Selecting an Agile Process:

Comparing the Leading Alternatives

Presented at SQuAD

October 15, 2002

By Mike Cohn

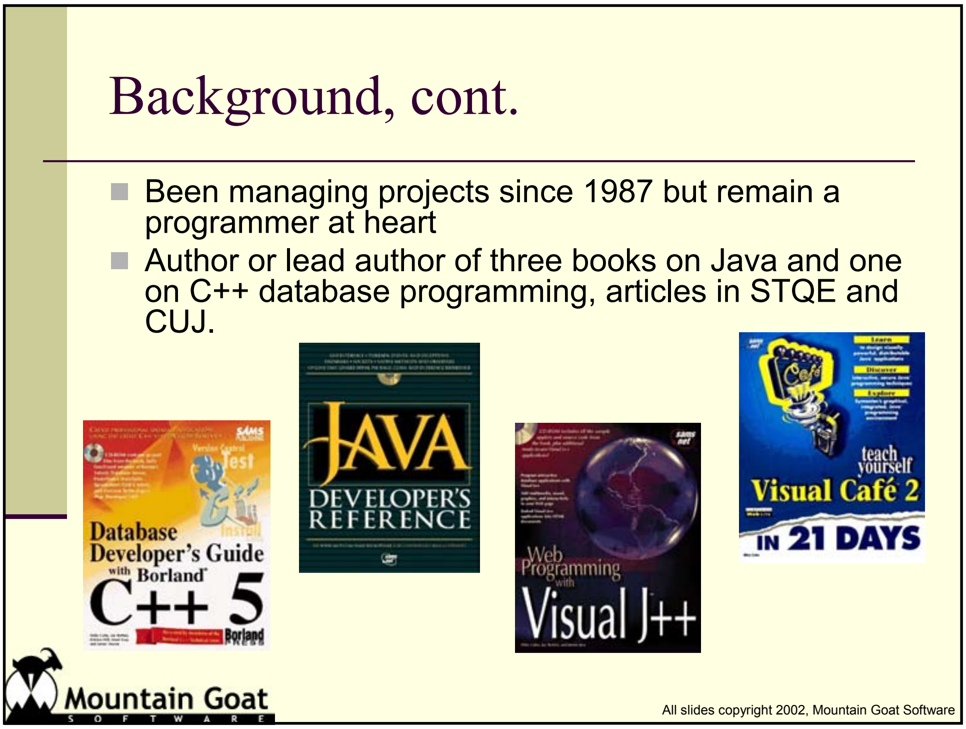
Mountain Goat SOFTWARE

Presenter background

- Spent much of the last 15 years consulting and running contract development projects:
  - Viacom, Procter & Gamble, NBC, United Nations, Citibank, other smaller companies
- Have periodically taken full-time positions:
  - Genomica, McKesson, Arthur Andersen
- Diverse background across:
  - Internal software vs. Shrinkwrap products
  - Web vs. Client-server
  - Java vs. Microsoft languages
- Master's degrees in CS and Economics

Agile Alliance founders

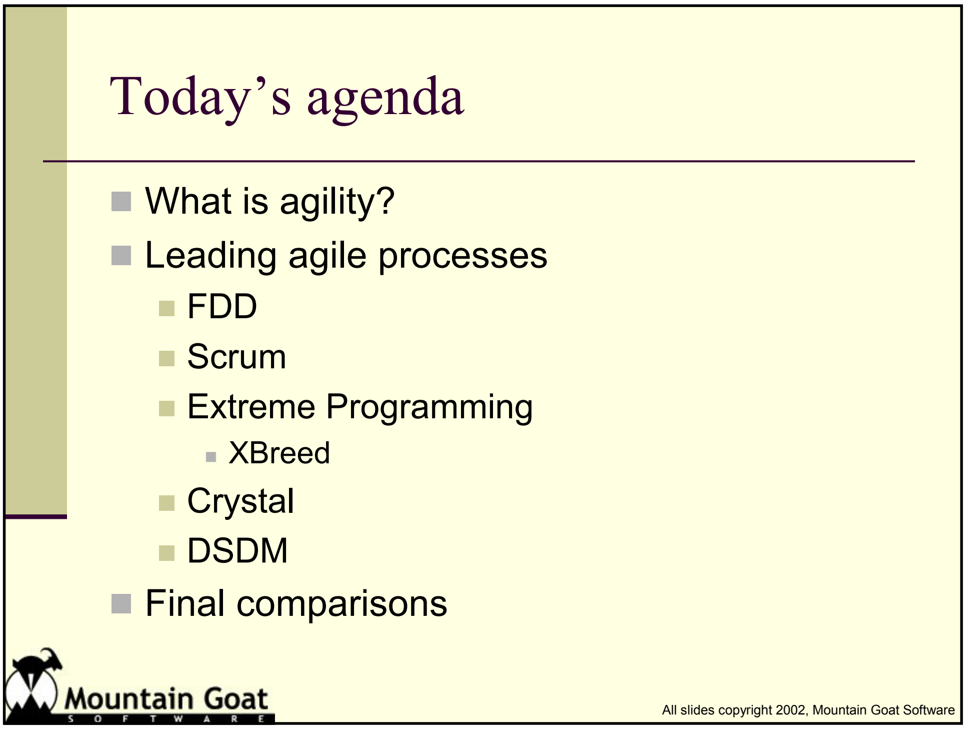
Mountain Goat SOFTWARE



Background, cont.

- Been managing projects since 1987 but remain a programmer at heart
- Author or lead author of three books on Java and one on C++ database programming, articles in STQE and CUJ.

Database Developer's Guide with Borland C++ 5
JAVA DEVELOPER'S REFERENCE
Web Programming with Visual J++
teach yourself Visual Café 2 IN 21 DAYS

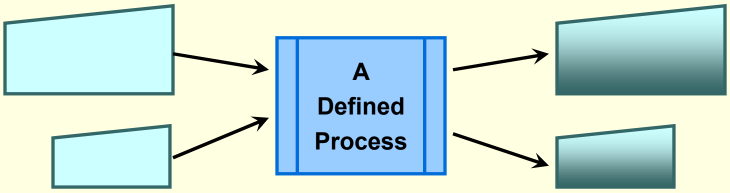


Today's agenda

- What is agility?
- Leading agile processes
  - FDD
  - Scrum
  - Extreme Programming
    - XBreed
  - Crystal
  - DSDM
- Final comparisons

What is agility?

A Defined Process



A Defined Process

- Every task must be completely understood.
- When given a well-defined set of inputs, the same outputs are generated every time.

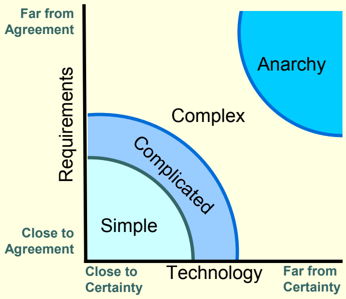
What is agility?

Software development: A defined process?

- Is every task completely understood?
  - Are we even getting closer?
- Given the exact same inputs (including people)
  - Will we get the same results every time?
  - Can we even have the exact same inputs?

What is agility?

