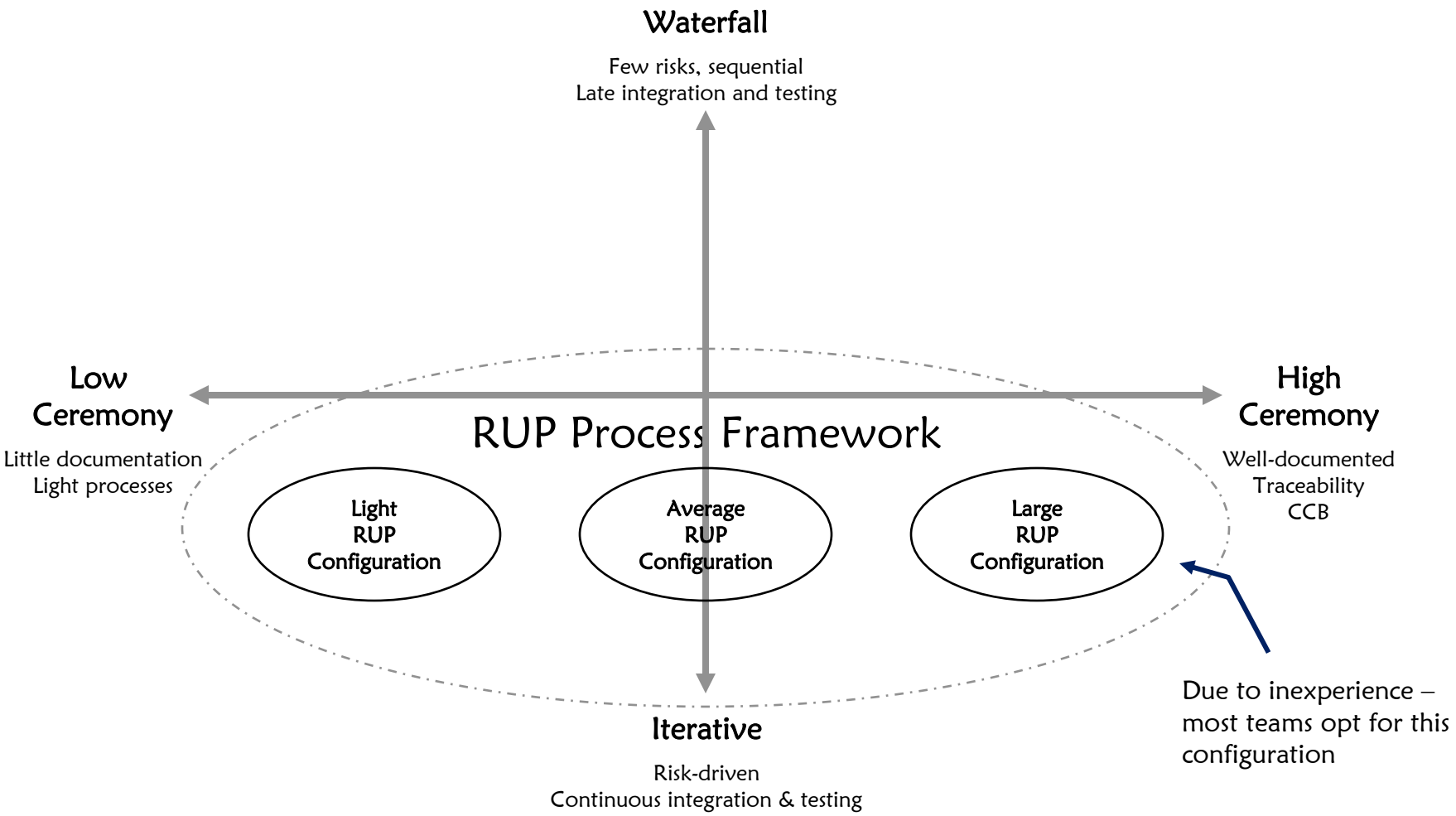

Per Kroll, *The Rational Unified Process Made Easy*, 2003, pg. 56

# DOD Standards

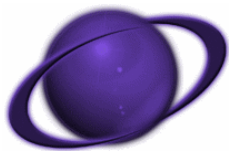



Per Kroll, *The Rational Unified Process Made Easy*, 2003, pg. 57

# Rational Unified Process (RUP) Framework



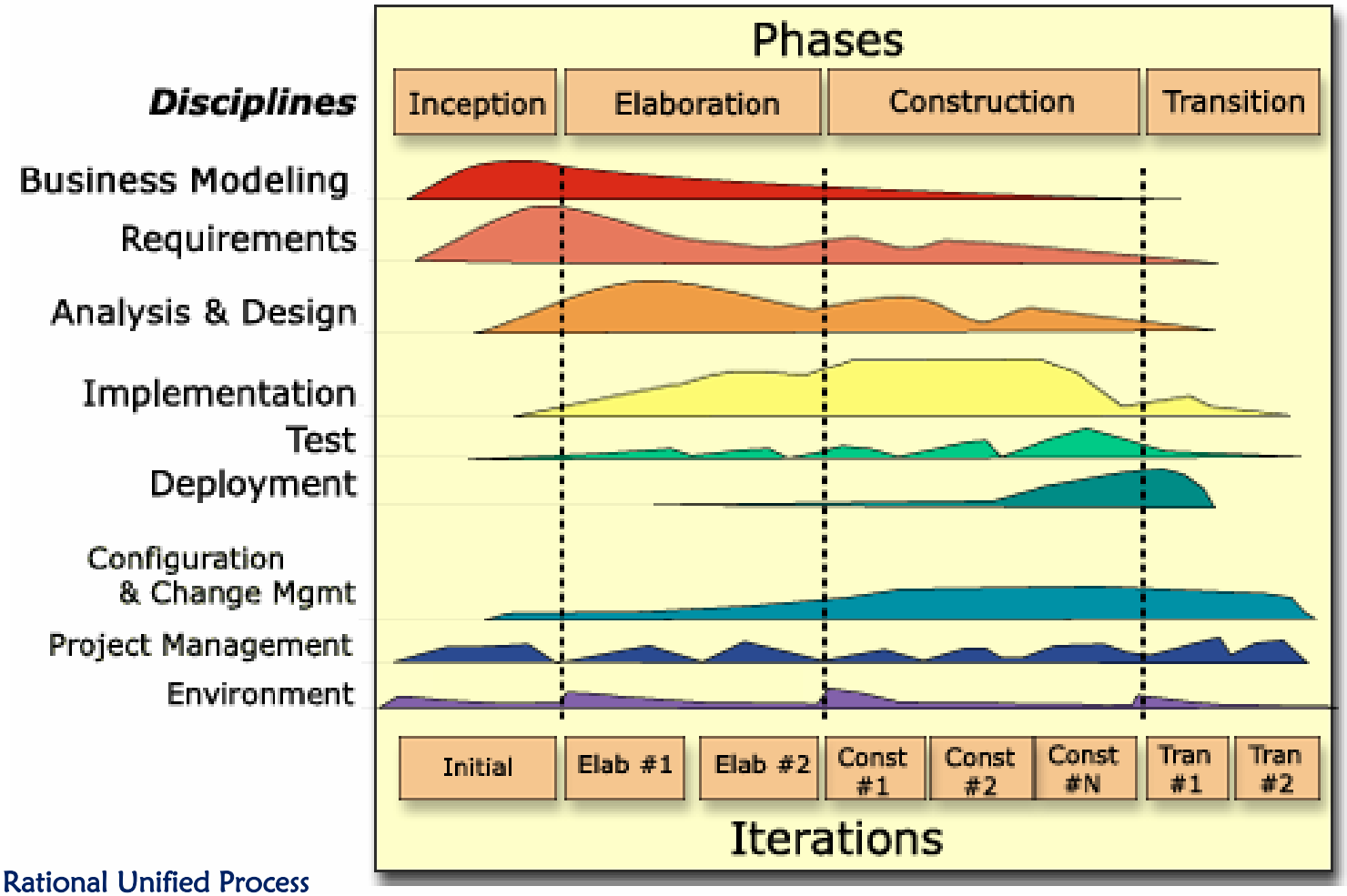

Per Kroll, *The Rational Unified Process Made Easy*, 2003, pg. 58



