Object Oriented

Analysis and Design

(OOA and OOD)

COMP 3711

Larman Chapter 1 – 3, 22

The World …….

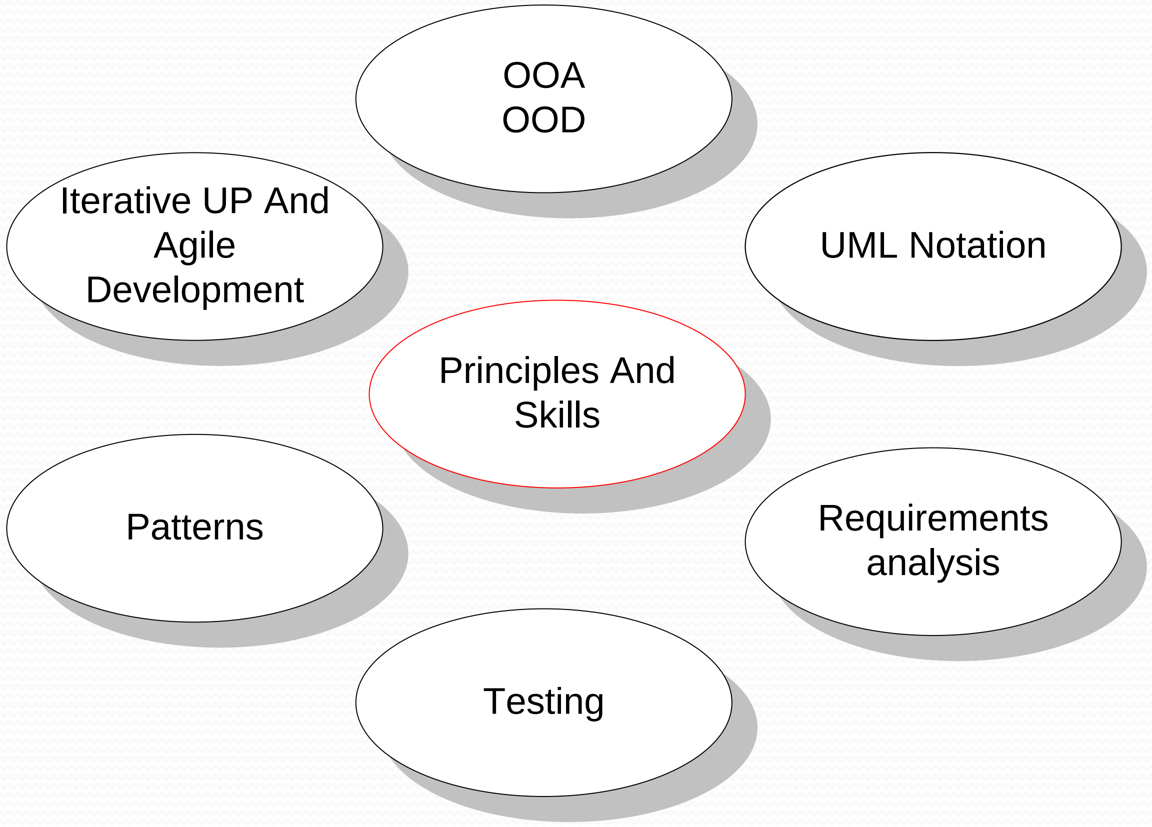
Describe

By

Abstract

Models

What is this course about?



Software System Characteristics

- Software system is *programs*, *documents*, and *data*.
- Software system is developed or engineered; it is not manufactured like hardware.
- Software system does not wear out, but it does *deteriorate*.
- Most software system is custom-built, rather than being assembled from existing components.

Nature of Software

- Non-material
- Ethereal
- Flexible
- Pliable
- Both strengths and weaknesses

Object-Oriented (OO)Approach

- Views software system as collection of interacting objects that work together to accomplish tasks
  - Objects - things in software system that can respond to messages.
  - No processes, programs, data entities, or files are defined – just objects.

Four Major Principles of OO

- Abstraction
- Encapsulation
- Polymorphism
- Inheritance

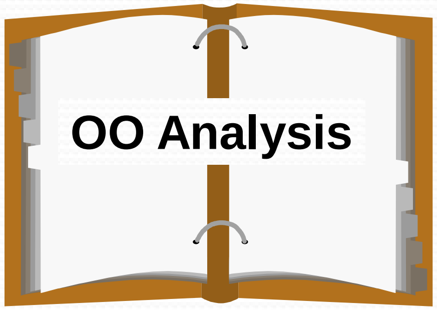
Four Major Principles of OO

Abstraction	The essential characteristics of an entity that distinguish it from all other kind of entities and thus provide crisply-defined boundaries relative to the perspective of the viewer.
Encapsulation	The physical localization of features (e.g., properties, behaviors) into a single black box abstraction that hides their implementation (and associated design decisions) behind a public interface. Encapsulation is also referred to as information hiding
Polymorphism	Polymorphism is the ability to define a single interface with multiple implementations.
Inheritance	The mechanism that makes generalization possible; a mechanism for creating a new class using an existing classes as a foundation.

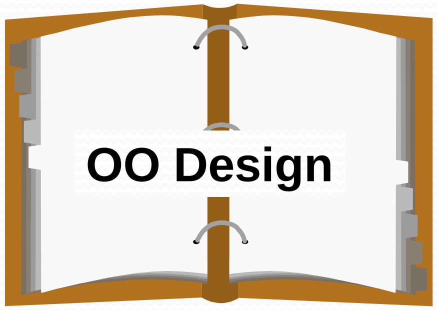
Object-Oriented Approach

- Object-Oriented Analysis (OOA)
  - Defines types of objects that do work of system
  - Shows how objects interact with users to complete tasks
- Object-Oriented Design(OOD)
  - Defines all additional types of objects necessary to communicate with people and devices
  - Shows how objects interact to complete tasks
  - Refine the definition of each type of object
- Object-Oriented Programming (OOP)
  - Writing programming statements with object features
- Object-Oriented Database (OODBMS)
  - Storing data as persistent objects in the database

OO Approach In Development

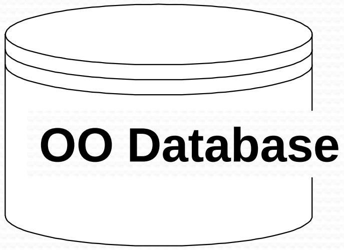


Analyse current situation and new requirement



Formulate (eleborate) conceptual solutions

Build and construct



Store data objects

Software Development Process

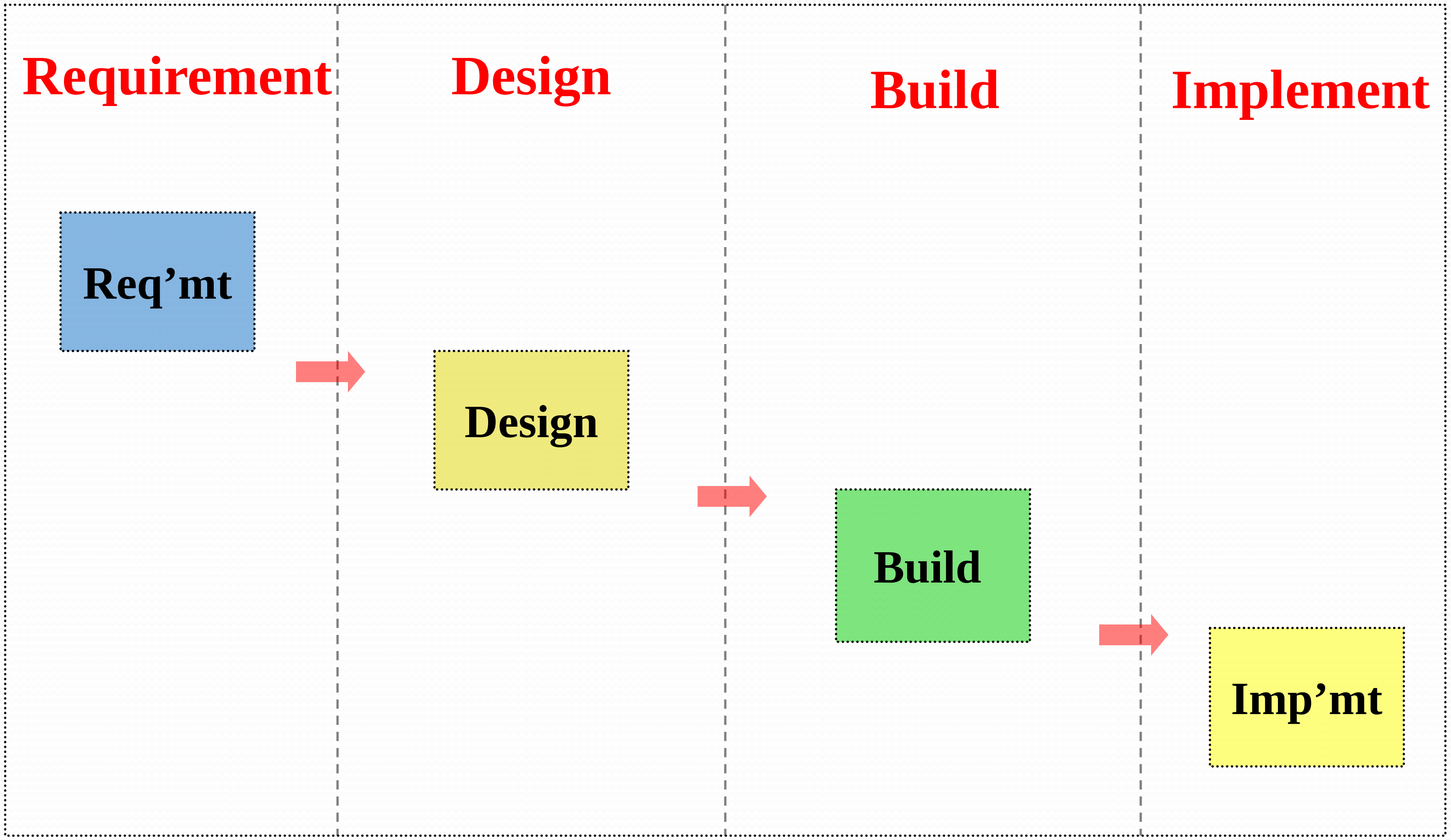
- Coherent sets of activities for specifying, designing, implementing and testing software systems

With a coherent set of tools

Systems Development Life Cycle

- Systems development life cycle (SDLC)
  - Provides overall framework for managing software system development process
- All projects use some variation of SDLC

A typical software project development methodology

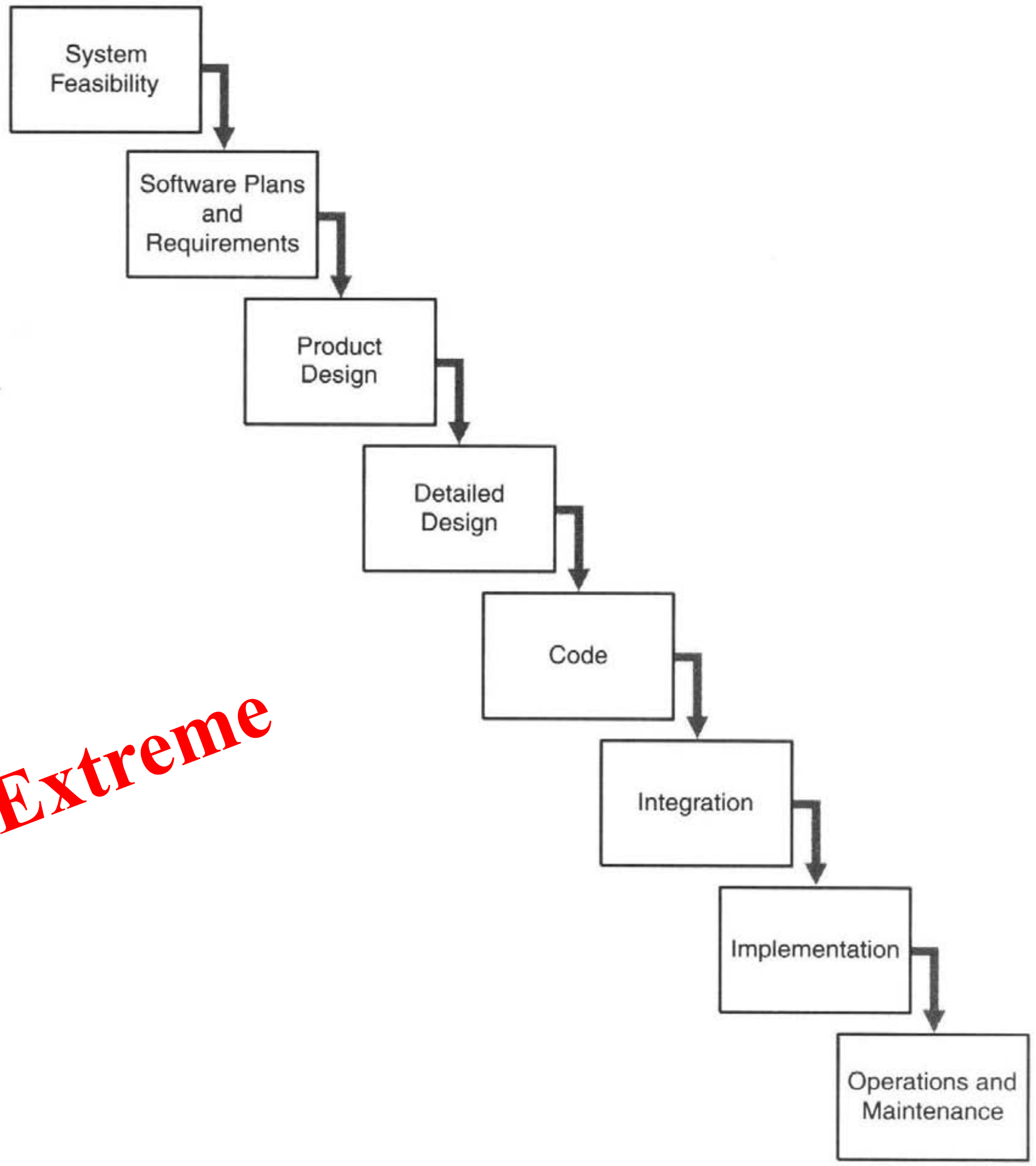


Example: Water Fall Model

Waterfall Approach

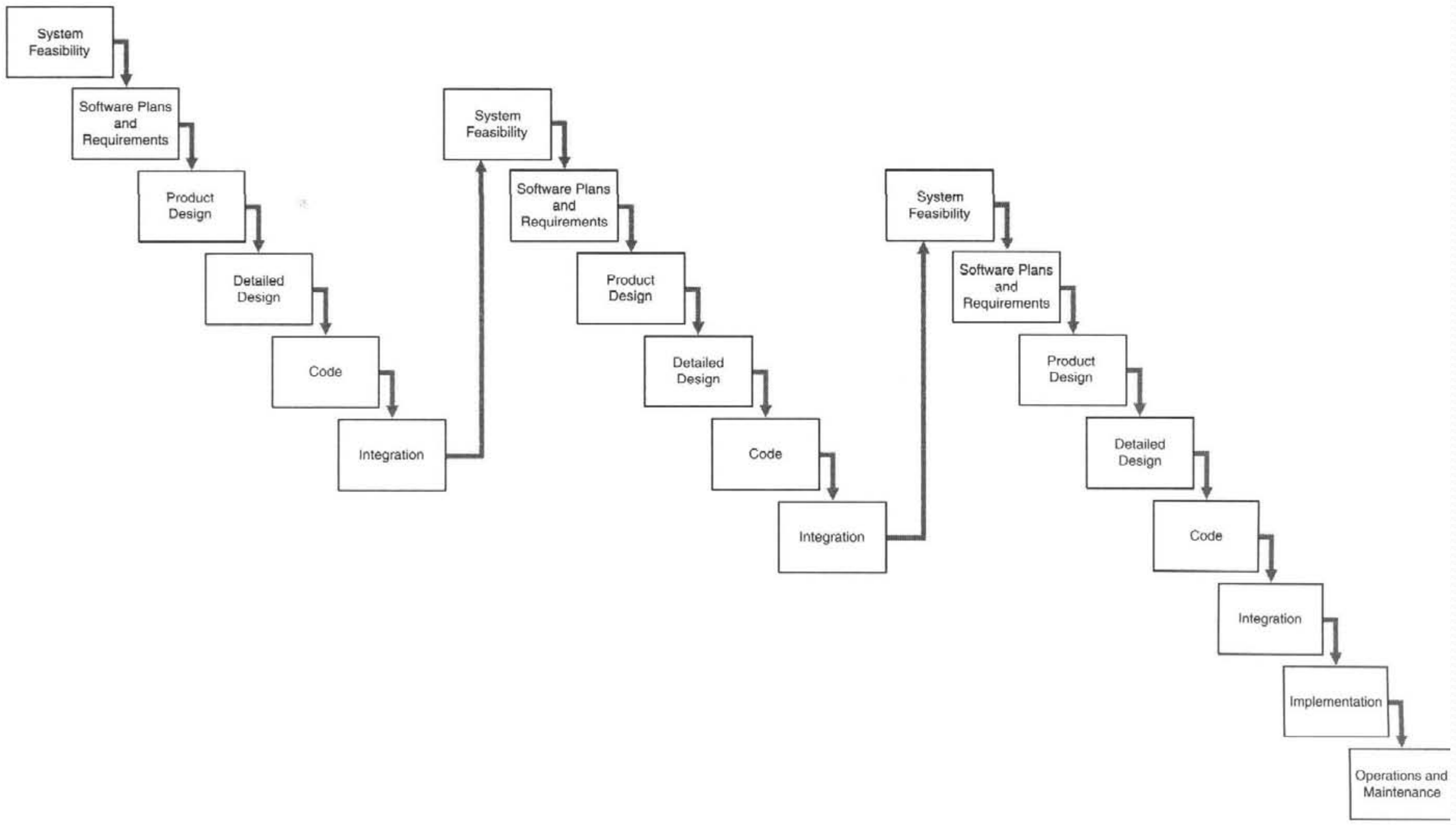
- Each phase falls into next phase
- Freeze planning specifications before analysis
- Freeze analysis specifications before design
- Once go over the waterfall for each phase, do not go back