Project Noise Level



Source: *Strategic Management and Organizational Dynamics* by Ralph Stacey in *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

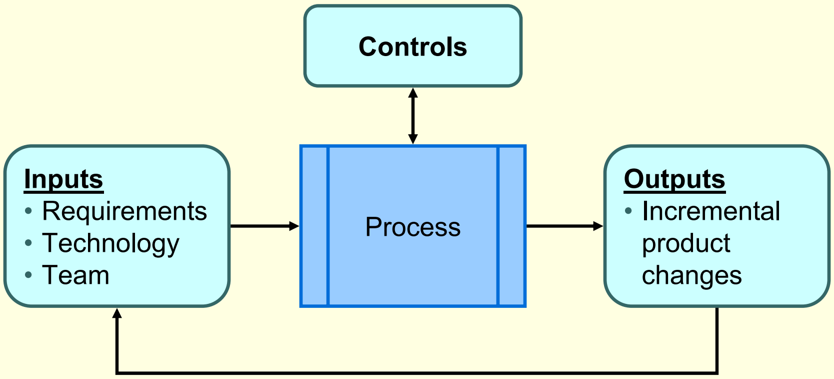
What is agility?

Empirical model of process control

- Useful when
  - Process cannot be sufficiently described to ensure repeatability
  - There is so much complexity or noise that the process leads to different outcomes
- Expects the unexpected
- Exercises control through frequent inspection and adaptation

What is agility?

Empirical model



Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

What is agility?

Defined vs. Empirical

"It is typical to adopt the defined (theoretical) modeling approach when the underlying mechanisms by which a process operates are reasonably well understood. When the process is too complicated for the defined approach, the empirical approach is the appropriate choice."

Process Dynamics, Modeling, and Control, Ogunnaike and Ray, Oxford University Press, 1992

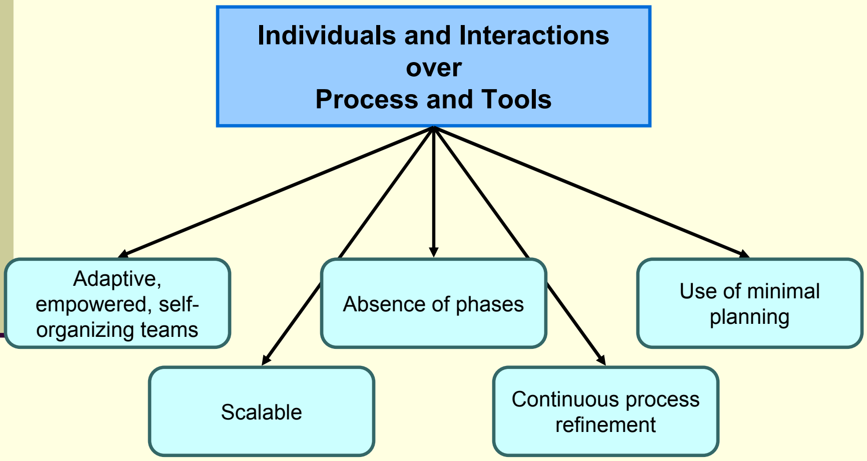
What is agility?

The Agile Manifesto

- 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

What is agility?

Individuals and interactions


Individuals and Interactions over Process and Tools

Adaptive, empowered, self-organizing teams

Absence of phases

Use of minimal planning

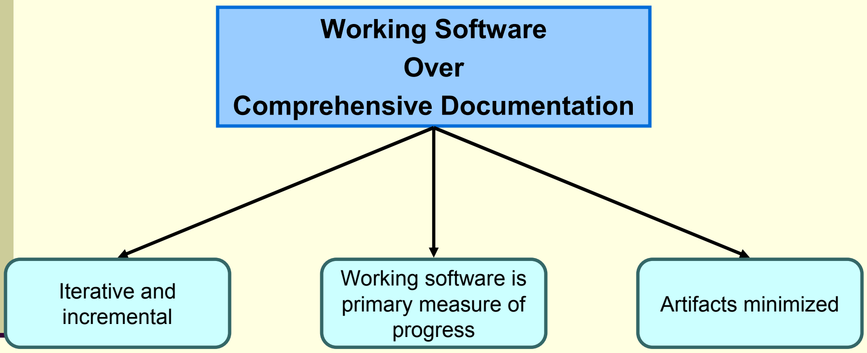
Scalable

Continuous process refinement

Adapted from: "Will the Real Agile Processes Please Stand Up", Ken Schwaber, *Cutter Consortium E-Project Management Advisory Service*, v. 2, no. 8.

Working software

What is agility?



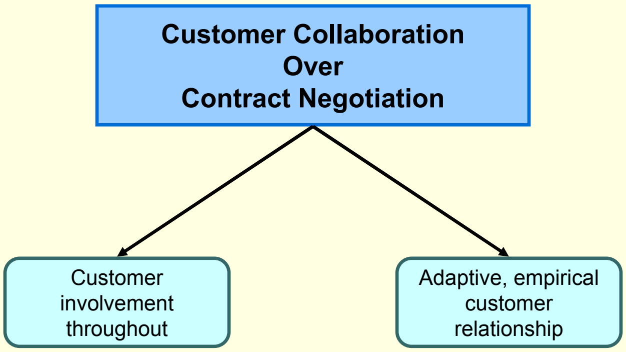
Working Software Over Comprehensive Documentation

- Iterative and incremental
- Working software is primary measure of progress
- Artifacts minimized

Mountain Goat SOFTWARE

Customer collaboration

What is agility?



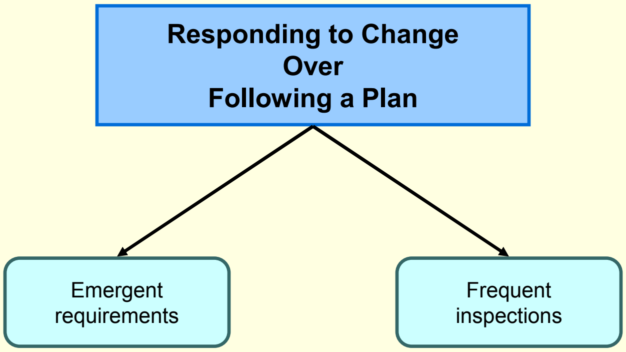
Customer Collaboration Over Contract Negotiation

- Customer involvement throughout
- Adaptive, empirical customer relationship

Mountain Goat SOFTWARE

What is agility?