# Rational Unified Process (RUP): Five Defining Characteristics

- Iterative
  - *You do the same activities in small pieces, over and over*
- Incremental
  - *You gain a bit more understanding, and add a bit more solution, at each iteration*
- Risk-Focused
  - *You address risk early and often, before developing the easy, "low hanging fruit" of the system*
- Use Case (i.e. requirements) Driven
  - *The goal is established by the requirements; and the operational requirements are captured in Use Cases*
- Architecture-centric
  - *Architectural stability is emphasized over software design details*

# RUP: Brief Overview

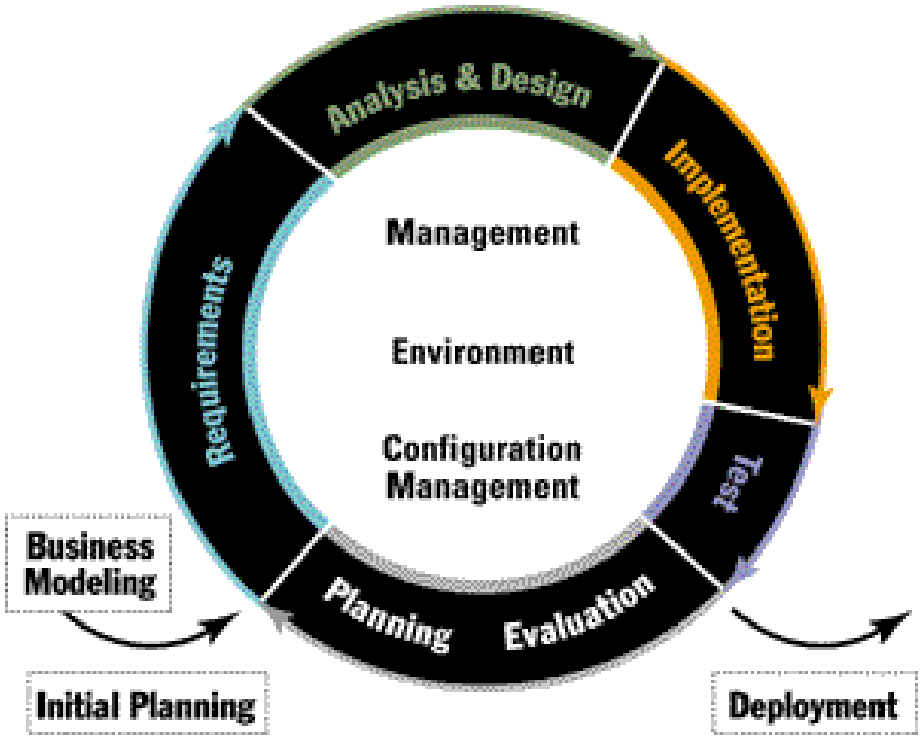



From Rational Unified Process version 2003.06.00.

# RUP: The Iterative Lifecycle

In each iteration, we do a cycle of:

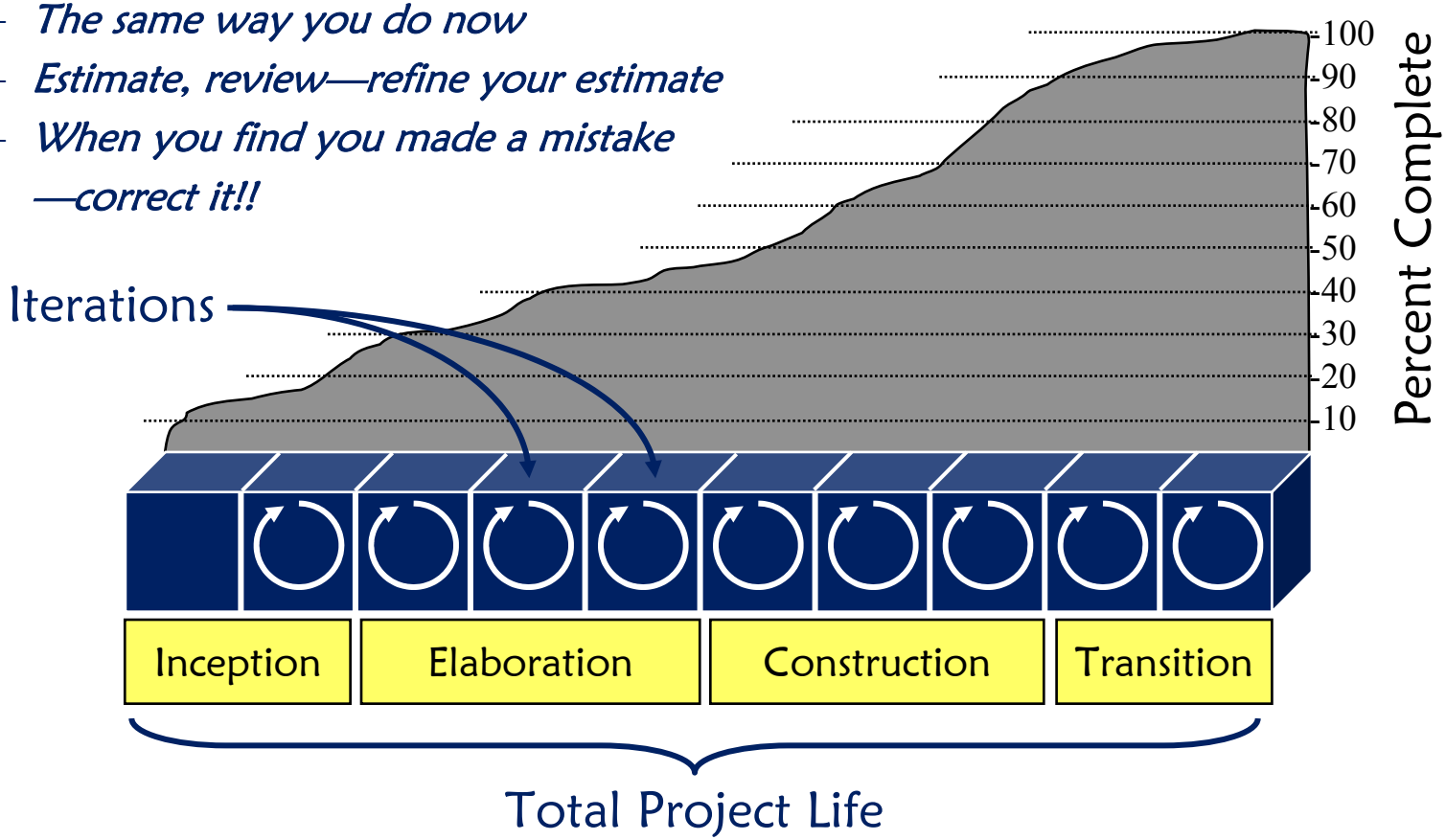
- *Planning*
- *Business Modeling*
- *Requirements*
- *Analysis & Design*
- *Implementation*
- *Test*
- *Deployment*
- *Evaluate ....*




From www.rational.com.
Copyright Rational Software Corp.