Very Traditional Waterfall Approach



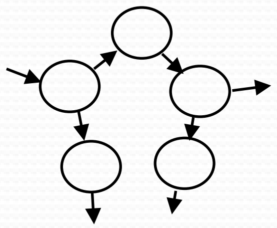
One Extreme

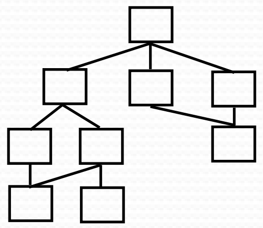
Incremental Waterfall Approach



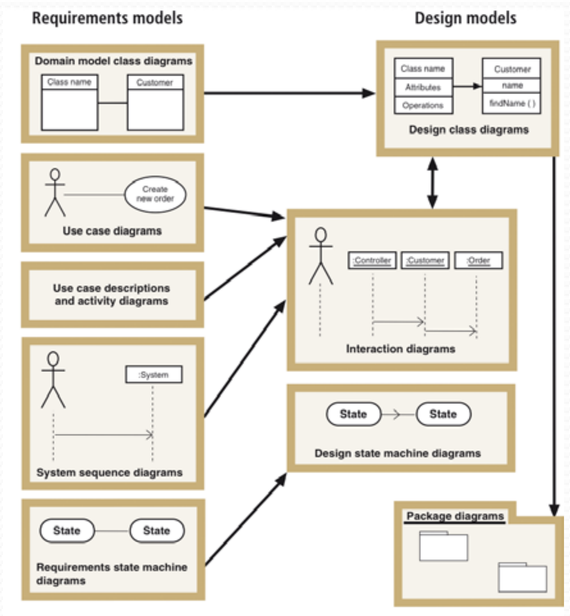
New Approach To SDLC

- Traditional approach
  - structured systems development and information engineering





- Object-oriented approach
  - Views information system as collection of interacting objects that work together to accomplish tasks with different approach to analysis, design, and programming



Why OO Approach ?

- Development time may be longer but reusability pays back in the long run

- Facilitates the construction of software that are easier to:

1. Understand
2. Maintain
3. Extend

Object-Oriented System Development Methodology

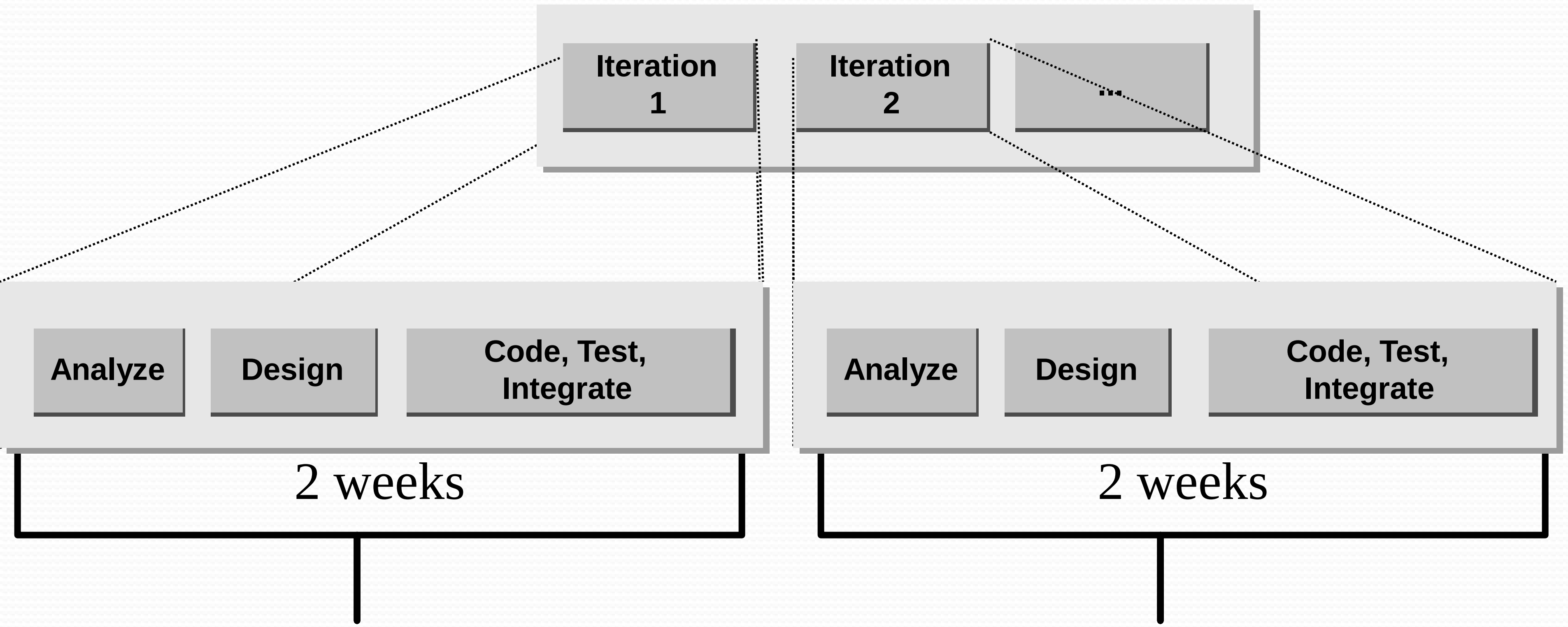
- ~~RUP - Rational Unified Process~~
  - ~~Rational Rose~~

- UP - Unified Process
  - IBM has taken over Rational Rose

Iterative Development

- Small steps, feedback and refinement.
- Iteration - Work activities are repeated
  - Each iteration refines previous result
  - Approach assumes no one gets it right the first time
  - There are a series of mini projects for each iteration

Iterative Development



Iterative Development

Iteration 1

Introduces just those analysis and design skills related to iteration one.

Iteration 2

Additional analysis and design skills introduced.

Iteration 3

Likewise.

Larman 3.2

Benefits Of Iterative

1. Less project failure
2. Early mitigation of risks
3. Early visible progress
4. Early feedback
5. Managed complexity
6. Learn as you go

Larman 2.14 history

Unified Process - UP

Requirement	Design	Develop	Implement
Inception	Elaboration	Construction	Transition

IP Supports Iterative Develop Methodology

Unified Process - UP

Requirement	Design	Develop	Implement
Inception	Elaboration	Construction	Transition
Approximate vision, business case, scope	Refine vision, core architecture, refine scope	Implementation of low risk core architecture, refine scope	Beta test and deployment

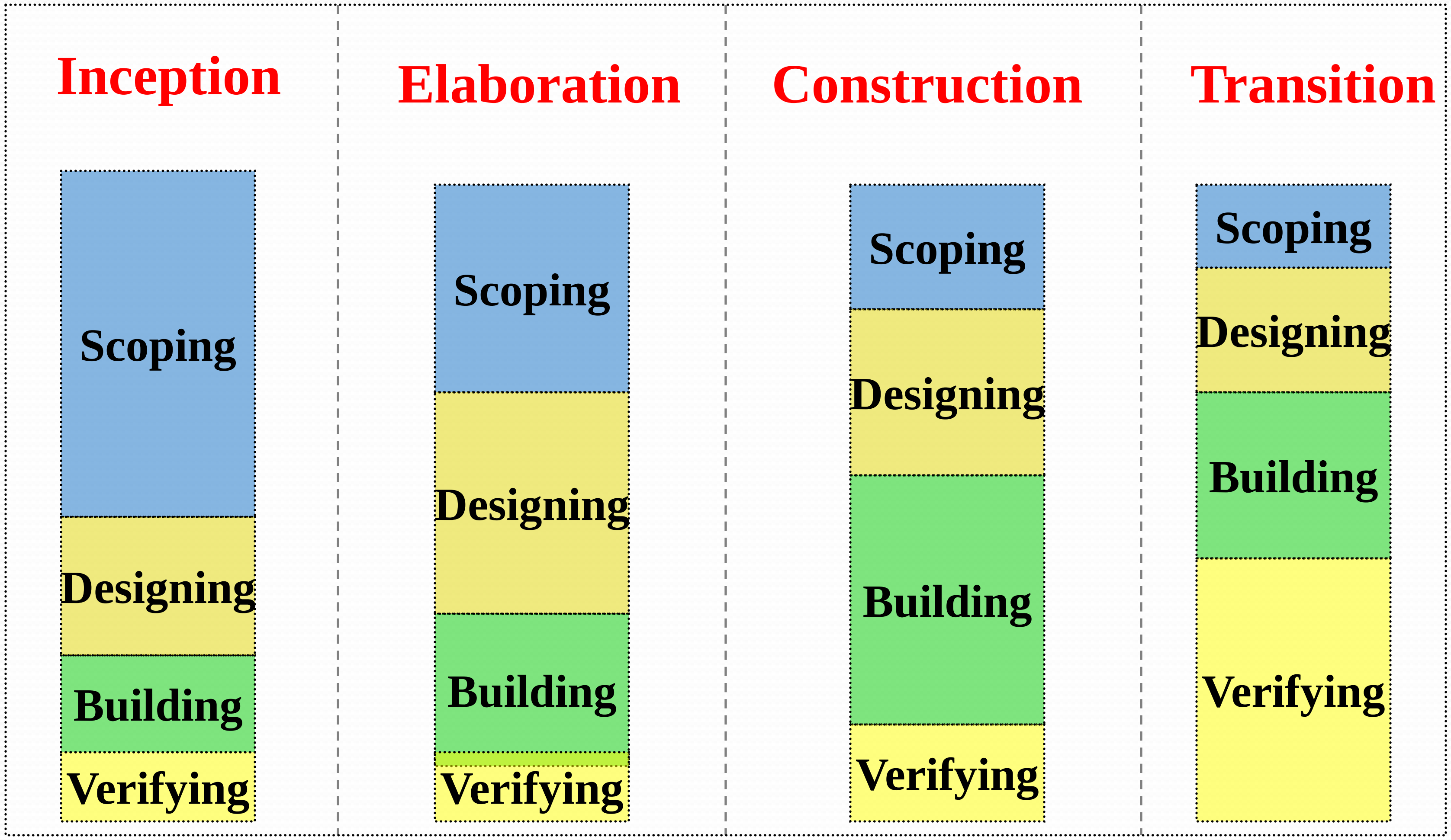
Iterative Development

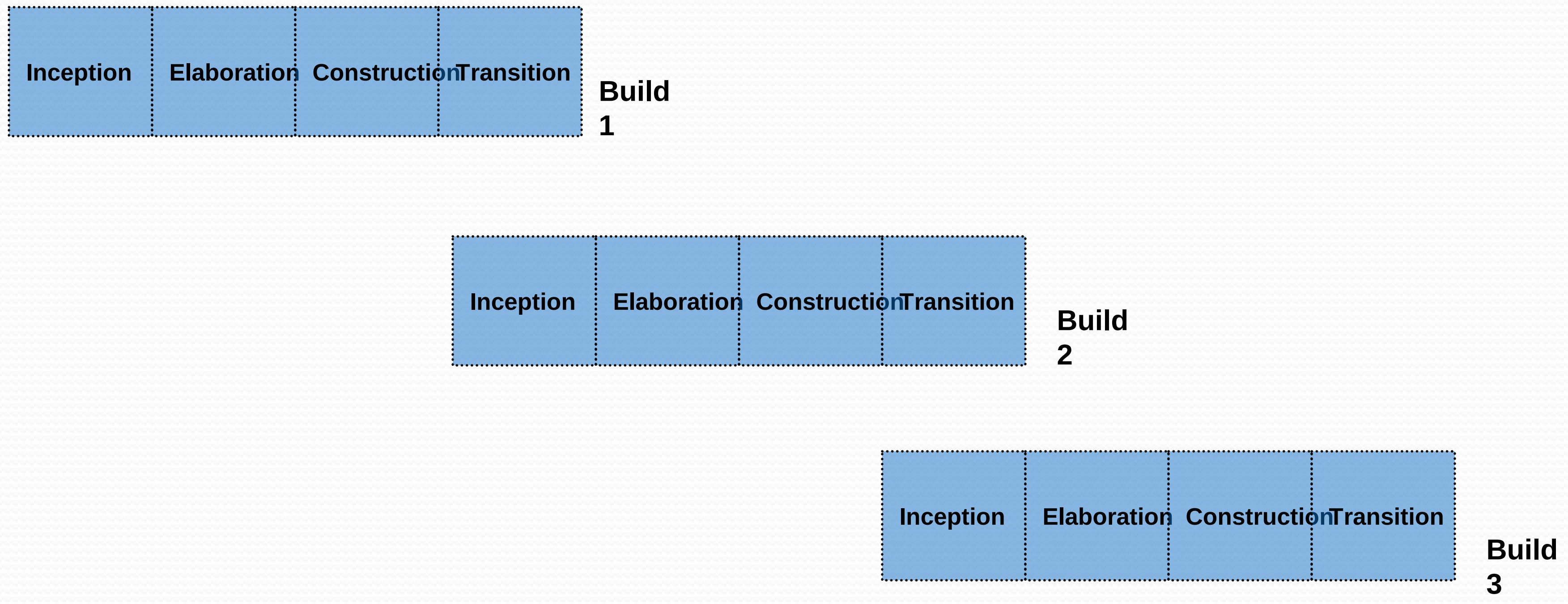
Larman 2.10

Four major activities

- Scoping
- Designing
- Building
- Verifying
- Iteration - Doing all over and over again in each of the four phases ***(Inception, Elaboration, Construction, Transition)***