Responding to Change



Responding to Change
Over
Following a Plan

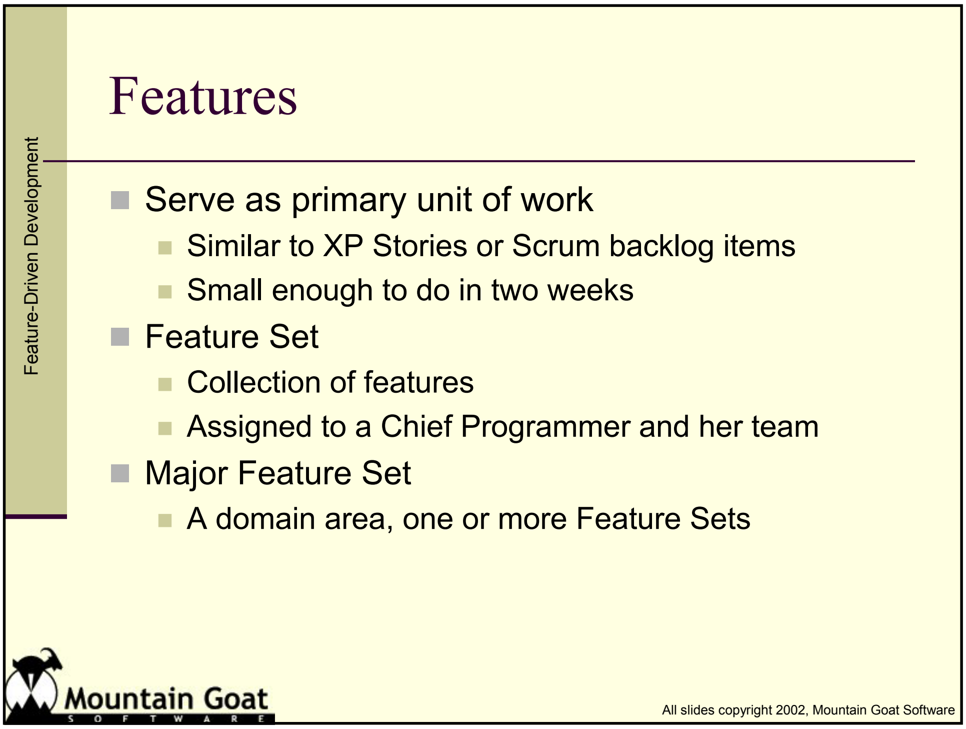
Emergent requirements

Frequent inspections

Feature-Driven Development

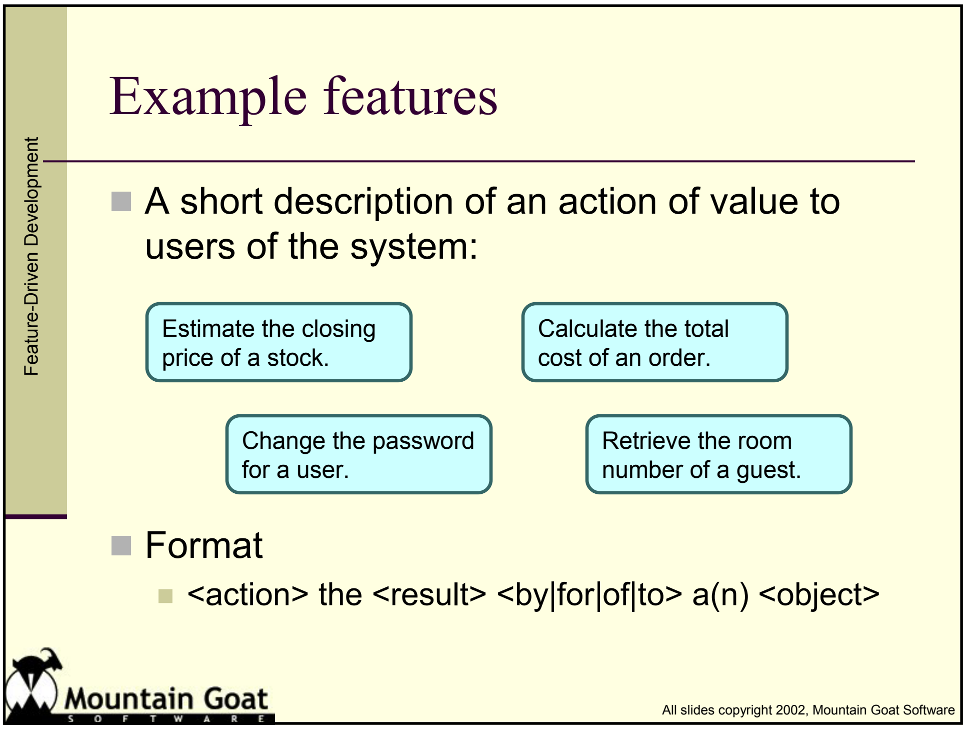
Feature-Driven Development

- Originates in *Java Modeling in Color with UML* by Coad, Lefebvre and De Luca in 1999
- Peter Coad
  - Founder of Togethersoft
  - Well-known OO methodologist
  - UML modeler
- Palmer and Felsing book in 2002



Features

- Serve as primary unit of work
  - Similar to XP Stories or Scrum backlog items
  - Small enough to do in two weeks
- Feature Set
  - Collection of features
  - Assigned to a Chief Programmer and her team
- Major Feature Set
  - A domain area, one or more Feature Sets



Example features

- A short description of an action of value to users of the system:

Estimate the closing price of a stock.

Calculate the total cost of an order.

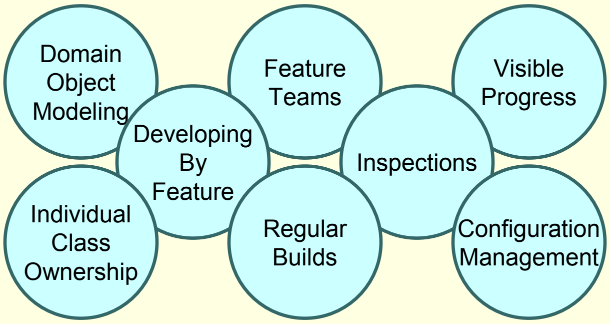
Change the password for a user.

Retrieve the room number of a guest.

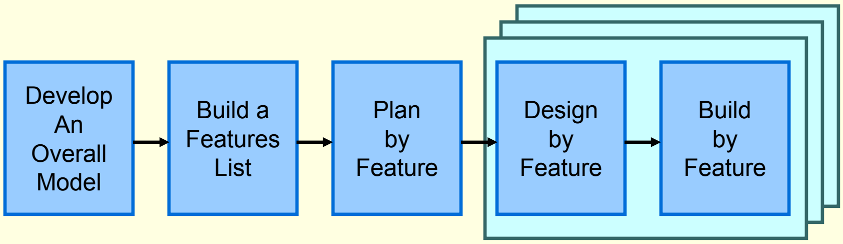
- Format
  - <action> the <result> <by|for|of|to> a(n) <object>

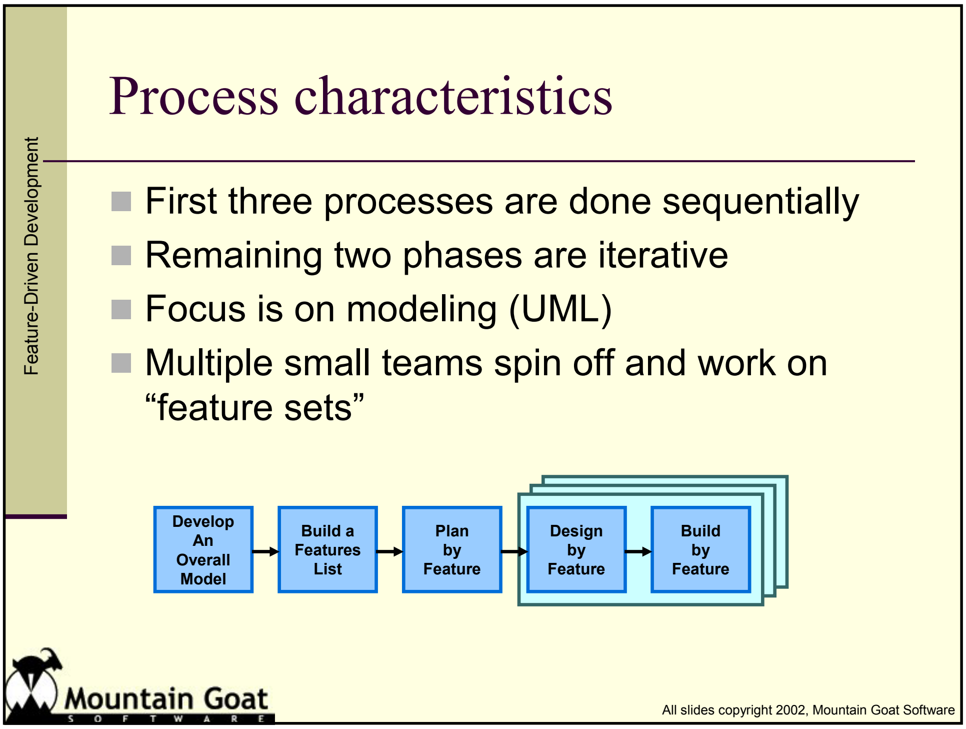
Eight “Best Practices”