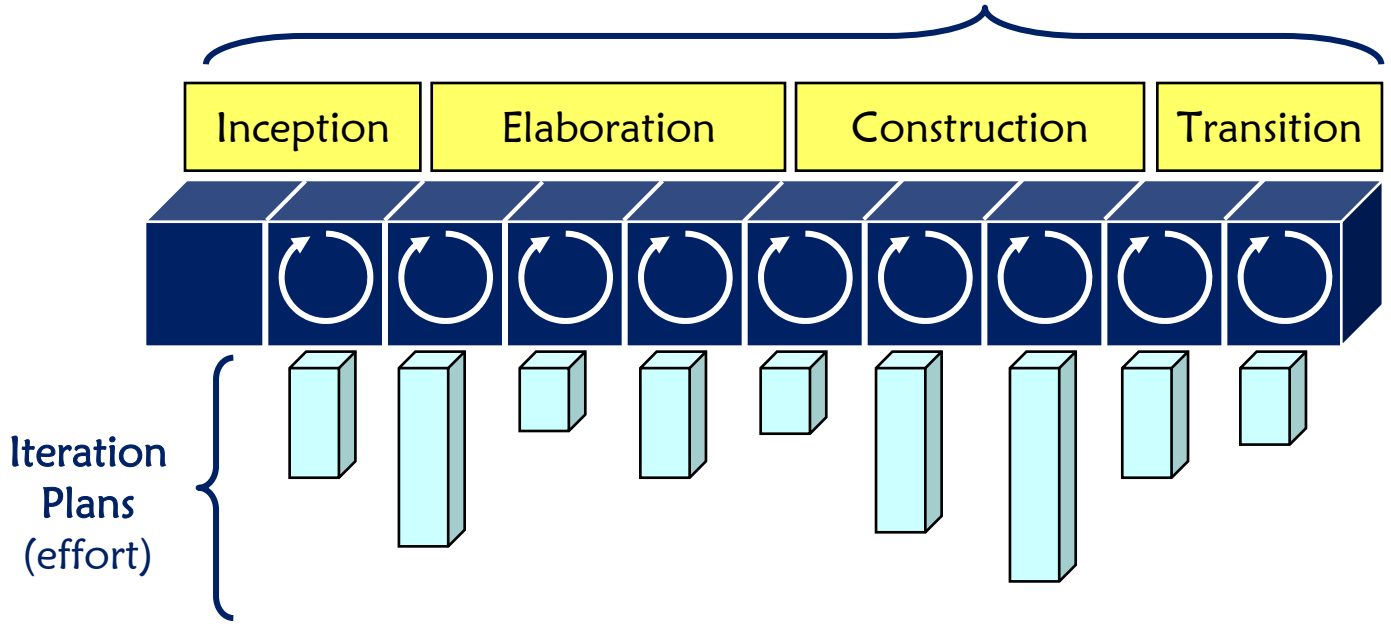
# Budgeting & Planning: An Iterative Process

- How do I budget a project in this iterative approach?
  - *The same way you do now*
  - *Estimate, review—refine your estimate*
  - *When you find you made a mistake —correct it!!*