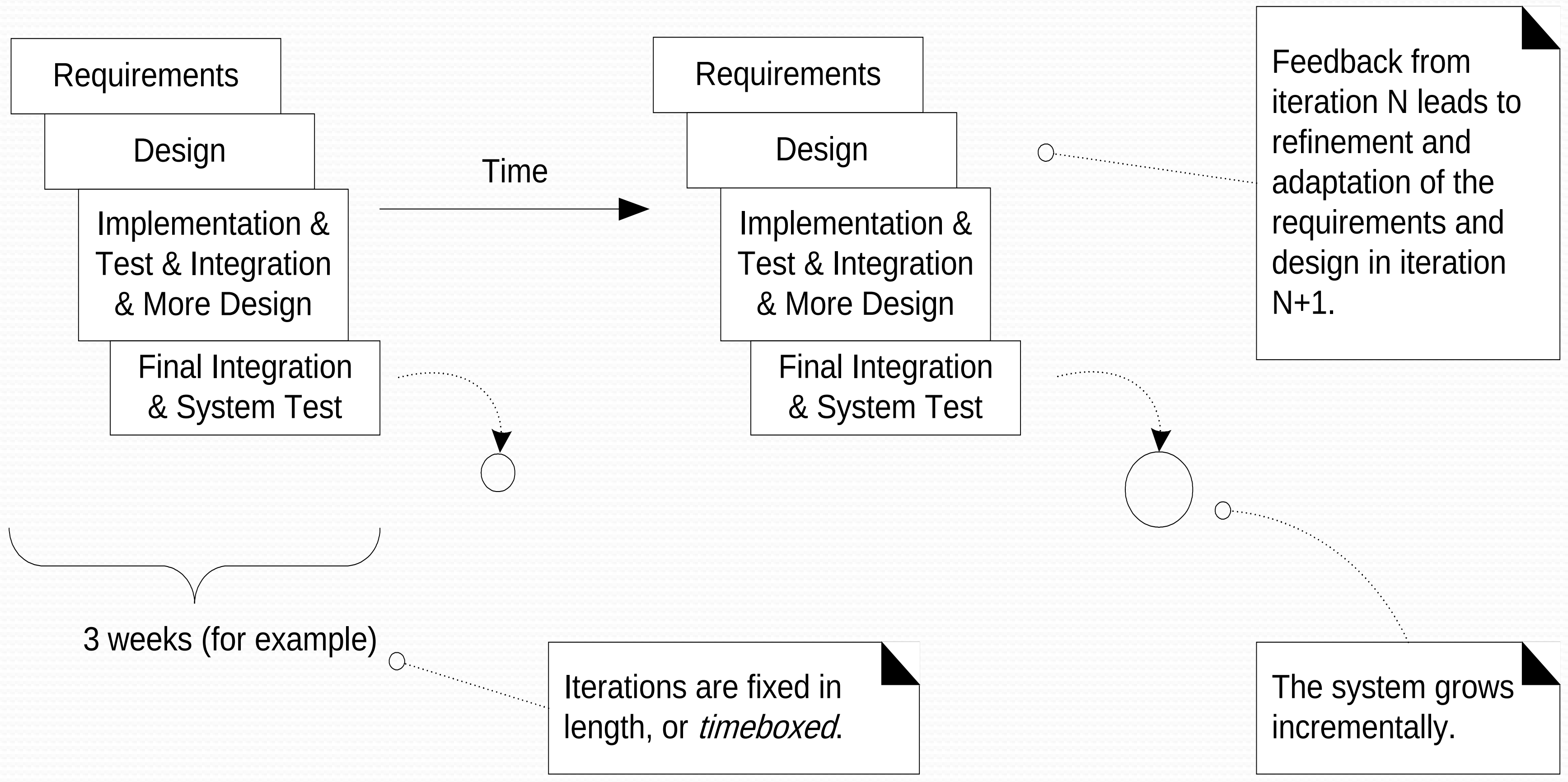
Unified Process (UP) - Iterative





Incremental - Iteration model

Iterative (Evolutionary) Development

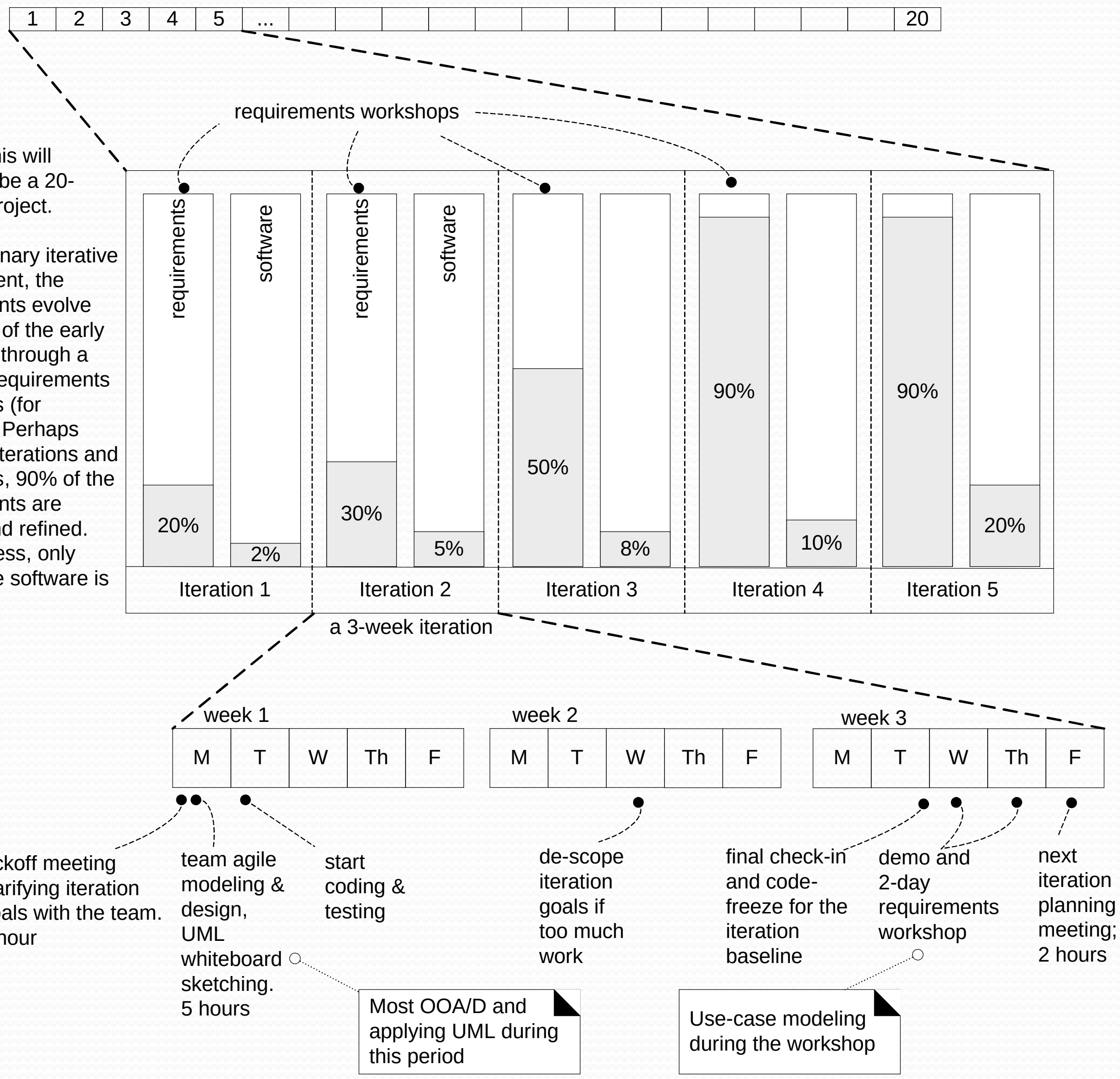

Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.
3 weeks (for example)
Iterations are fixed in length, or *timeboxed*.
The system grows incrementally.

Larman 2.2

Iterative (Evolutionary) Analysis And Design

Imagine this will ultimately be a 20-iteration project.

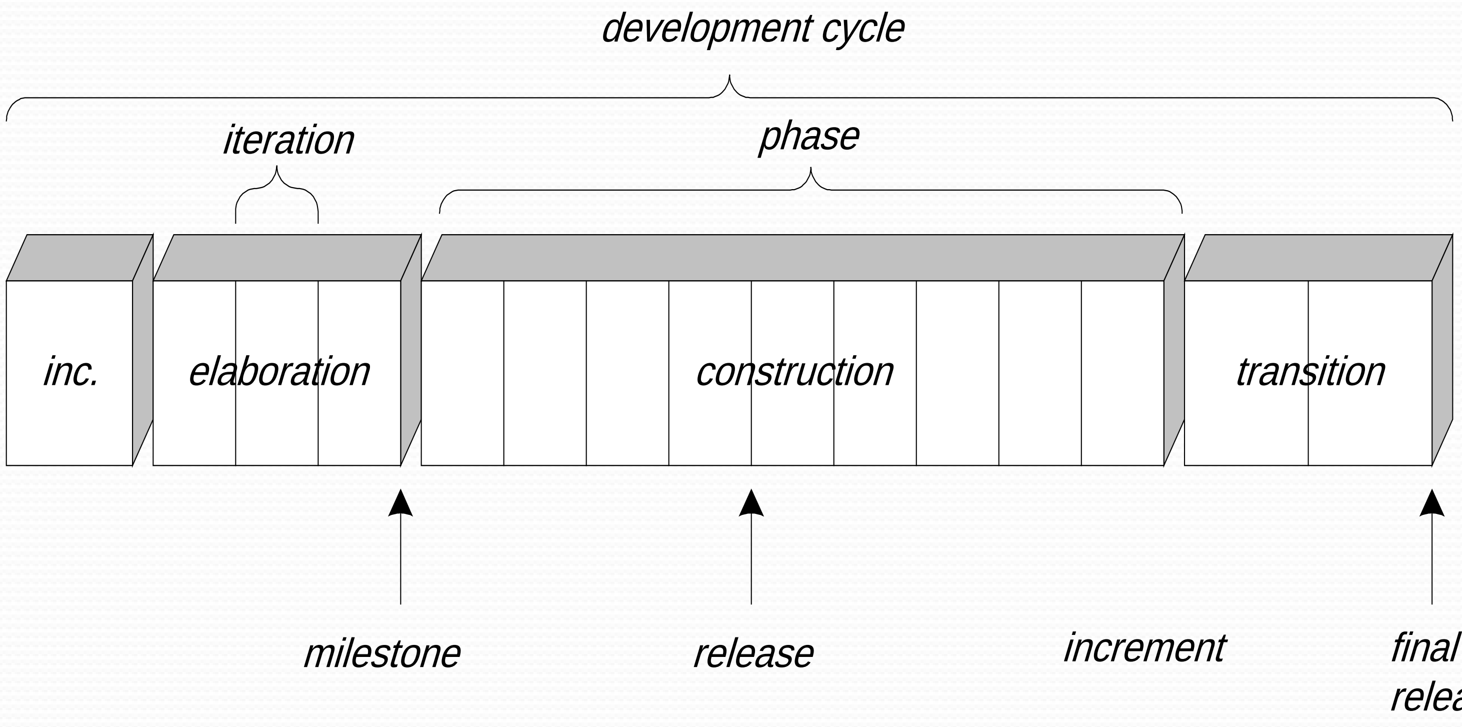
In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.



1 2 3 4 5 ... 20
requirements workshops
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M T W Th F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
next iteration planning meeting; 2 hours
Most OOA/D and applying UML during this period
Use-case modeling during the workshop

Larman Fig 2.4

UP Phases



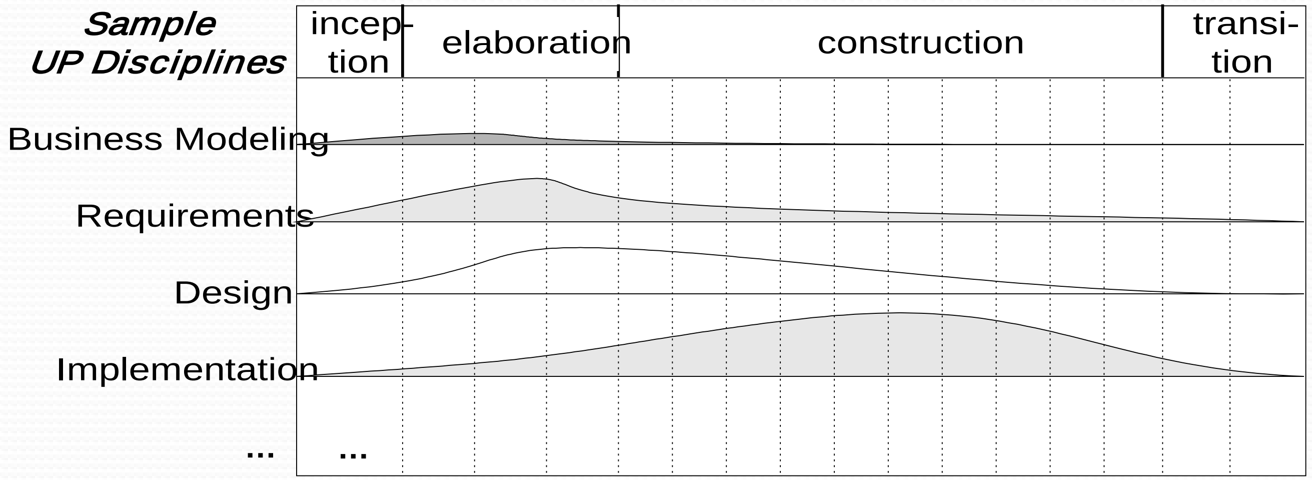
An iteration end-point when some significant decision or evaluation occurs.

A stable executable subset of the final product. The end of each iteration is a minor release.

The difference (delta) between the releases of 2 subsequent iterations.

At this point, the system is released for production use.

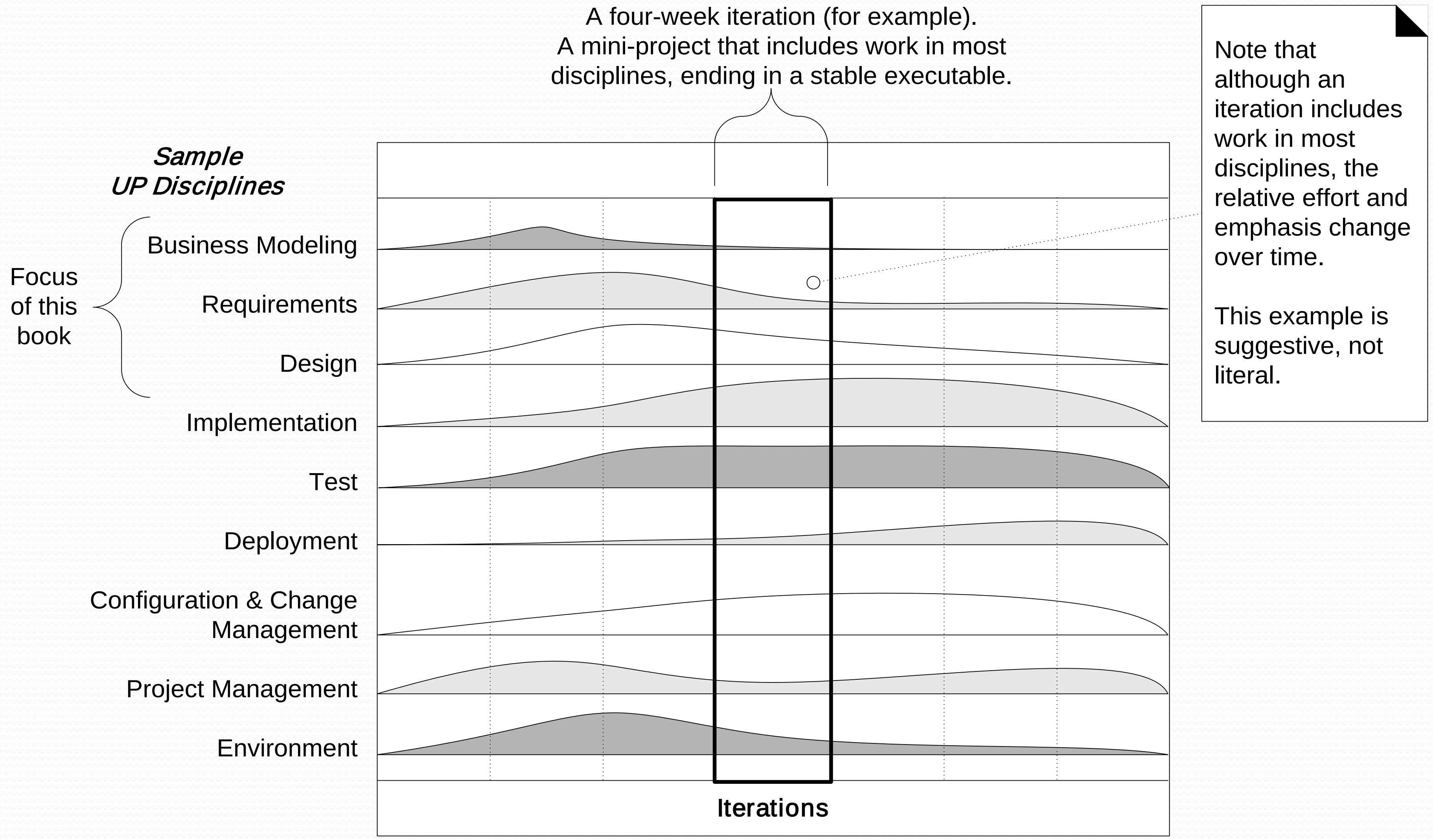
Larman Fig 2.6



The relative effort in
disciplines shifts
across the phases.