- Need all 8 to be FDD



Five processes

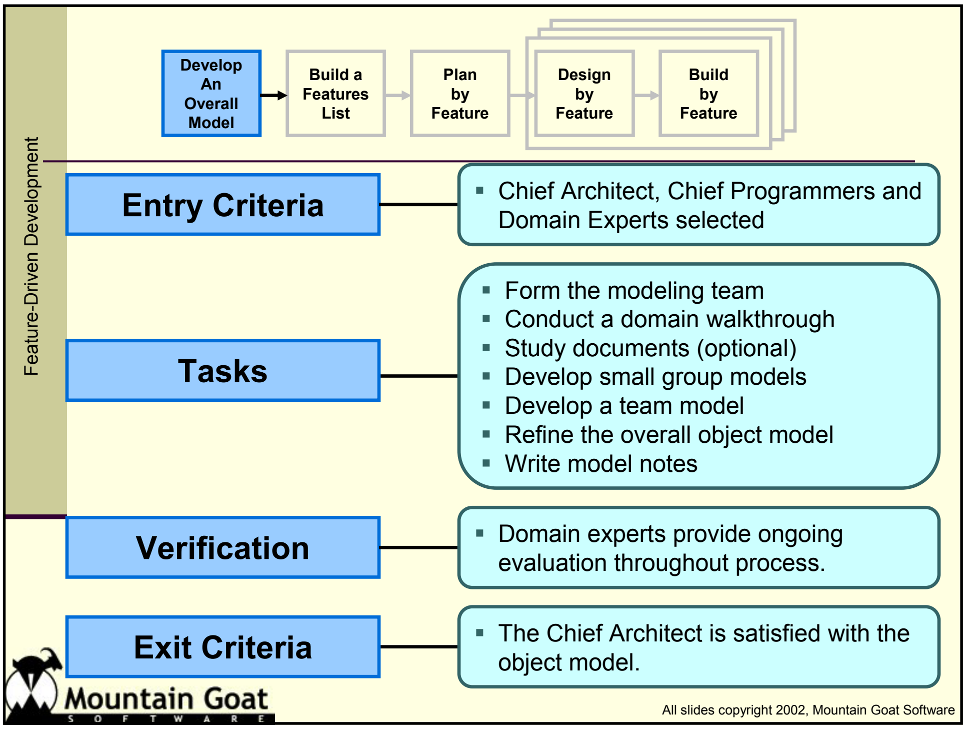




Feature-Driven Development

Process characteristics

- First three processes are done sequentially
- Remaining two phases are iterative
- Focus is on modeling (UML)
- Multiple small teams spin off and work on “feature sets”



Feature-Driven Development

Entry Criteria

- Chief Architect, Chief Programmers and Domain Experts selected

Tasks

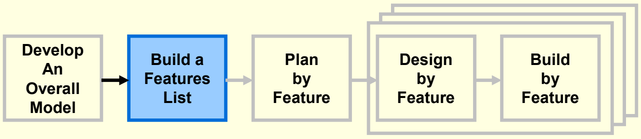
- Form the modeling team
- Conduct a domain walkthrough
- Study documents (optional)
- Develop small group models
- Develop a team model
- Refine the overall object model
- Write model notes

Verification

- Domain experts provide ongoing evaluation throughout process.

Exit Criteria

- The Chief Architect is satisfied with the object model.



Entry Criteria

- An overall model has been developed.

Tasks

- Form the Features List Team
- Build the Features List

Verification

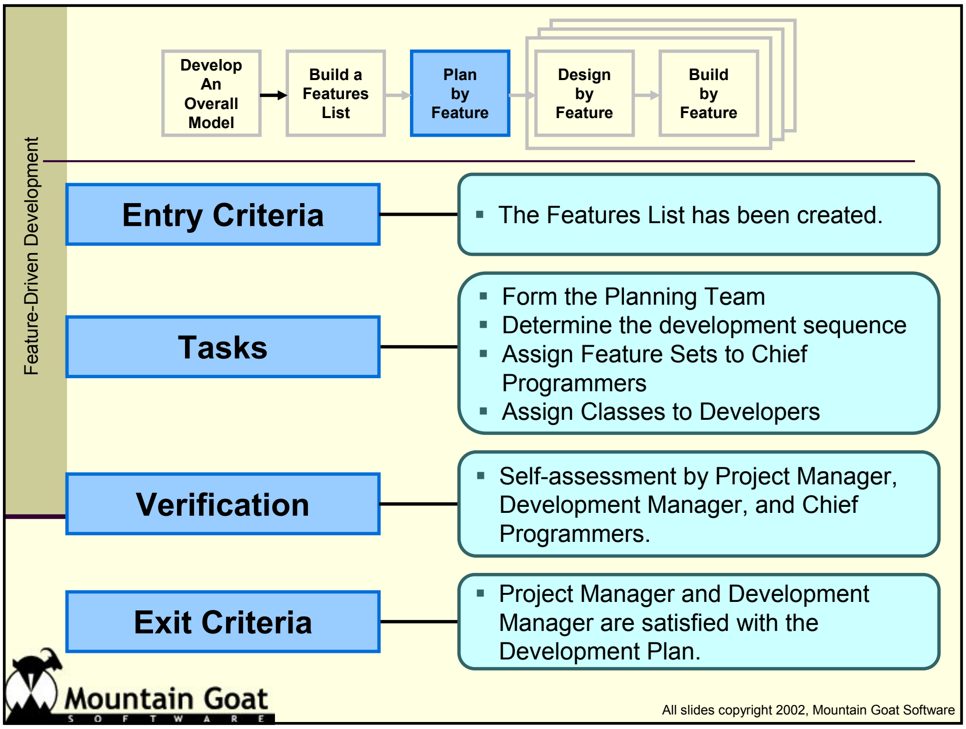
- Self-assessment by modelers on the features list team.
- External verification by Domain Experts as necessary.

Exit Criteria

- Project Manager and Development Manager are satisfied with Features List.

Sample Features List

- Making a reservation
  - Reserve a room for a guest
  - Look up a rate for a guest
  - …
- Reporting
  - Calculate RevPAR for a hotel
  - Calculate RevPAR for a Competitive Set
  - …



Develop An Overall Model → Build a Features List → Plan by Feature → Design by Feature → Build by Feature

Entry Criteria

- The Features List has been created.