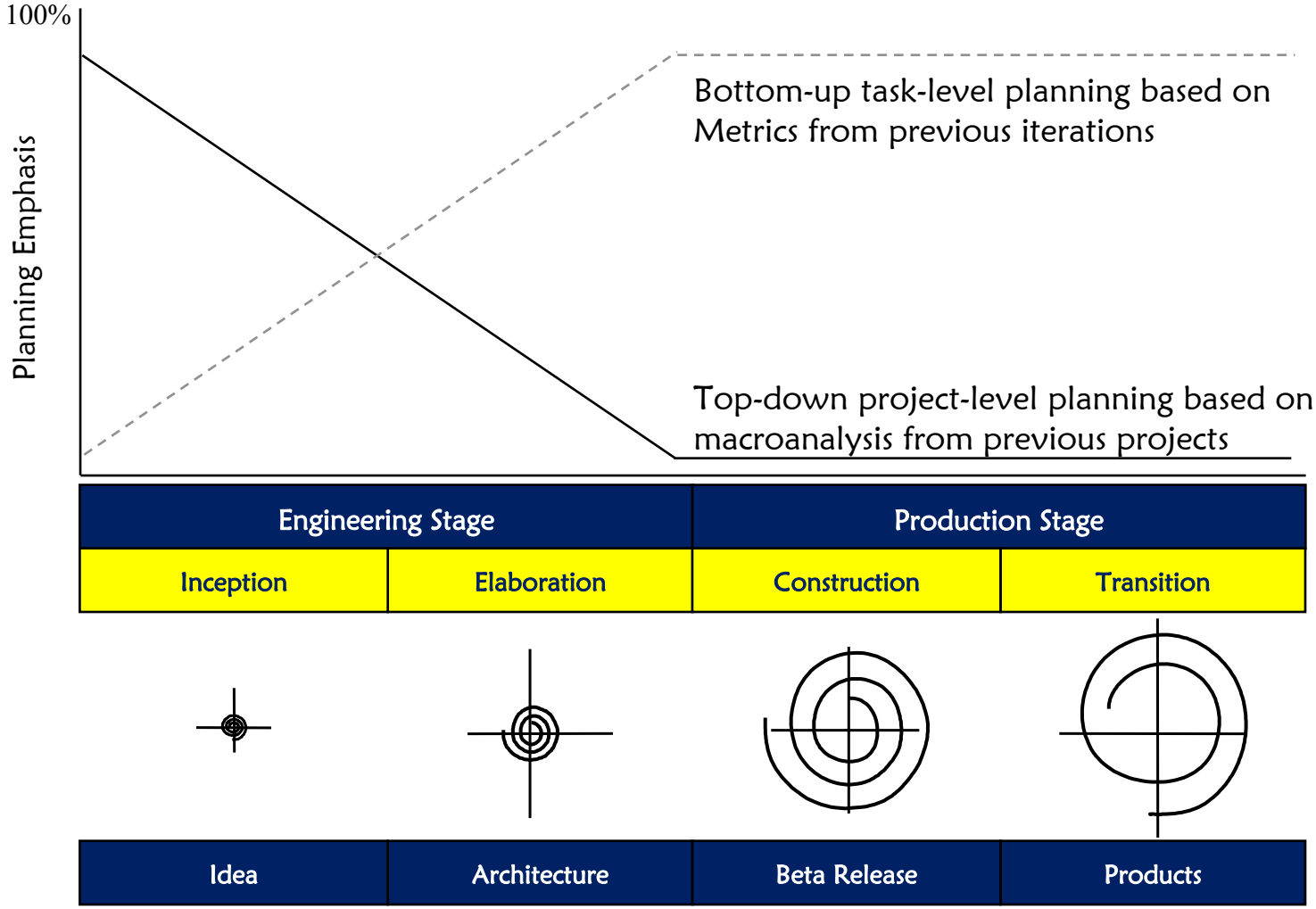


# Budgeting & Planning: An Iterative Process

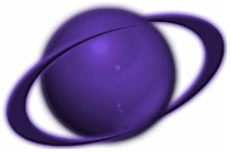
- How do I plan a project in this iterative approach?
  - *Plan "in the large" as you do now for the entire project*
    - *Resources*
    - *Features*
    - *Cost, etc.*
  - *Plan "in the small" for each iteration*



# RUP: Iteration Planning Process




Adapted from Walker Royce, *Software Project Management*, 2003, pg. 151



# The Unified Process:
# Can It Be Agile Or Is It Too Heavy?

Hmmm…. Let's See

# 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** over processes and tools

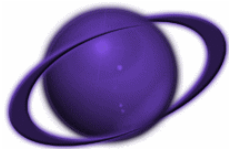
**Working software over** comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