This example is
suggestive, not litera

Larman Fig 2.8


A four-week iteration (for example).
A mini-project that includes work in most disciplines, ending in a stable executable.
Note that although an iteration includes work in most disciplines, the relative effort and emphasis change over time.
This example is suggestive, not literal.
Sample UP Disciplines
Focus of this book
Business Modeling
Requirements
Design
Implementation
Test
Deployment
Configuration & Change Management
Project Management
Environment
Iterations

Larman Fig 2.7

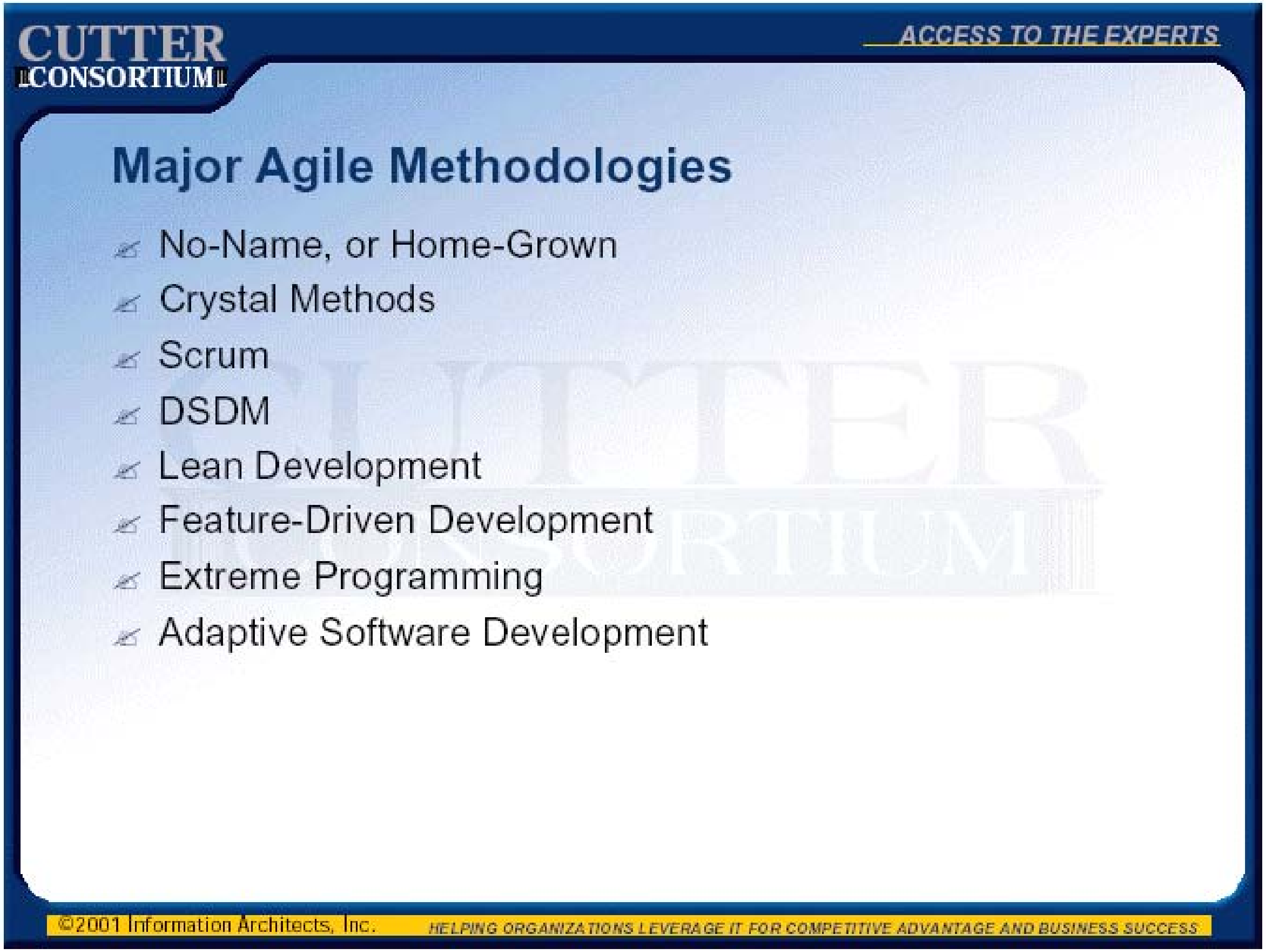
The Unified Process (UP)

- Reinforces six best practices
  - Develop iteratively
  - Define and manage system requirements
  - Use component architectures
  - Create visual models
  - Verify quality
  - Control changes

Variations of OO/UP Methodologies

- Extreme Programming (XP)
  - Ken Beck
  - Light weight
- Agile Modeling
  - Scott Ambler
  - Combine UP/XP
- Scrum
  - Takeuchi, Nonaka
  - Based on Agile
  - Adaptive Methodology

Manifesto Of Agile Alliance
•Uncover better ways
•Helping each other



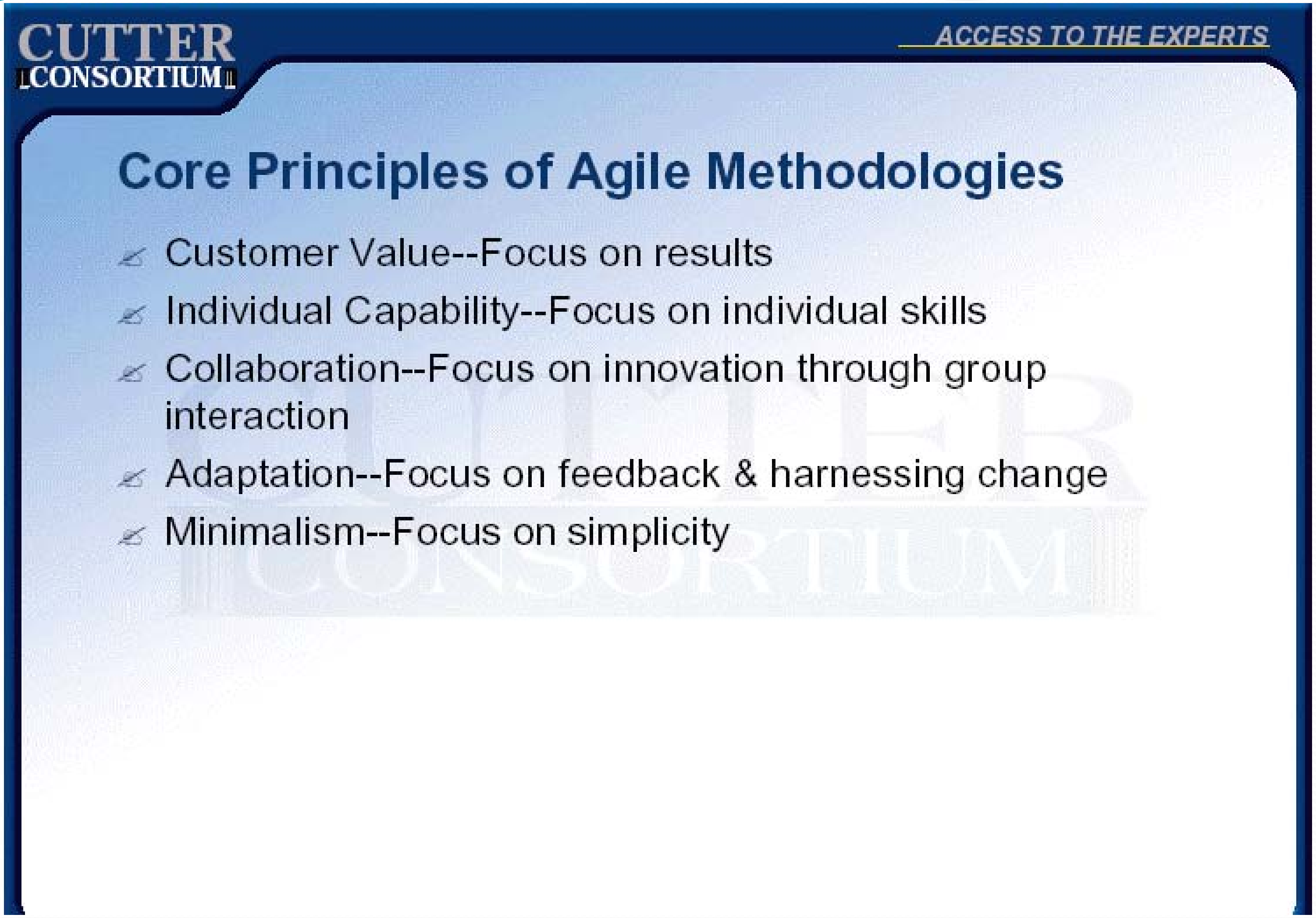
CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Major Agile Methodologies

- No-Name, or Home-Grown
- Crystal Methods
- Scrum
- DSDM
- Lean Development
- Feature-Driven Development
- Extreme Programming
- Adaptive Software Development

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS

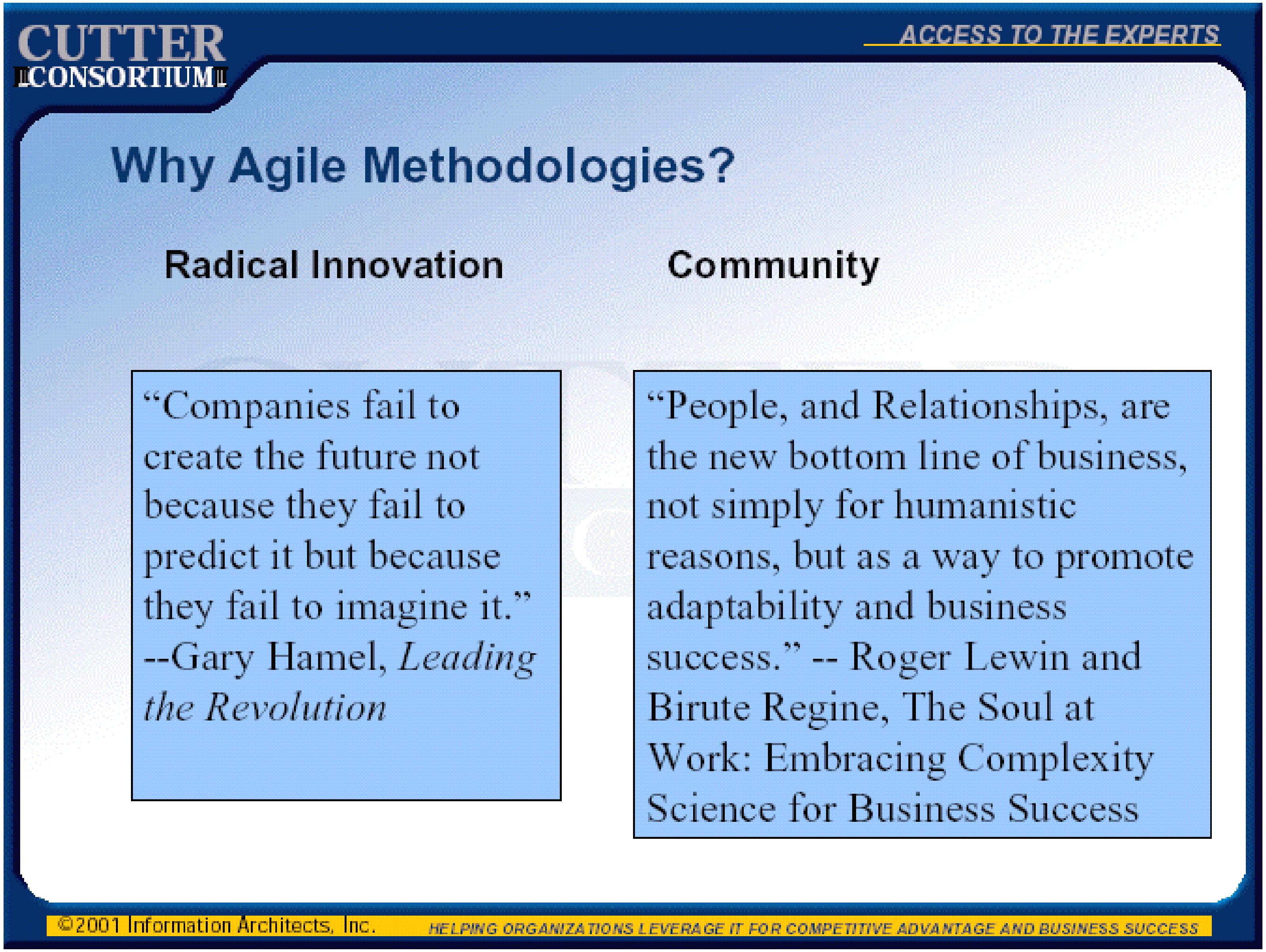


CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Core Principles of Agile Methodologies

- Customer Value--Focus on results
- Individual Capability--Focus on individual skills
- Collaboration--Focus on innovation through group interaction
- Adaptation--Focus on feedback & harnessing change
- Minimalism--Focus on simplicity



CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Why Agile Methodologies?

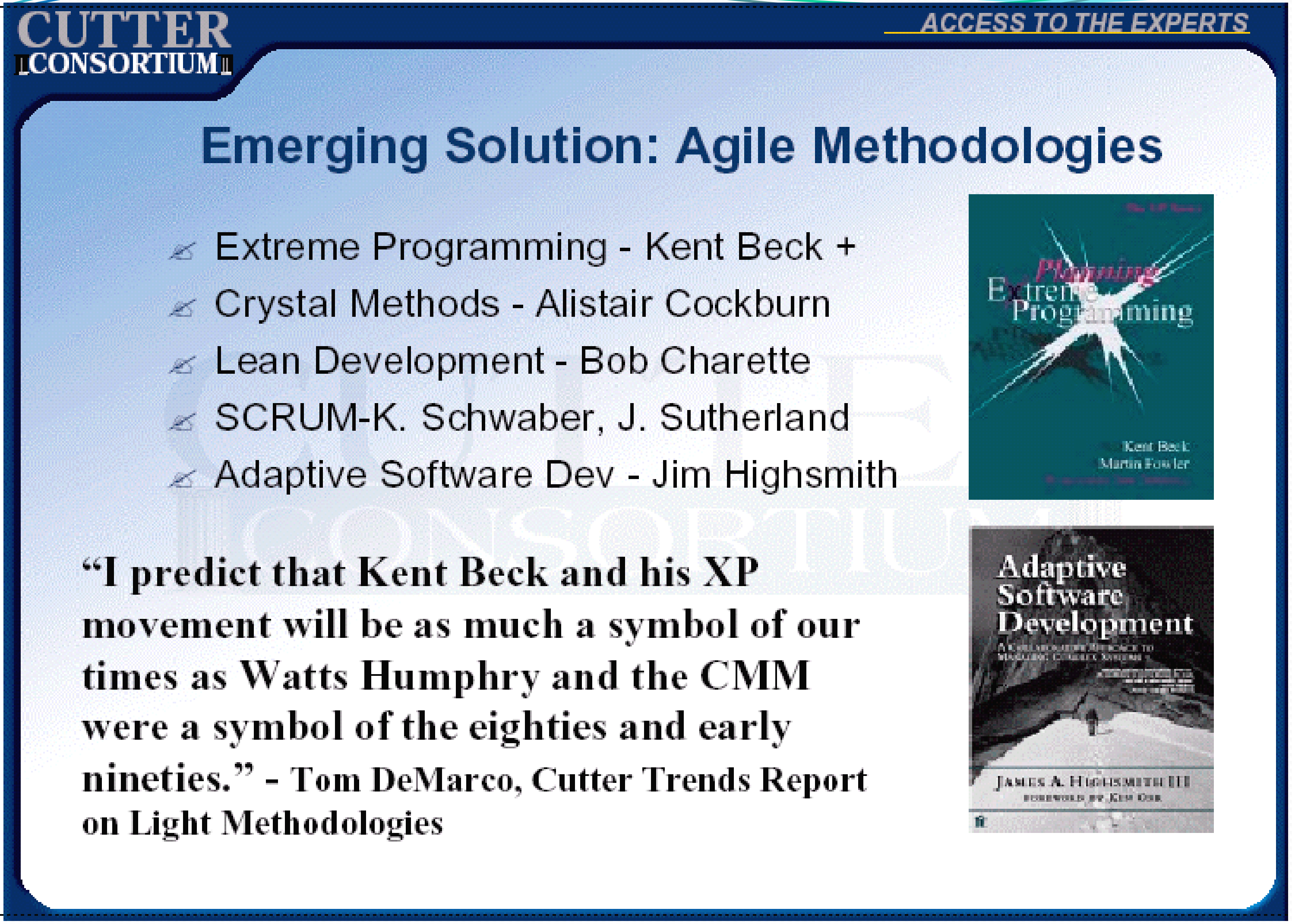
Radical Innovation

"Companies fail to create the future not because they fail to predict it but because they fail to imagine it." --Gary Hamel, *Leading the Revolution*

Community

"People, and Relationships, are the new bottom line of business, not simply for humanistic reasons, but as a way to promote adaptability and business success." -- Roger Lewin and Birute Regine, The Soul at Work: Embracing Complexity Science for Business Success

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



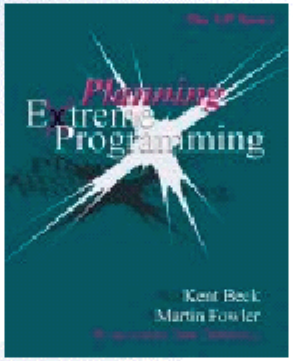
CUTTER CONSORTIUM

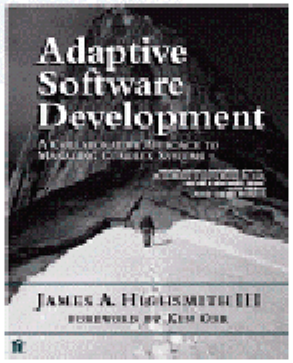
ACCESS TO THE EXPERTS

Emerging Solution: Agile Methodologies

- Extreme Programming - Kent Beck +
- Crystal Methods - Alistair Cockburn
- Lean Development - Bob Charette
- SCRUM-K. Schwaber, J. Sutherland
- Adaptive Software Dev - Jim Highsmith