Tasks

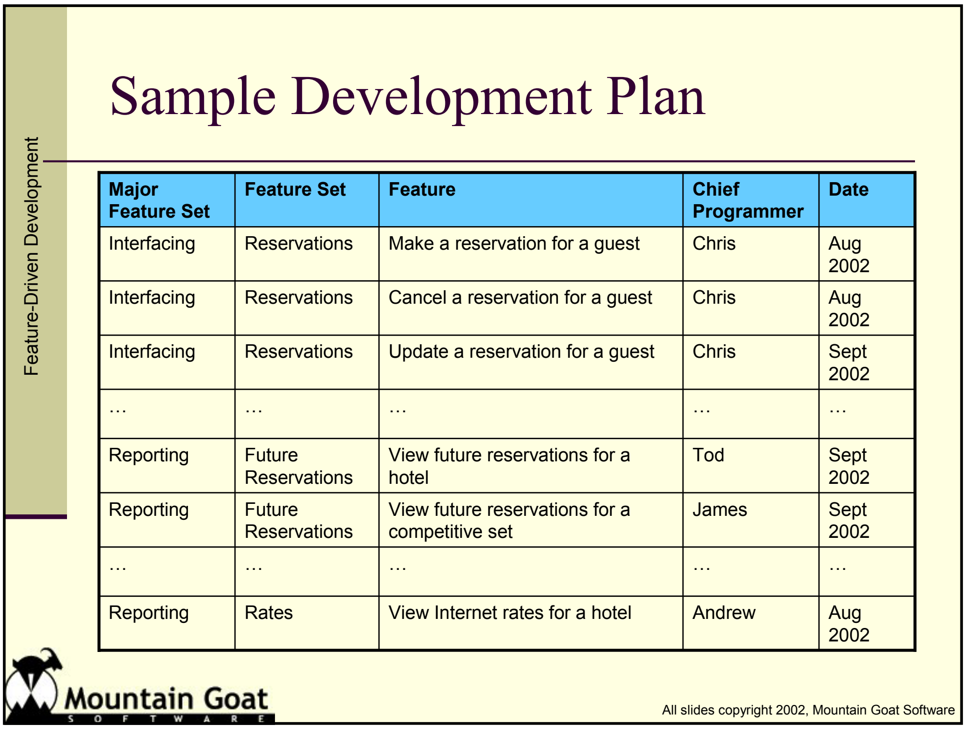
- Form the Planning Team
- Determine the development sequence
- Assign Feature Sets to Chief Programmers
- Assign Classes to Developers

Verification

- Self-assessment by Project Manager, Development Manager, and Chief Programmers.

Exit Criteria

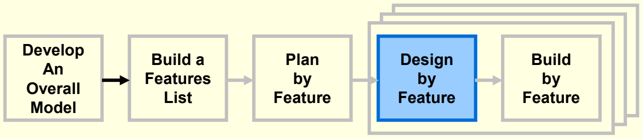
- Project Manager and Development Manager are satisfied with the Development Plan.



Sample Development Plan

Major Feature Set	Feature Set	Feature	Chief Programmer	Date
Interfacing	Reservations	Make a reservation for a guest	Chris	Aug 2002
Interfacing	Reservations	Cancel a reservation for a guest	Chris	Aug 2002
Interfacing	Reservations	Update a reservation for a guest	Chris	Sept 2002
…	…	…	…	…
Reporting	Future Reservations	View future reservations for a hotel	Tod	Sept 2002
Reporting	Future Reservations	View future reservations for a competitive set	James	Sept 2002
…	…	…	…	…
Reporting	Rates	View Internet rates for a hotel	Andrew	Aug 2002

Feature-Driven Development



Entry Criteria

- The Development Plan has been completed.

Tasks

- Form a Feature Team
- Conduct a domain walkthrough (optional)
- Study the referenced documents (optional)
- Develop the sequence diagrams
- Refine the object model
- Write class and method prologue
- Design inspection

Verification

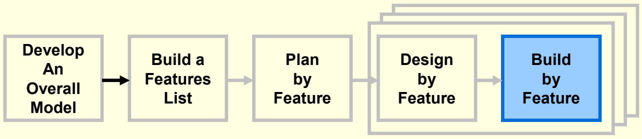
- The design inspection.

Exit Criteria

- A successful design inspection.

Mountain Goat SOFTWARE

Feature-Driven Development



Entry Criteria

- The Design by Feature process has been completed for the selected features.

Tasks

- Implement classes and methods.
- Conduct a code inspection.
- Unit test.
- Promote to the build.

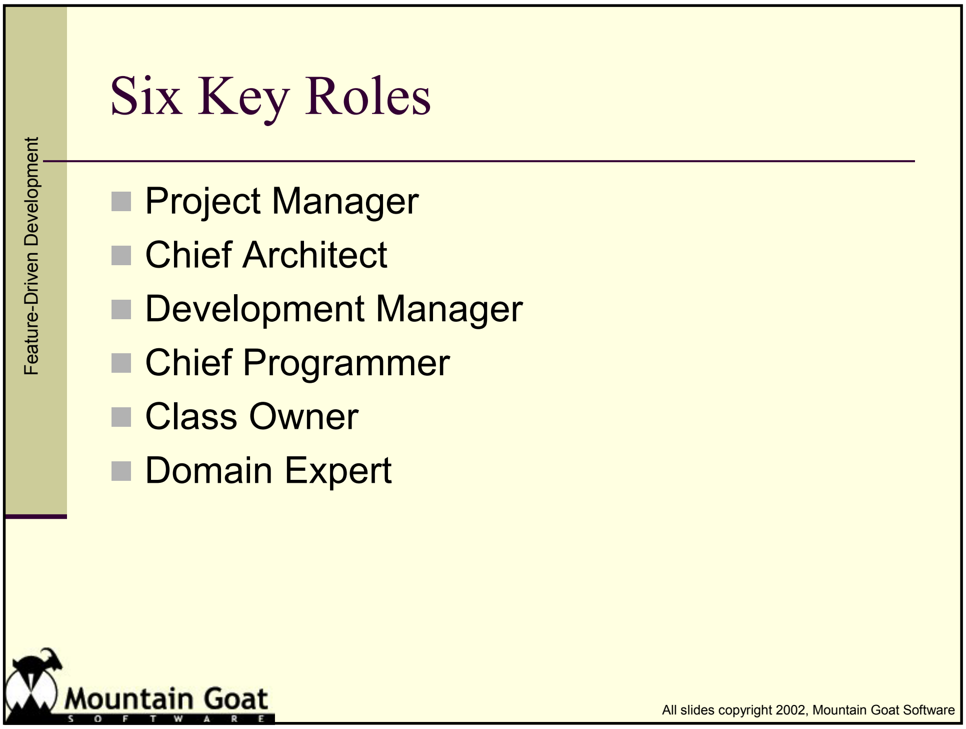
Verification

- A successful code inspection and passing the unit tests.

Exit Criteria

- Completion of at least one feature that is of value to (visible to) the client.