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

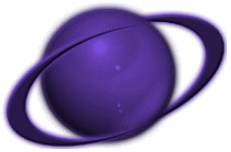
* www.agilemanifesto.org



# Agile Modeling Principles*

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

* Scott Ambler, *Are You Agile or Fragile?*, 2003



## Core RUP Artifacts to Maximize Agility

# Gotcha!!!

## But I Do Have Recommendations

# Rational Unified Process: Full Artifact List – 1

- Project Management Artifact Set (11)
  - *Deployment Plan*
  - *Business Case*
  - *Risk List*
  - *Software Development Plan*
    - *Quality Assurance Plan*
    - *Risk Management Plan*
    - *Measurement Plan*
    - *Product Acceptance Plan*
    - *Problem Resolution Plan*
  - *Work Order*
  - *Project Measurements*
  - *Issues List*
  - *Iteration Plan*
  - *Iteration Assessment*
  - *Status Assessment*
  - *Review Record*
- Requirements Artifact Set (10)
  - *Software Requirements*
  - *Vision Document*
  - *Glossary*
  - *Stakeholder Requests*
  - *Storyboard*
  - *Software Requirements Specification*
  - *Supplementary Specifications*
  - *Use Case Model*
  - *Requirements Management Plan*
  - *Requirement Attributes*

RUP Emphasizes Activities Over Artifacts

From Rational Unified Process version 2003.06.00.

# Rational Unified Process: Full Artifact List – 2

- Analysis & Design Artifacts Set (9)
  - *Software Architecture Document*
  - *Architectural Proof-of-Concept*
  - *Deployment Model*
  - *Reference Architecture*
  - *Design Model*
  - *Analysis Model*
  - *User-Interface Prototype*
  - *Navigation Map*
  - *Data Model*
- Implementation Artifact Set (4)
  - *Build*
  - *Implementation Model*
  - *Integration Build Plan*
  - *Developer Test*
- Test Artifact Set (15)
  - *Test Strategy*
  - *Test Results*
  - *Test-Idea List*
  - *Test Suite*
  - *Test Log*
  - *Test Plan*
  - *Test Data*
  - *Test Script*
  - *Test Case*
  - *Test Environment Configuration*
  - *Workload Analysis Model*
  - *Test Evaluation Summary*
  - *Test Interface Specification*
  - *Test Automation Architecture*
  - *Defect List*

RUP Emphasizes Activities Over Artifacts

From Rational Unified Process version 2003.06.00.

# Rational Unified Process: Full Artifact List – 3

- Deployment Artifact Set (3)
  - *Product*
    - » *Deployment Unit(s)*
  - *End-User Support Material*
  - *Manual Style Guide*
- Configuration Artifact Set (5)
  - *Change Request*
  - *Project Repository*
  - *Workspace*
  - *Configuration Management Plan*
  - *Configuration Audit Findings*
- Environment Artifact Set (4)
  - *Development Process*
    - » *Development Case*
    - » *Project Specific Templates*
    - » *Project Specific Guidelines*
  - *Development Infrastructure*
  - *Tools*
  - *Development Organization Assessment*

RUP Emphasizes Activities Over Artifacts

From Rational Unified Process version 2003.06.00.

# Agile Philosophy

- Emphasizes
  - *Maximizing stakeholder value,*
  - *Flexibility,*
  - *Communications,*
  - *Trust,*
  - *Model to discover and communicate,*
  - *Keep artifacts that enable us to get to code*
- Agility asks us to examine artifact vs. project history
  - *Certain artifacts (I call transient deliverables or artifacts) were created to help us get to code, but they do not retain value once we have coded (e.g., sequence diagram, use case diagram)*
  - *Agility recognizes the resources we expend to update these transient deliverables – and asks "why should we update it?"*
- Remember, it's not what you plan to do, it's what you DELIVER

Proof Through Code Vs. Promise Through Documentation

# RUP: Recommended Artifacts* – Green Field

- Project Management (4)
  - *Risk List*
  - *Project Plan (broad & shallow)*
  - *Iteration Plan (narrow & deep)*
  - *Iteration Assessment*
- Requirements (2)
  - *Vision*
  - *Use Case Model*
- Analysis & Design (2)
  - *Software Architecture Document*
  - *Design Model*
    - *Class Diagram*
    - *Sequence Diagram*
    - *State Machine*
- Implementation (0)
  - *None*
- Test (3)
  - *Test Plan*
  - *Test Case*
  - *Defect List*
- Deployment (1)
  - *Product: Deployment Units*
- Configuration Management (3)
  - *Change Request*
  - *Project Repository*
  - *Configuration Management Plan*
- Environment (2)
  - *Development Case*
  - *Project Specific Guidelines*
    - *Coding Standards & Guidelines*

Always Review This List And Balance The Agile Manifesto And Agile Modeling Principles !!!