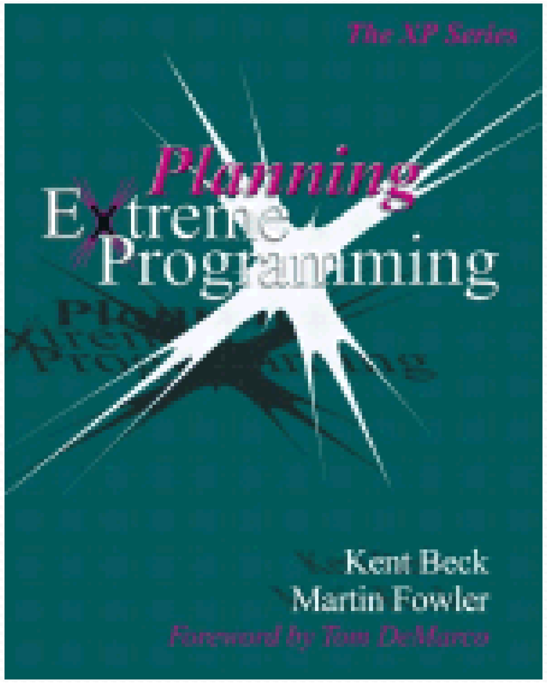
"I predict that Kent Beck and his XP movement will be as much a symbol of our times as Watts Humphry and the CMM were a symbol of the eighties and early nineties." - Tom DeMarco, Cutter Trends Report on Light Methodologies



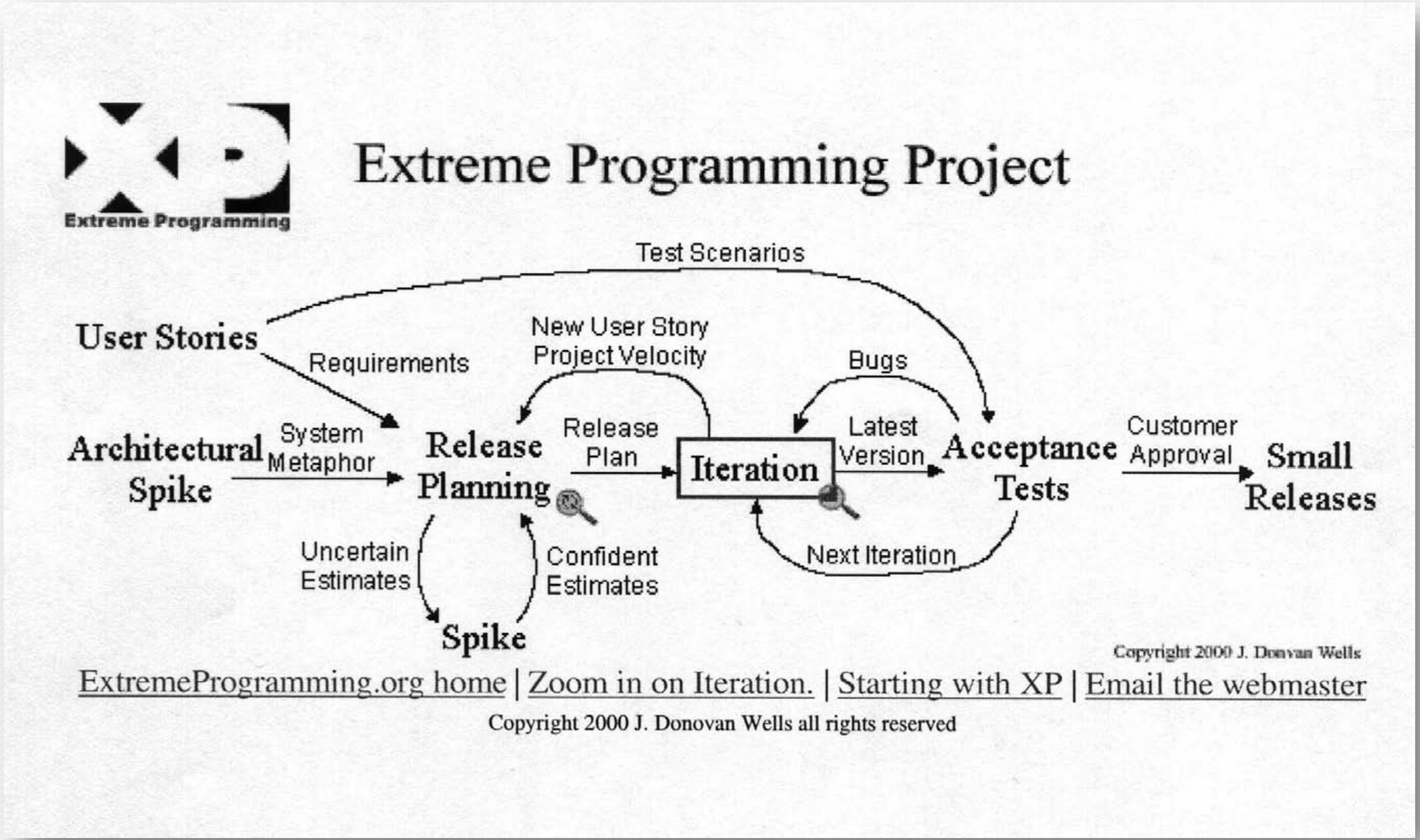


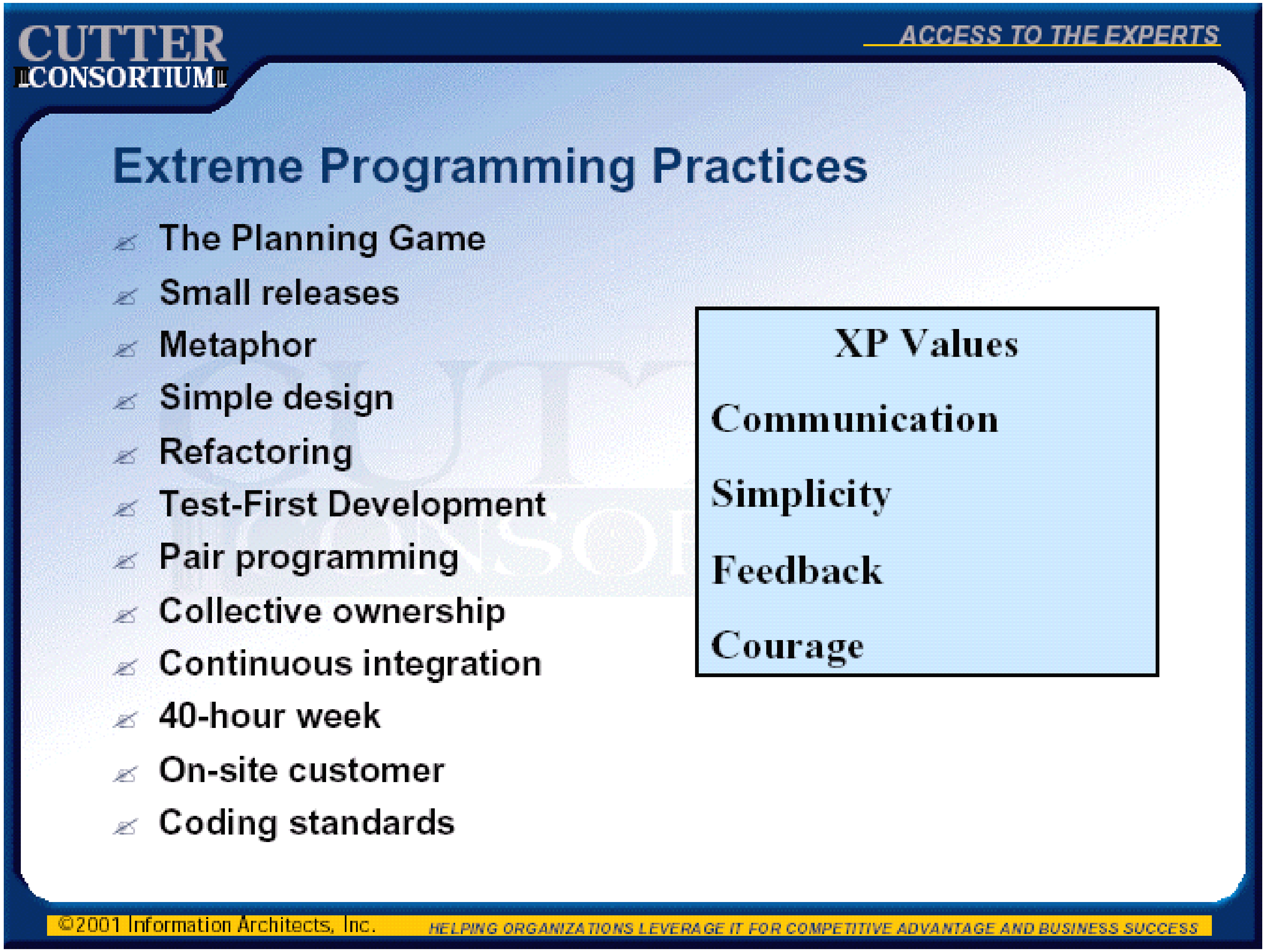
What is Extreme Programming

- By Ken Beck, Ward Cunningham, Ron Jeffries
- It is a ...
  - Discipline
  - Software Development Methodology
  - Lightweight
  - Evolutionary Design
  - Humanistic
  - Incremental Planning
  - Flexible scheduling
  - Automated testing



Example Of XP





CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Extreme Programming Practices

- The Planning Game
- Small releases
- Metaphor
- Simple design
- Refactoring
- Test-First Development
- Pair programming
- Collective ownership
- Continuous integration
- 40-hour week
- On-site customer
- Coding standards

XP Values

Communication

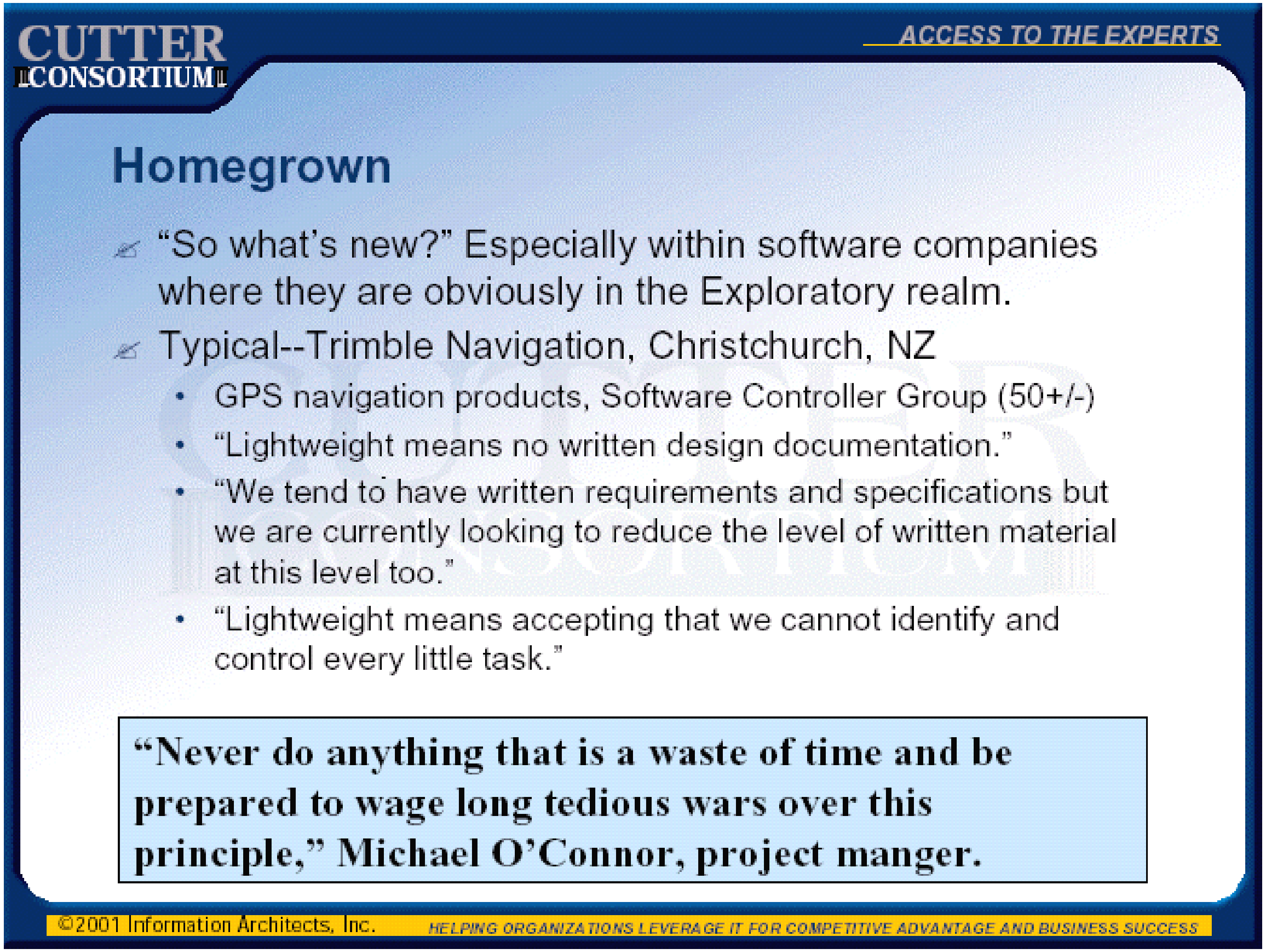
Simplicity

Feedback

Courage

©2001 Information Architects, Inc.

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



CUTTER CONSORTIUM

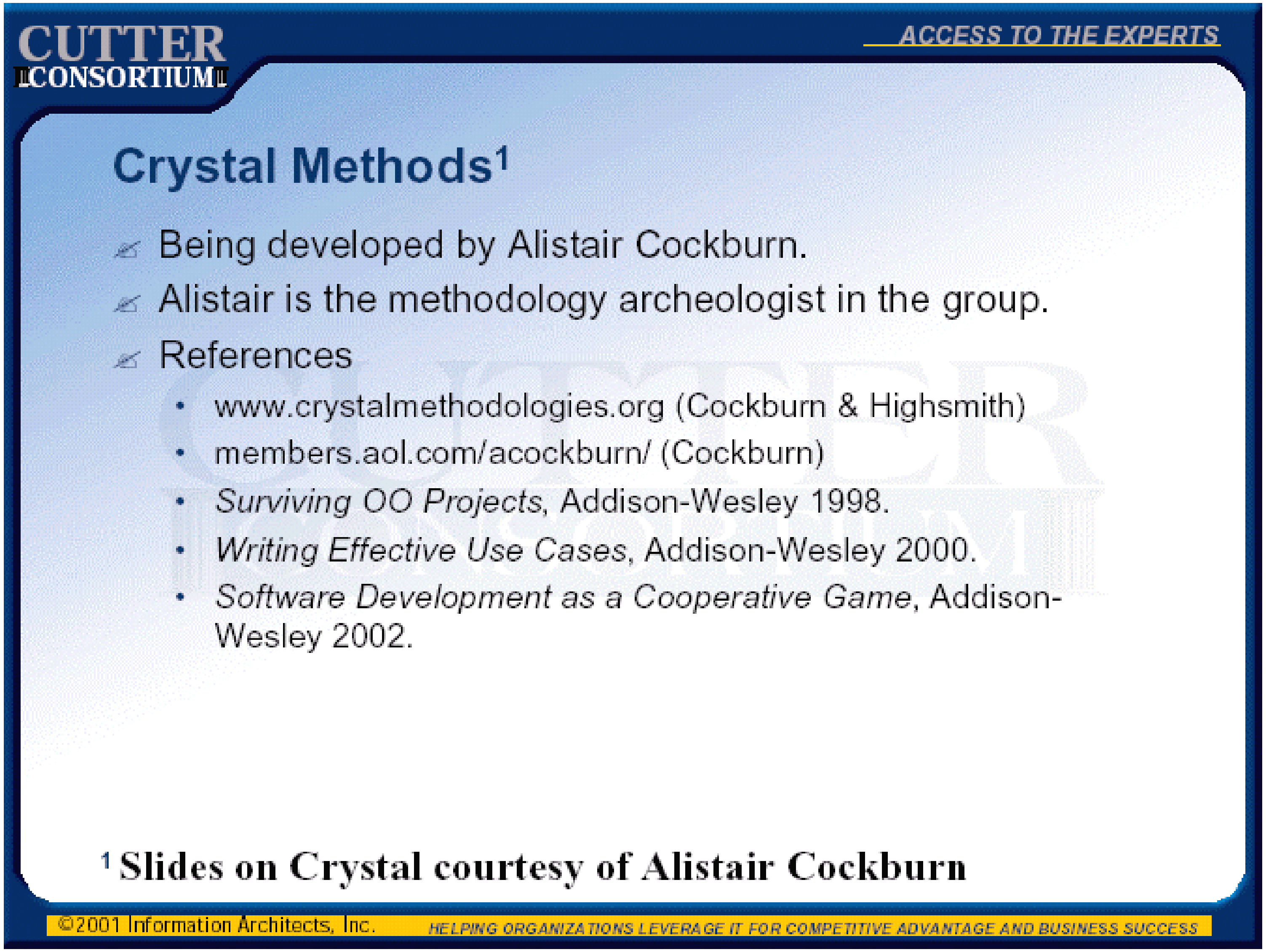
ACCESS TO THE EXPERTS

Homegrown

- "So what's new?" Especially within software companies where they are obviously in the Exploratory realm.
- Typical--Trimble Navigation, Christchurch, NZ
  - GPS navigation products, Software Controller Group (50+/-)
  - "Lightweight means no written design documentation."
  - "We tend to have written requirements and specifications but we are currently looking to reduce the level of written material at this level too."
  - "Lightweight means accepting that we cannot identify and control every little task."