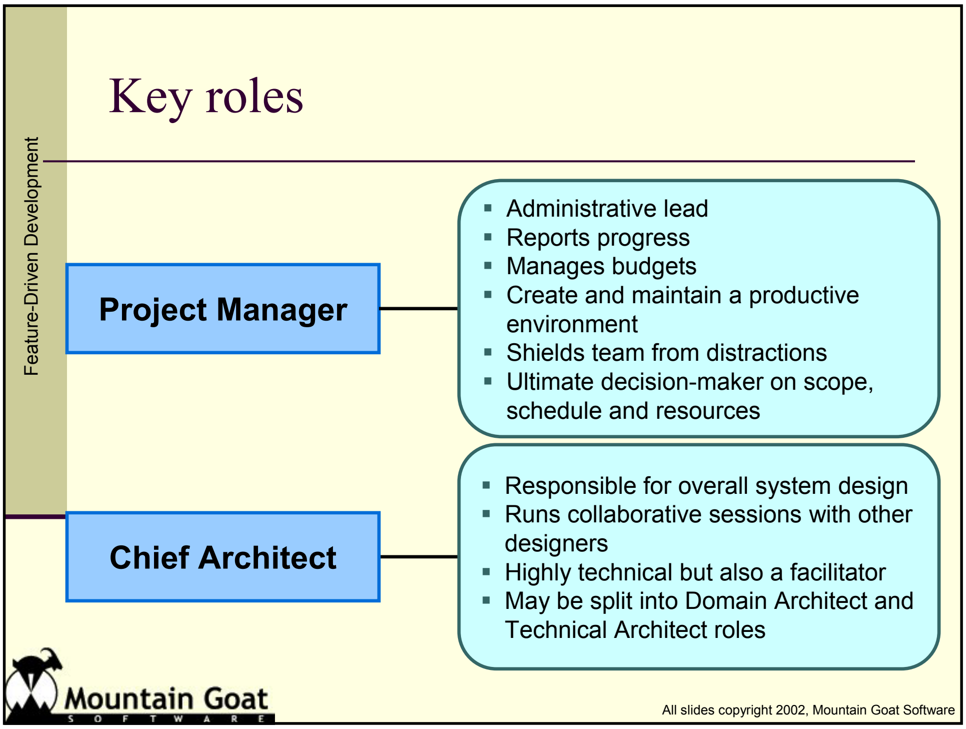
Mountain Goat SOFTWARE



Feature-Driven Development

Six Key Roles

- Project Manager
- Chief Architect
- Development Manager
- Chief Programmer
- Class Owner
- Domain Expert



Feature-Driven Development

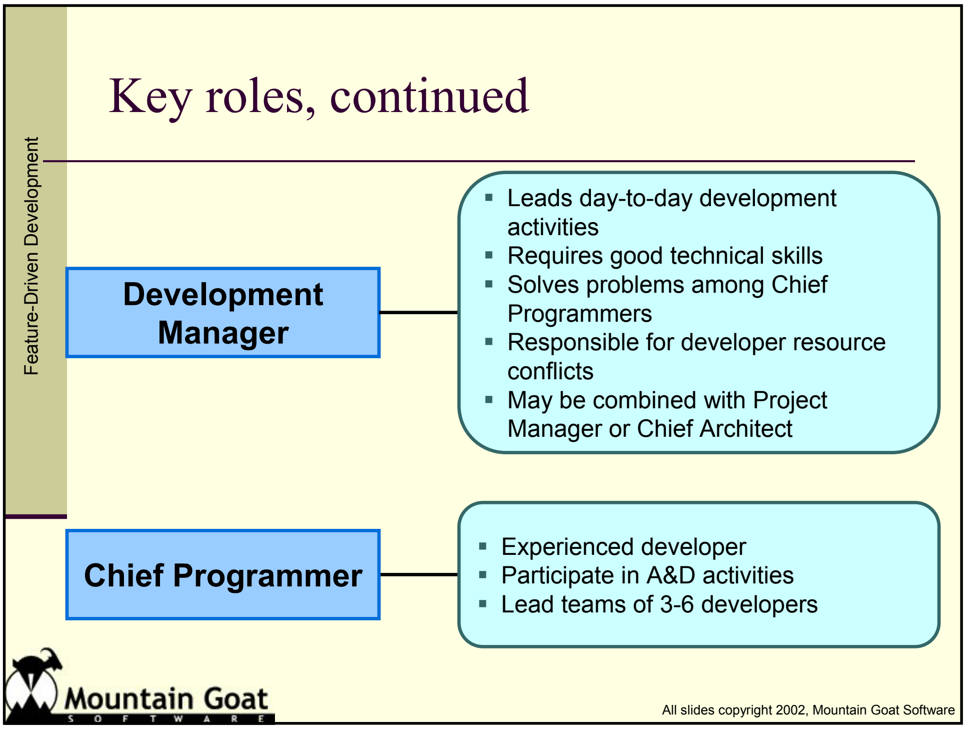
Key roles

Project Manager

- Administrative lead
- Reports progress
- Manages budgets
- Create and maintain a productive environment
- Shields team from distractions
- Ultimate decision-maker on scope, schedule and resources

Chief Architect

- Responsible for overall system design
- Runs collaborative sessions with other designers
- Highly technical but also a facilitator
- May be split into Domain Architect and Technical Architect roles



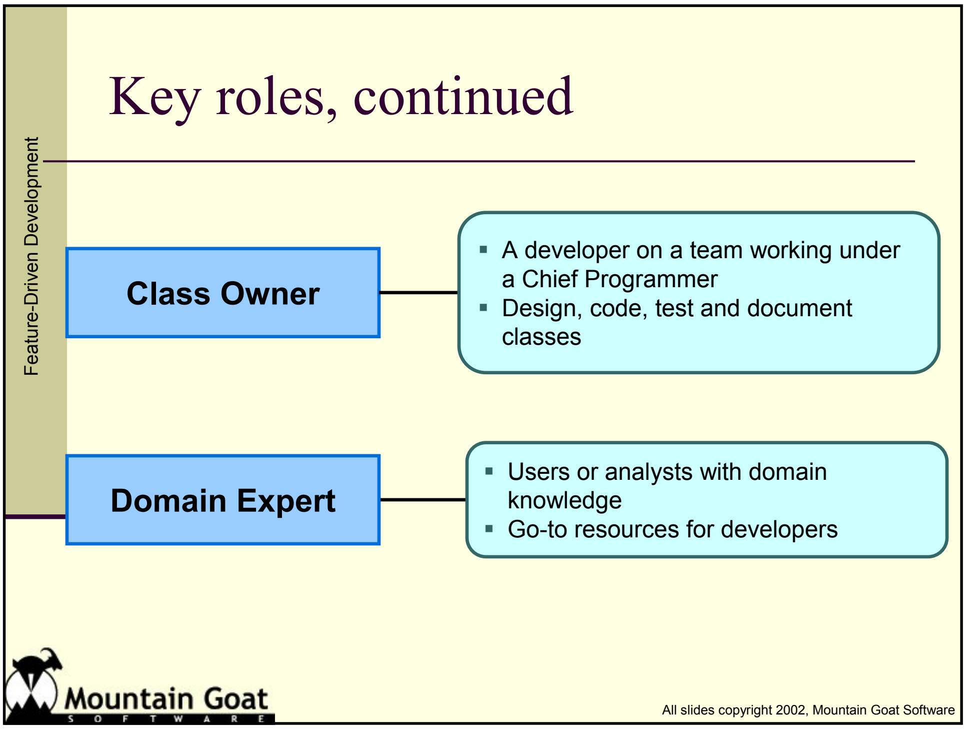
Key roles, continued

Development Manager

- Leads day-to-day development activities
- Requires good technical skills
- Solves problems among Chief Programmers
- Responsible for developer resource conflicts
- May be combined with Project Manager or Chief Architect