* Michael Hirsch, Making RUP Agile, 2004

# RUP: Recommended Artifacts – Maintenance

- Project Management (4)
  - *Risk List*
  - *Project Plan (broad & shallow)*
  - *Iteration Plan (narrow & deep)*
  - *Iteration Assessment*
- Requirements (1)
  - *Vision*
  - *Use Case Model (possible deltas)*
- Analysis & Design (1)
  - *Software Architecture Document*
  - *Design Model (possible deltas)*
    - *Class Diagram*
    - *Sequence Diagram*
    - *State Machine*
- Implementation (0)
  - *None*
- Test (3)
  - *Test Plan*
  - *Test Case*
  - *Defect List*
- Deployment (1)
  - *Product: Deployment Units (possible deltas)*
- Configuration Management (1)
  - *Change Request*
  - *Project Repository*
  - *Configuration Management Plan*
- Environment (0)
  - *Development Case*
  - *Project Specific Guidelines*
    - *Coding Standards & Guidelines*

Always Review This List And Balance The Agile Manifesto And Agile Modeling Principles !!!

# RUP: Recommended Artifacts – Hot Fix

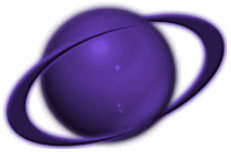
- Project Management (4)
  - *Risk List*
  - *Project Plan (broad & shallow)*
  - *Iteration Plan (narrow & deep)*
  - *Iteration Assessment*
- Requirements (0)
  - *Vision*
  - *Use Case Model (possible deltas)*
- Analysis & Design (0)
  - *Software Architecture Document*
  - *Design Model (possible deltas)*
    - *Class Diagram*
    - *Sequence Diagram*
    - *State Machine*
- Implementation (0)
  - *None*
- Test (2)
  - *Test Plan*
  - *Test Case*
  - *Defect List*
- Deployment (1)
  - *Product: Deployment Units (possible deltas)*
- Configuration Management (1)
  - *Change Request*
  - *Project Repository*
  - *Configuration Management Plan*
- Environment (0)
  - *Development Case*
  - *Project Specific Guidelines*
    - *Coding Standards & Guidelines*

Always Review This List And Balance The Agile Manifesto And Agile Modeling Principles !!!

# Indicators of Not Being Iterative

- Statements like
  - *"Let's get all the requirements done first and then we'll be iterative"*
  - *"Let's get an estimate with 90% accuracy (during Inception)"*
    - *"Estimates are Lies and Detailed Estimates Are Detailed Lies" ☺ **
  - *"I know the customer really needs that but it's not in our project plan"*
- Demand on Freezing Requirements and treating new understanding of Requirements as "Change Orders"
- Expecting
  - *Build Plan to be Frozen*
  - *Expecting the Project Plan to be built upfront and Frozen*
- Reporting % Complete on Model & Documents
  - *Really more indicative of not knowing what to measure*

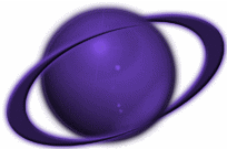
* Paul Glen, *Leading Geeks*, 2002



# The Unified Process: Can It Be Agile?

# YES!!!

Always Review Artifacts And Balance The Agile Manifesto And Agile Modeling Principles !!!



# Adopting The Unified Process: Can It Be Agile Or Is It Too Heavy?

Presented by:
William F. Nazzaro
Principal, Nazzaro & Associates

bill@williamnazzaro.com
www.williamnazzaro.com