"Never do anything that is a waste of time and be prepared to wage long tedious wars over this principle," Michael O'Connor, project manger.

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS


CUTTER CONSORTIUM

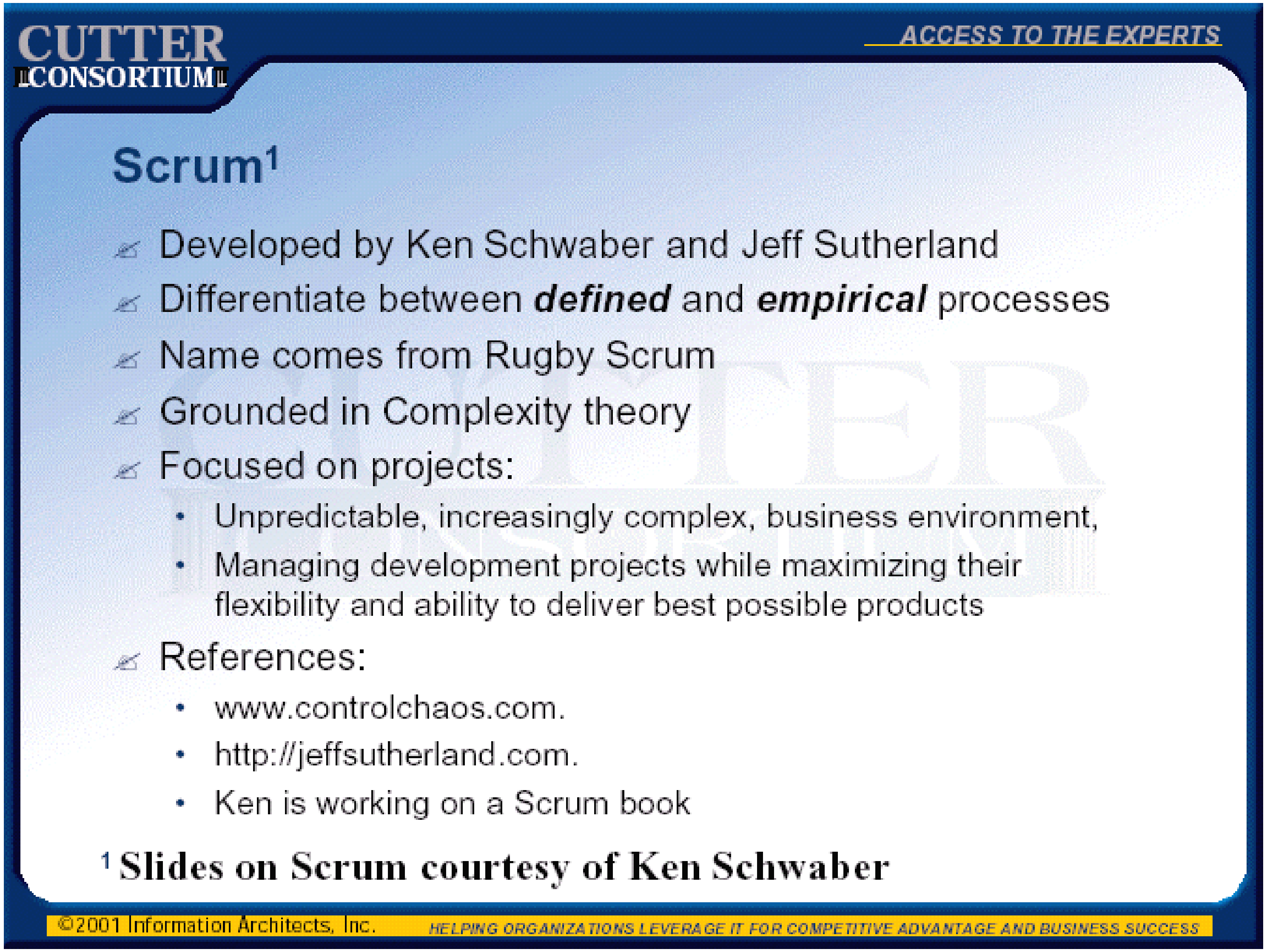
ACCESS TO THE EXPERTS

Crystal Methods[1]

- Being developed by Alistair Cockburn.
- Alistair is the methodology archeologist in the group.
- References
  - www.crystalmethodologies.org (Cockburn & Highsmith)
  - members.aol.com/acockburn/ (Cockburn)
  - *Surviving OO Projects*, Addison-Wesley 1998.
  - *Writing Effective Use Cases*, Addison-Wesley 2000.
  - *Software Development as a Cooperative Game*, Addison-Wesley 2002.

[1] Slides on Crystal courtesy of Alistair Cockburn

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



CUTTER CONSORTIUM

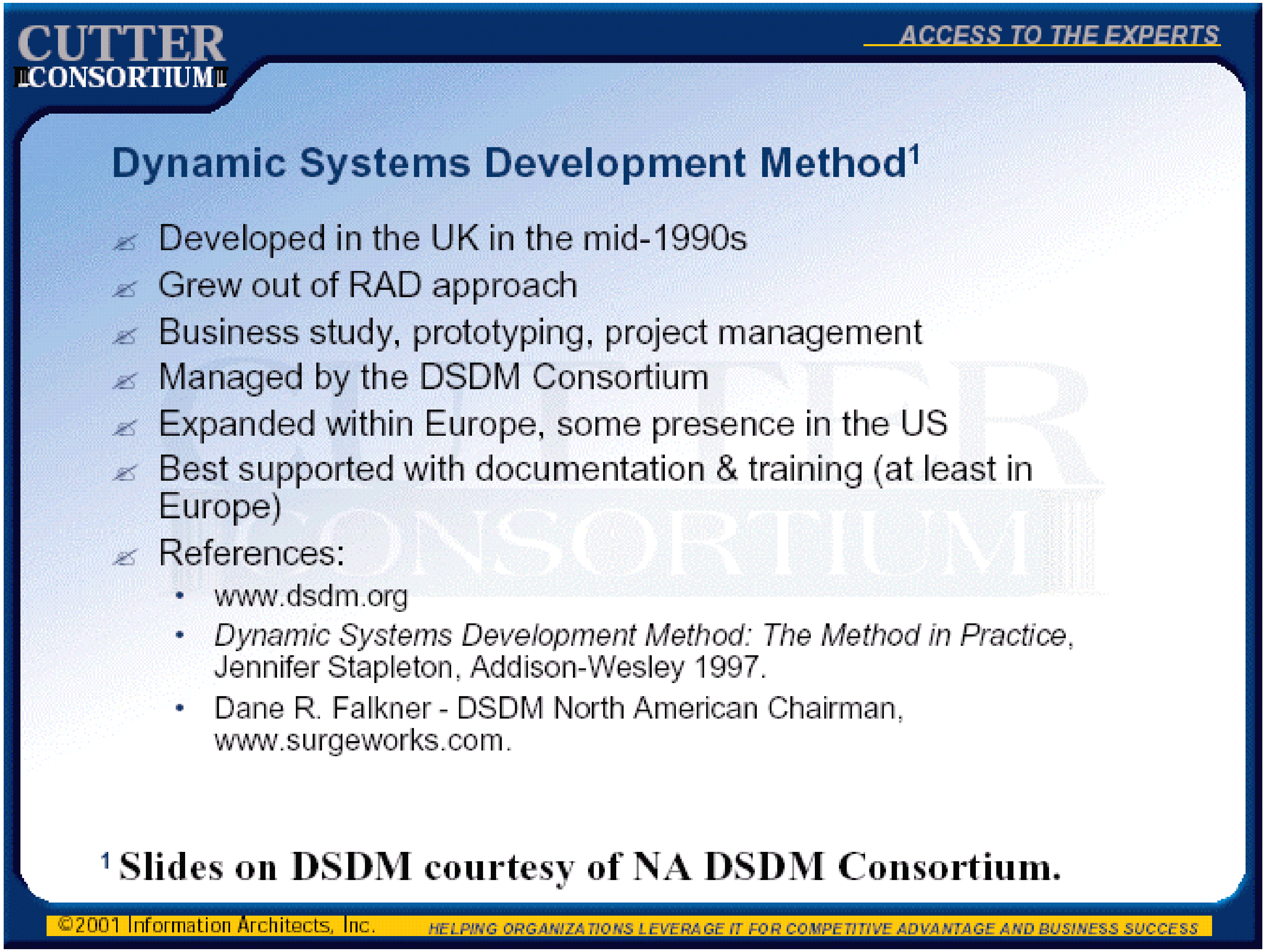
ACCESS TO THE EXPERTS

Scrum[1]

- Developed by Ken Schwaber and Jeff Sutherland
- Differentiate between ***defined*** and ***empirical*** processes
- Name comes from Rugby Scrum
- Grounded in Complexity theory
- Focused on projects:
  - Unpredictable, increasingly complex, business environment,
  - Managing development projects while maximizing their flexibility and ability to deliver best possible products
- References:
  - www.controlchaos.com.
  - http://jeffsutherland.com.
  - Ken is working on a Scrum book

[1] **Slides on Scrum courtesy of Ken Schwaber**

©2001 Information Architects, Inc. HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



CUTTER CONSORTIUM

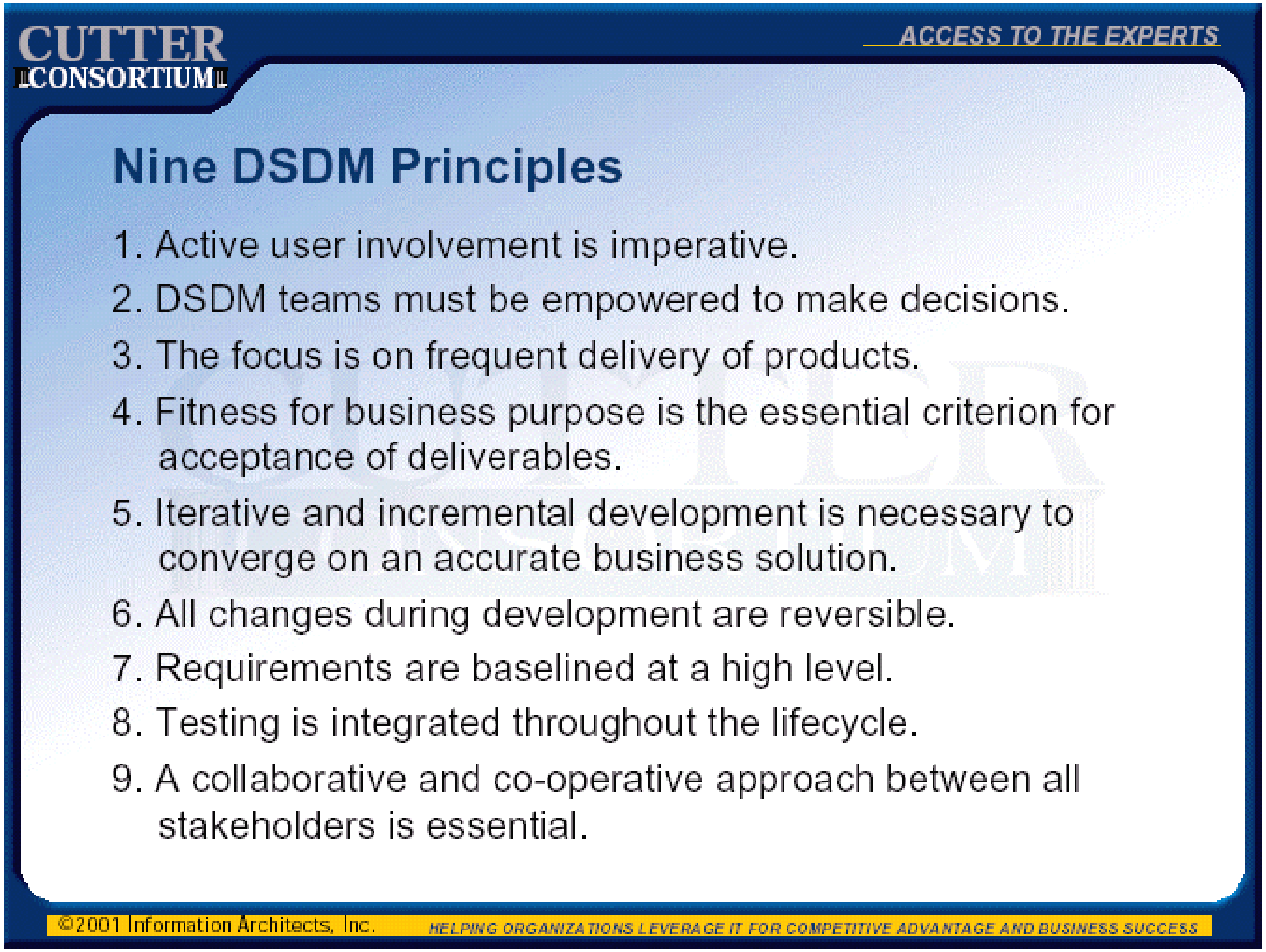
ACCESS TO THE EXPERTS

Dynamic Systems Development Method[1]

- Developed in the UK in the mid-1990s
- Grew out of RAD approach
- Business study, prototyping, project management
- Managed by the DSDM Consortium
- Expanded within Europe, some presence in the US
- Best supported with documentation & training (at least in Europe)
- References:
  - www.dsdm.org
  - *Dynamic Systems Development Method: The Method in Practice*, Jennifer Stapleton, Addison-Wesley 1997.
  - Dane R. Falkner - DSDM North American Chairman, www.surgeworks.com.

[1] **Slides on DSDM courtesy of NA DSDM Consortium.**

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



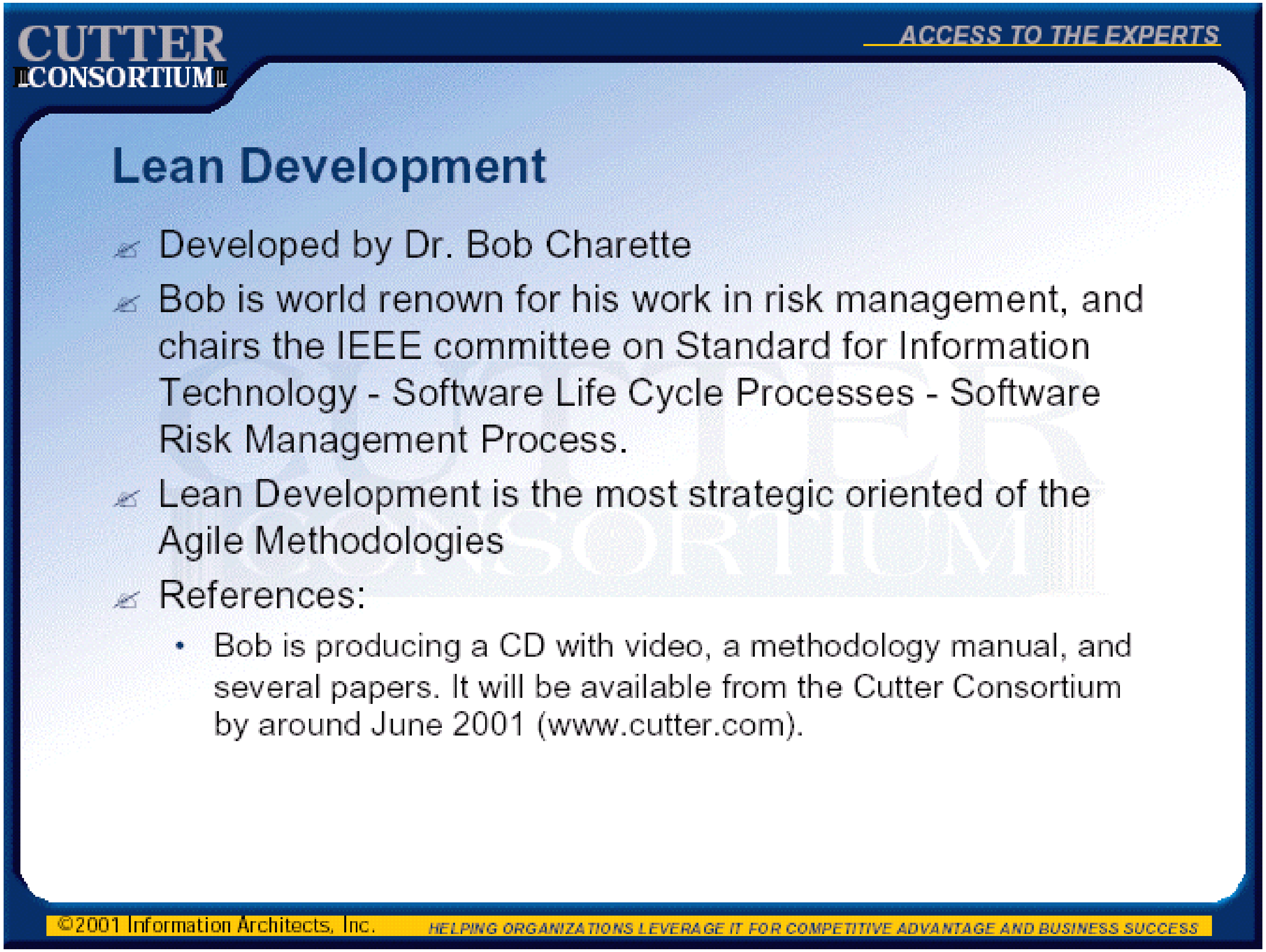
CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Nine DSDM Principles

1. Active user involvement is imperative.
2. DSDM teams must be empowered to make decisions.
3. The focus is on frequent delivery of products.
4. Fitness for business purpose is the essential criterion for acceptance of deliverables.
5. Iterative and incremental development is necessary to converge on an accurate business solution.
6. All changes during development are reversible.
7. Requirements are baselined at a high level.
8. Testing is integrated throughout the lifecycle.
9. A collaborative and co-operative approach between all stakeholders is essential.