Chief Programmer

- Experienced developer
- Participate in A&D activities
- Lead teams of 3-6 developers



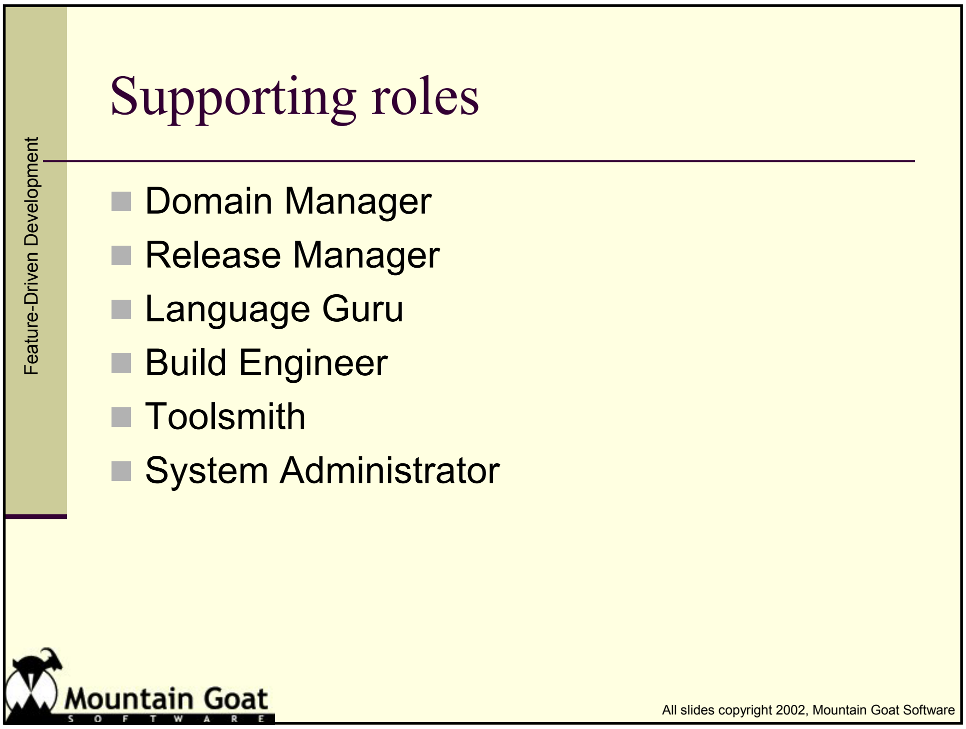
Key roles, continued

Class Owner

- A developer on a team working under a Chief Programmer
- Design, code, test and document classes

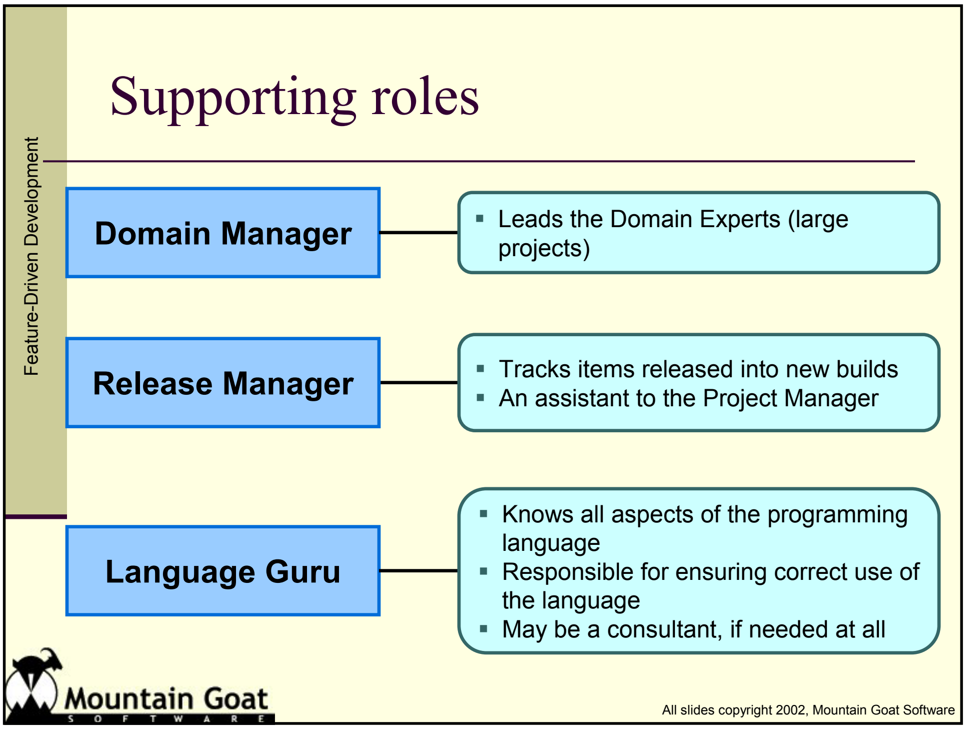
Domain Expert

- Users or analysts with domain knowledge
- Go-to resources for developers



Supporting roles

- Domain Manager
- Release Manager
- Language Guru
- Build Engineer
- Toolsmith
- System Administrator



Supporting roles

Domain Manager

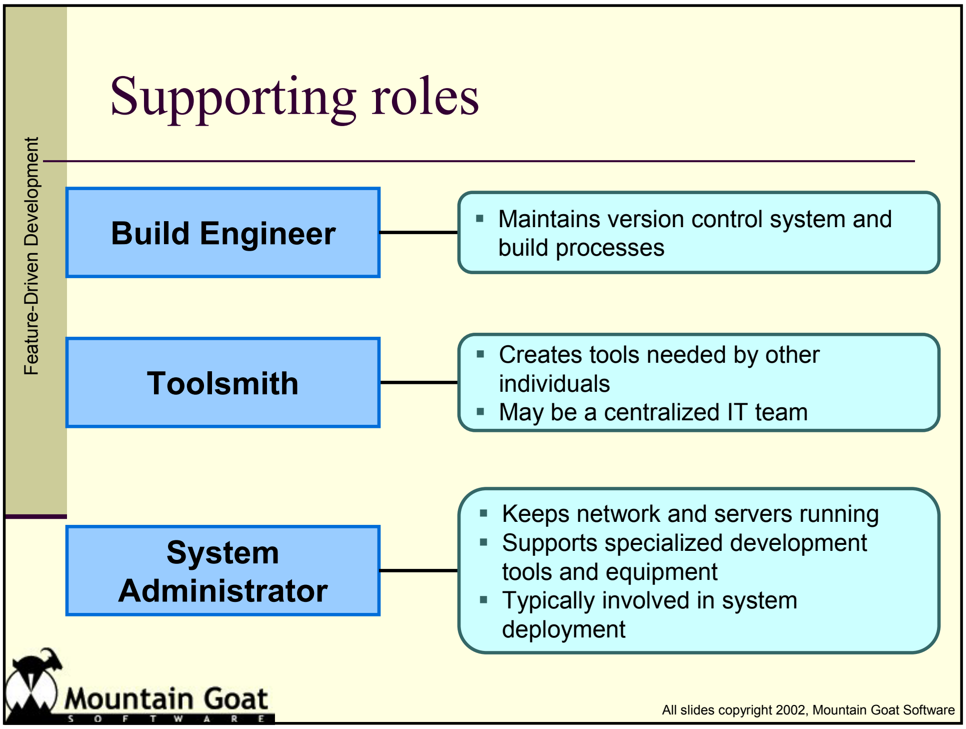
- Leads the Domain Experts (large projects)