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



CUTTER CONSORTIUM

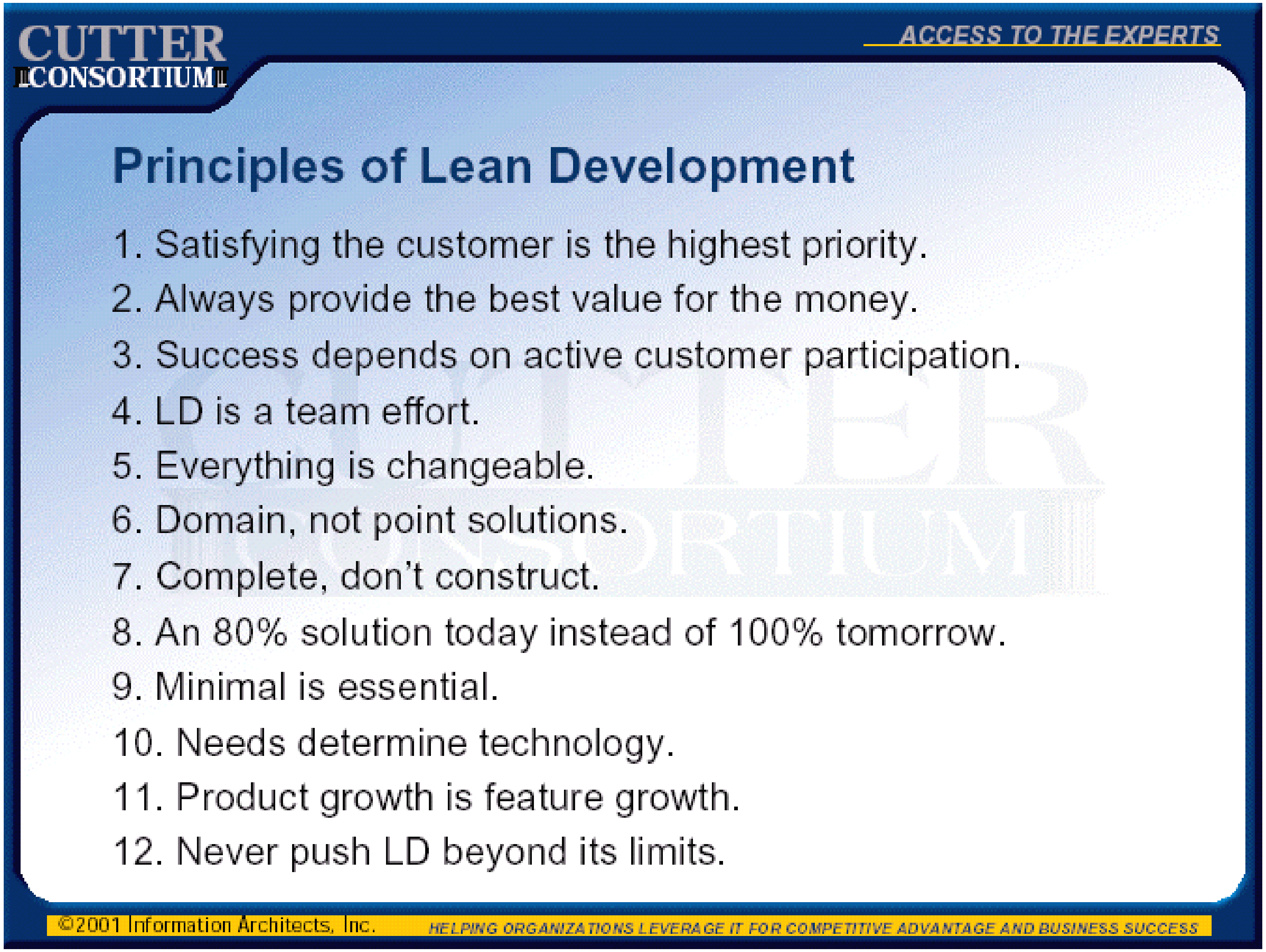
ACCESS TO THE EXPERTS

Lean Development

- Developed by Dr. Bob Charette
- Bob is world renown for his work in risk management, and chairs the IEEE committee on Standard for Information Technology - Software Life Cycle Processes - Software Risk Management Process.
- Lean Development is the most strategic oriented of the Agile Methodologies
- References:
  - Bob is producing a CD with video, a methodology manual, and several papers. It will be available from the Cutter Consortium by around June 2001 (www.cutter.com).

©2001 Information Architects, Inc.

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



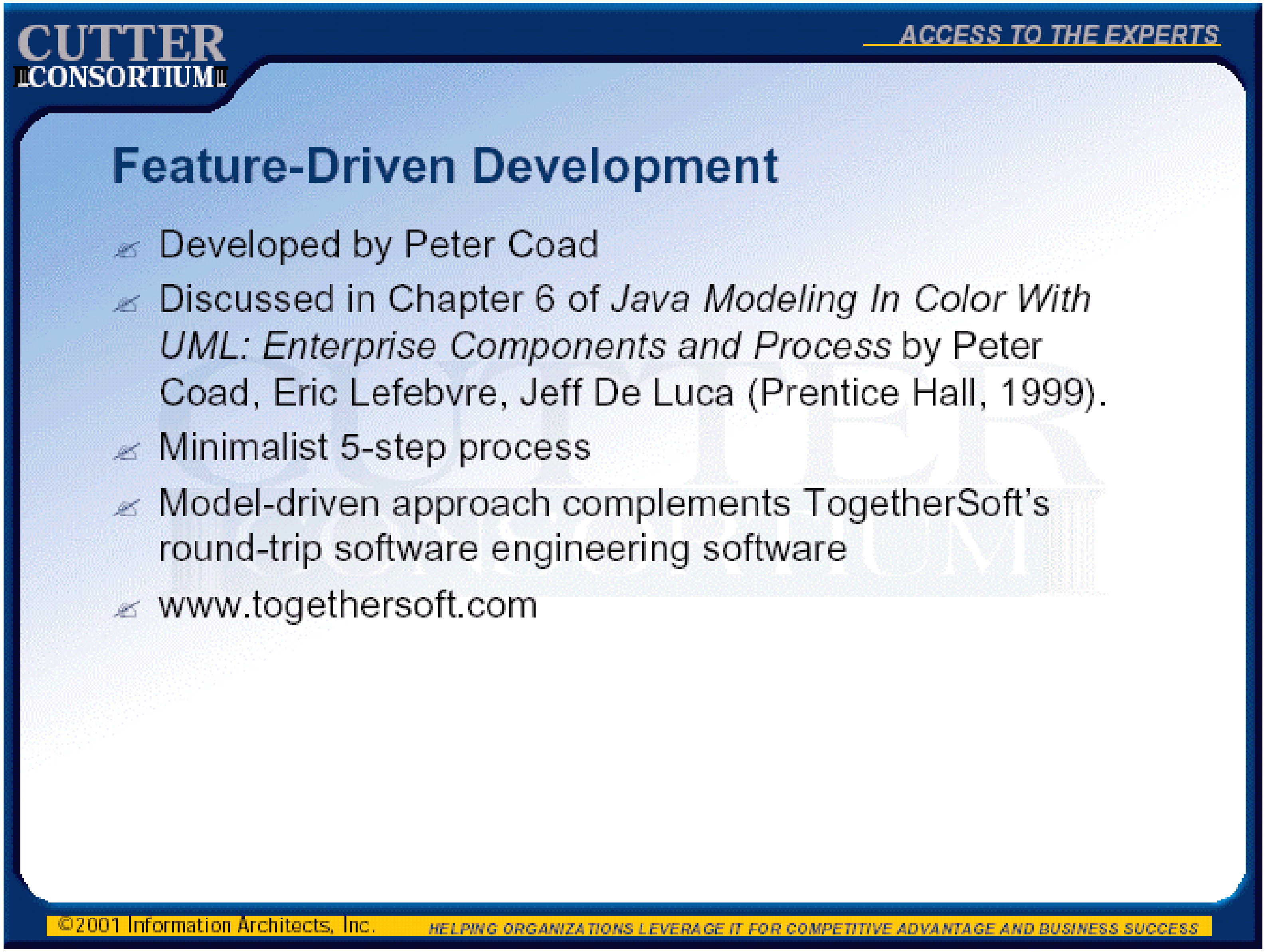
CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Principles of Lean Development

1. Satisfying the customer is the highest priority.
2. Always provide the best value for the money.
3. Success depends on active customer participation.
4. LD is a team effort.
5. Everything is changeable.
6. Domain, not point solutions.
7. Complete, don't construct.
8. An 80% solution today instead of 100% tomorrow.
9. Minimal is essential.
10. Needs determine technology.
11. Product growth is feature growth.
12. Never push LD beyond its limits.

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



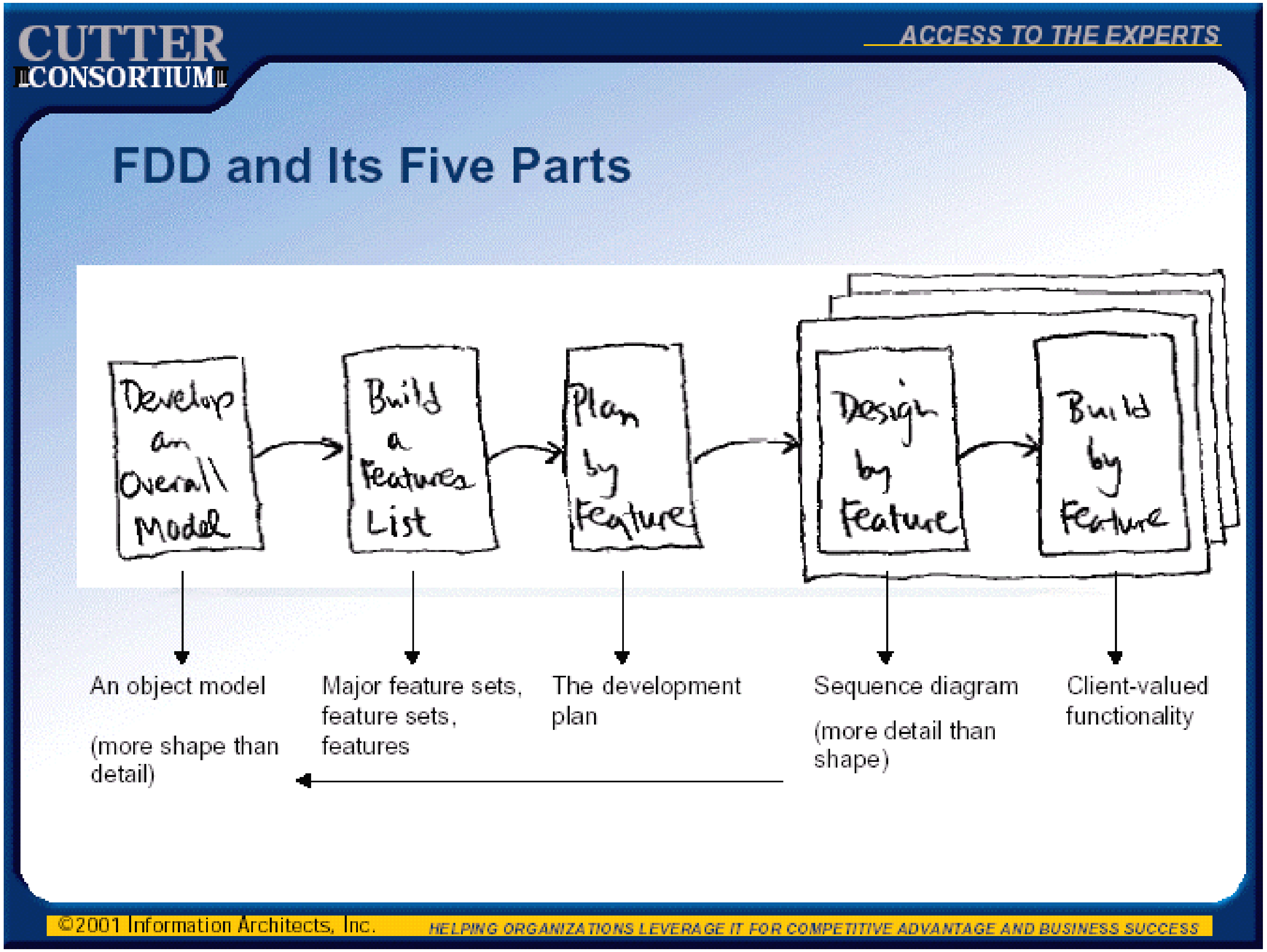
CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Feature-Driven Development

- Developed by Peter Coad
- Discussed in Chapter 6 of *Java Modeling In Color With UML: Enterprise Components and Process* by Peter Coad, Eric Lefebvre, Jeff De Luca (Prentice Hall, 1999).
- Minimalist 5-step process
- Model-driven approach complements TogetherSoft's round-trip software engineering software
- www.togethersoft.com

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS



CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

FDD and Its Five Parts

An object model

(more shape than detail)

Major feature sets, feature sets, features

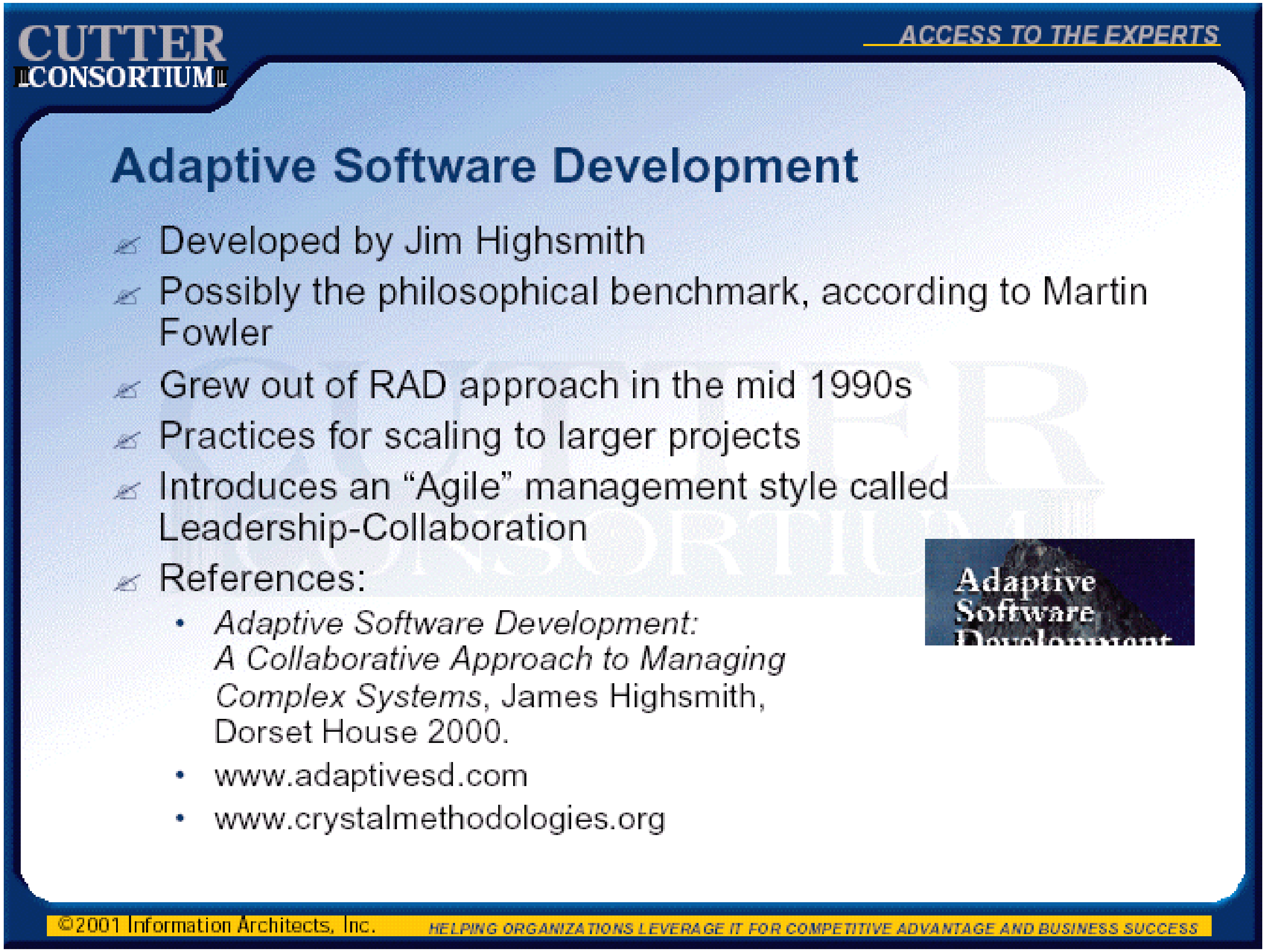
The development plan

Sequence diagram

(more detail than shape)

Client-valued functionality

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS


CUTTER CONSORTIUM

ACCESS TO THE EXPERTS

Adaptive Software Development

- Developed by Jim Highsmith
- Possibly the philosophical benchmark, according to Martin Fowler
- Grew out of RAD approach in the mid 1990s
- Practices for scaling to larger projects
- Introduces an "Agile" management style called Leadership-Collaboration
- References:
  - *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, James Highsmith, Dorset House 2000.
  - www.adaptivesd.com
  - www.crystalmethodologies.org

HELPING ORGANIZATIONS LEVERAGE IT FOR COMPETITIVE ADVANTAGE AND BUSINESS SUCCESS

Process vs Models

- Process (Methodologies)
    - Provides guidelines to follow
    - Include specific models, tools, techniques, documentation
- Model (Abstraction)
    - Representation of an important aspect of the “real world”
    - Use of drawings, diagrams, notations, symbols, conventions

Benefits Of Object Modeling

- Communication
- Visualization
- Test before Build
- Manage complexity
- Manage risk

Three Object Oriented Gurus

- Grady Booch
- James Rumbaugh
- Ivar Jacobson

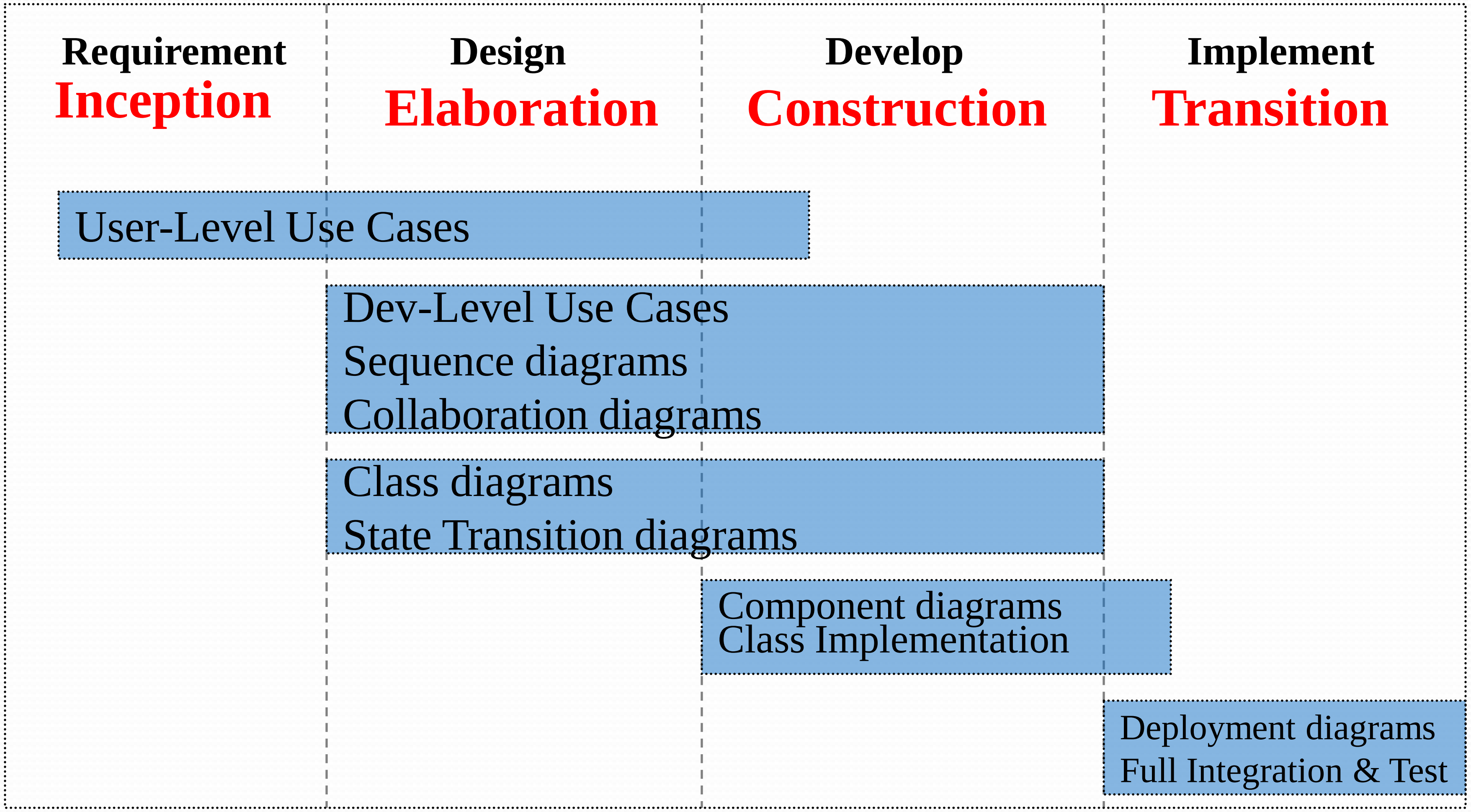
Proponents of UML (Unified Modeling Language) as standard OO modeling notation that can be used with any methodology.

What is UML?

- Quote :

  “The Unified Modeling Language is a visual language for specifying, constructing and documenting the artifacts of systems” [OMG03a]

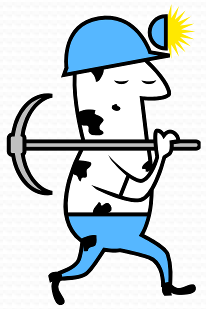
Using UML Models In UP



Iterative Development

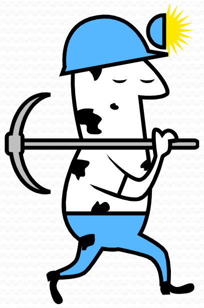
Tools Support Software Development

- Computer-Aided System Engineering (CASE)
  - Automated tools to improve the speed and quality of system development work
  - Contains repository of documentation
  - Upper CASE – support for analysis and design
  - Lower CASE – support for implementation
  - ICASE – integrated CASE tools



Tools Support Software Development

- Other names for CASE:
  - IDE - integrated development environment
  - integrated application development tools
  - visual modeling tools
  - round-trip engineering tools

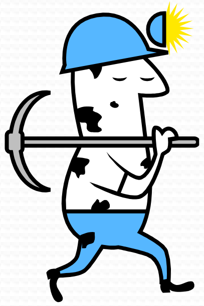


Larman 22.1

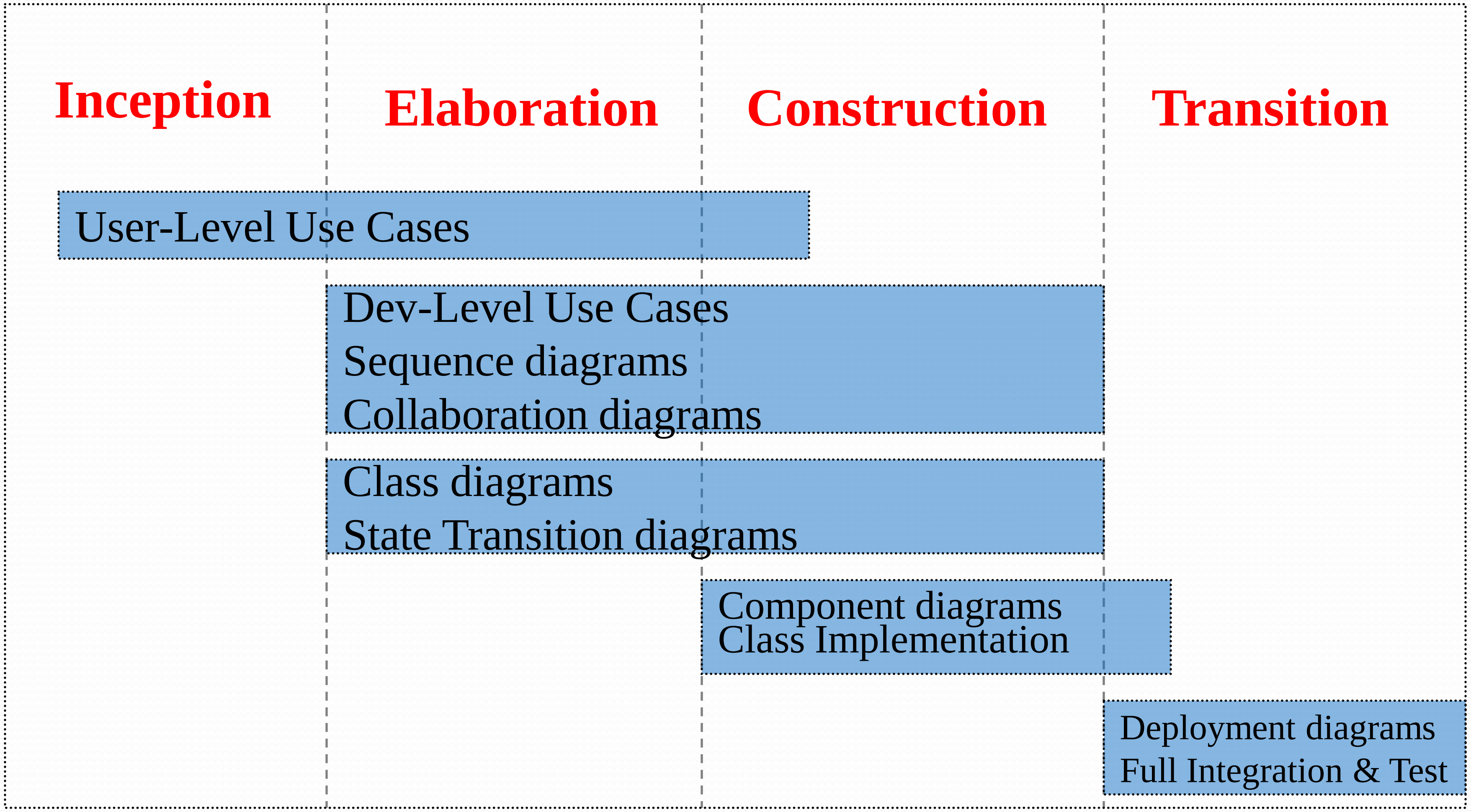
Tools – Some Examples

- IDE with round-trip engineering
  - Rational XDE (formerly called Rational Rose)
  - Embarcadero Describe
  - Netbeans

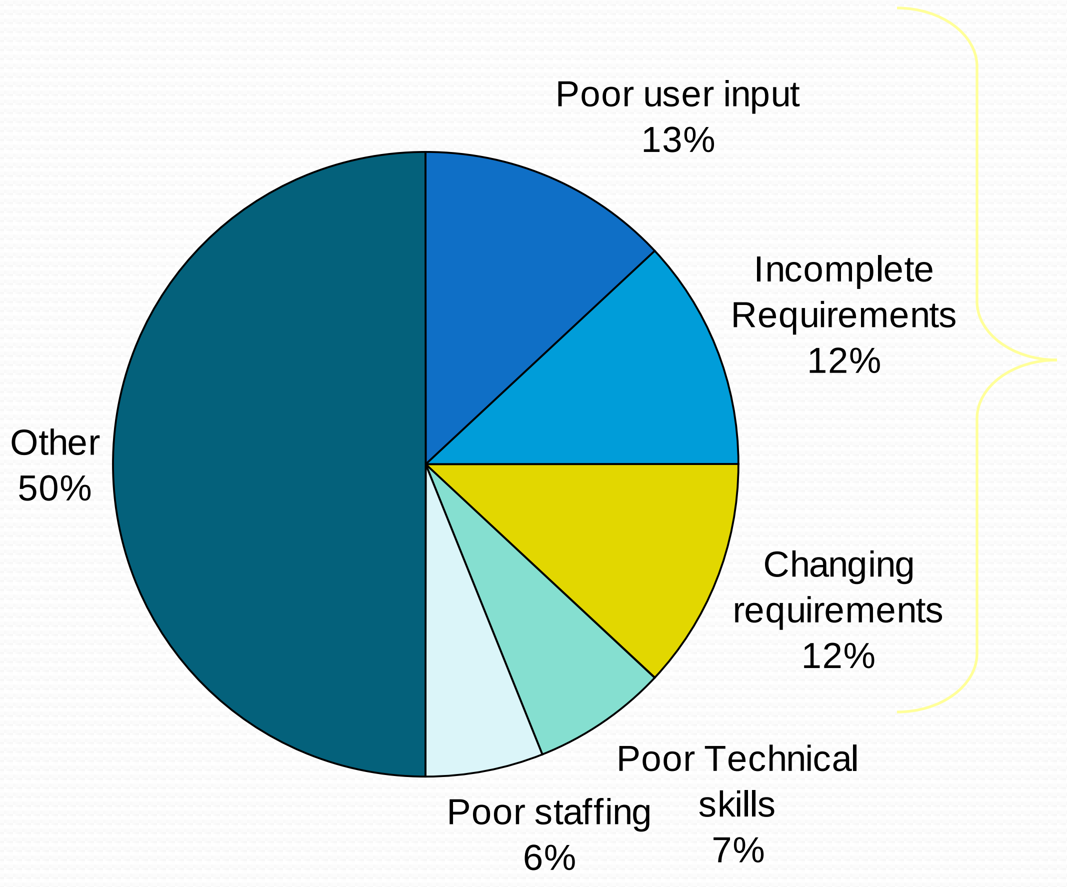
- Diagramming
  - Microsoft Visio
  - Visible Analyst (www.visible.com)



UML And UP


Inception
Elaboration
Construction
Transition
User-Level Use Cases
Dev-Level Use Cases
Sequence diagrams
Collaboration diagrams
Class diagrams
State Transition diagrams
Component diagrams
Class Implementation
Deployment diagrams
Full Integration & Test

Factors on challenged software projects



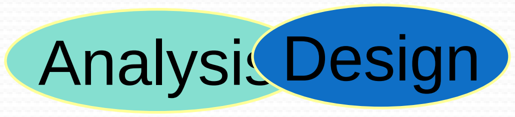
37% relate to problems with requirements

Purposes of Analysis and Design

- To evolve the requirements and transform them into a design of the system to-be
- To evolve a robust architecture for the system
- To adapt the design to match the implementation environment, designing it for performance

Analysis Versus Design

- Analysis
  - Focus on understanding the problem domain
  - Idealized design
  - Behavior
  - Functional requirements
  - System structure
- Design
  - Focus on understanding the solution
  - Operations and Attributes
  - Performance
  - Close to real code
  - Object lifecycles
  - Non-functional requirements
  - A large model



Larman 1.4

Where does COMP 3711 fit?

- Phases:
  - Analyses and Design
  - Some implementation mostly as related to Testing
- Phase Scheduling:
  - Iterative
- Approach:
  - Object Oriented
- Methodologies
  - Unified Process (UP) – Mostly Elaboration Phase
  - Agile
  - Iterative
- Modeling
  - UML
- These associated with OO Approach