Release Manager

- Tracks items released into new builds
- An assistant to the Project Manager

Language Guru

- Knows all aspects of the programming language
- Responsible for ensuring correct use of the language
- May be a consultant, if needed at all



Supporting roles

Build Engineer

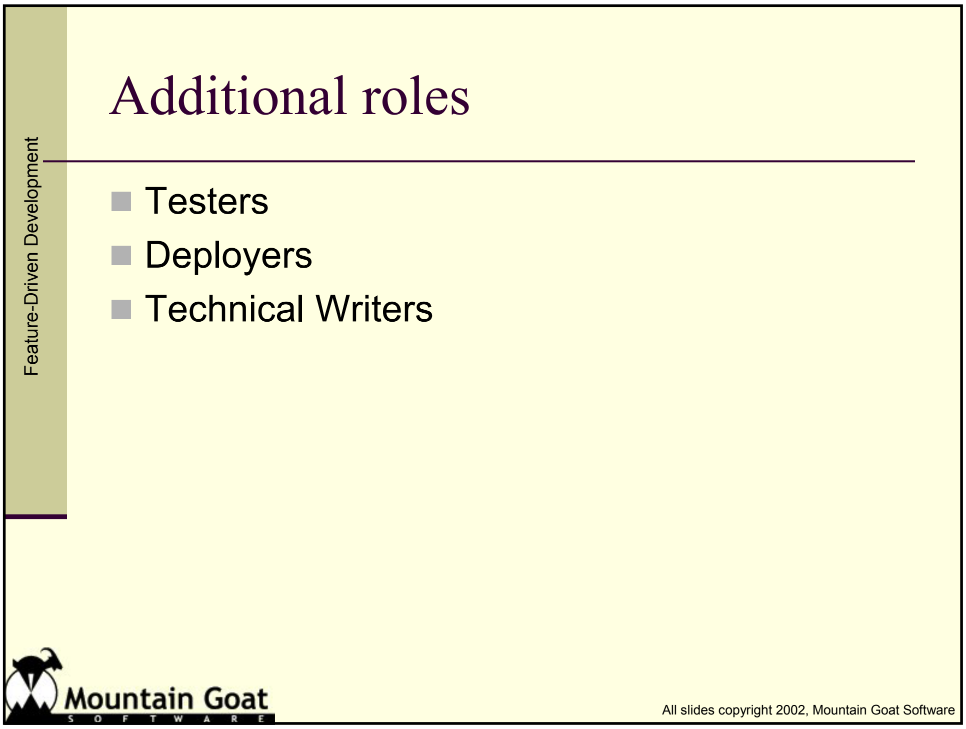
- Maintains version control system and build processes

Toolsmith

- Creates tools needed by other individuals
- May be a centralized IT team

System Administrator

- Keeps network and servers running
- Supports specialized development tools and equipment
- Typically involved in system deployment



Additional roles

- Testers
- Deployers
- Technical Writers

Feature-Driven Development

Additional roles

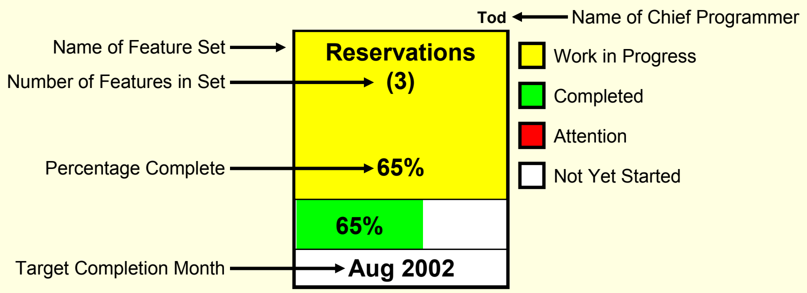
Testers
- Independently verify system meets requirements
- May be part of the project or a separate group

Deployers
- Plan and carry out physical deployment of new system
- Convert data from old system
- May be part of project or separate

Technical Writers
- Write online and printed documentation
- May be part of project or separate

Feature-Driven Development

Tracking progress



Feature-Driven Development

So where’s the testing?

- Testing is conspicuous by its absence
- Why?
  - FDD authors thought most organizations already have good test practices
    - Do they?
    - Are they complementary to FDD?
  - Wanted to address “core development processes”
    - Isn’t testing “core”?
- Why else?
  - Testing doesn’t sell UML tools

TogetherSoft

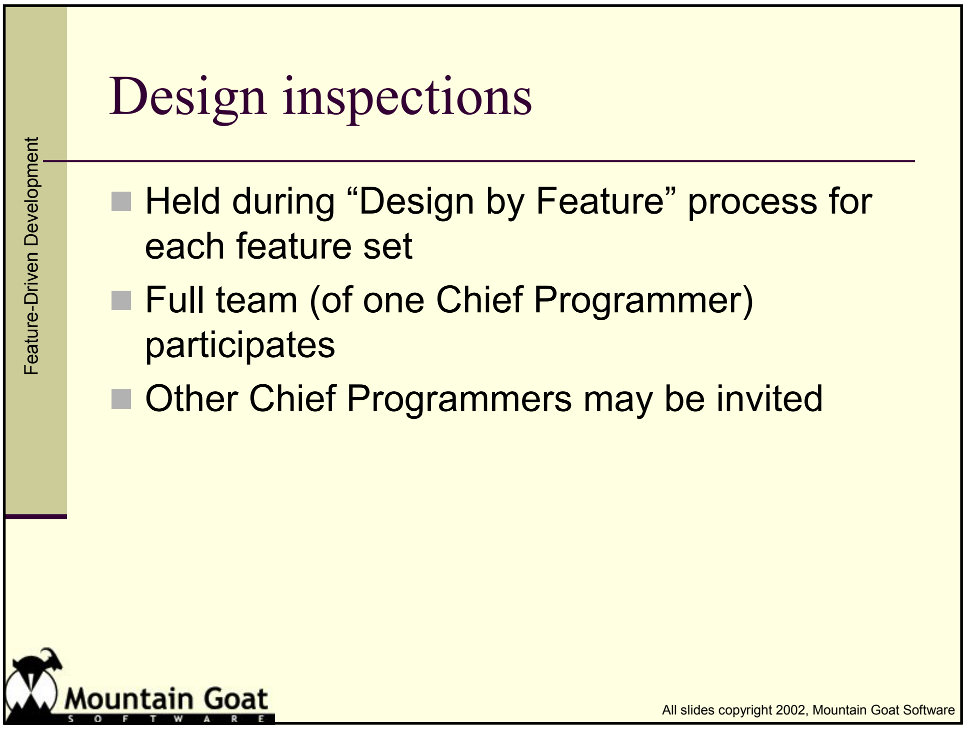
Mountain Goat SOFTWARE

Feature-Driven Development

Unit testing

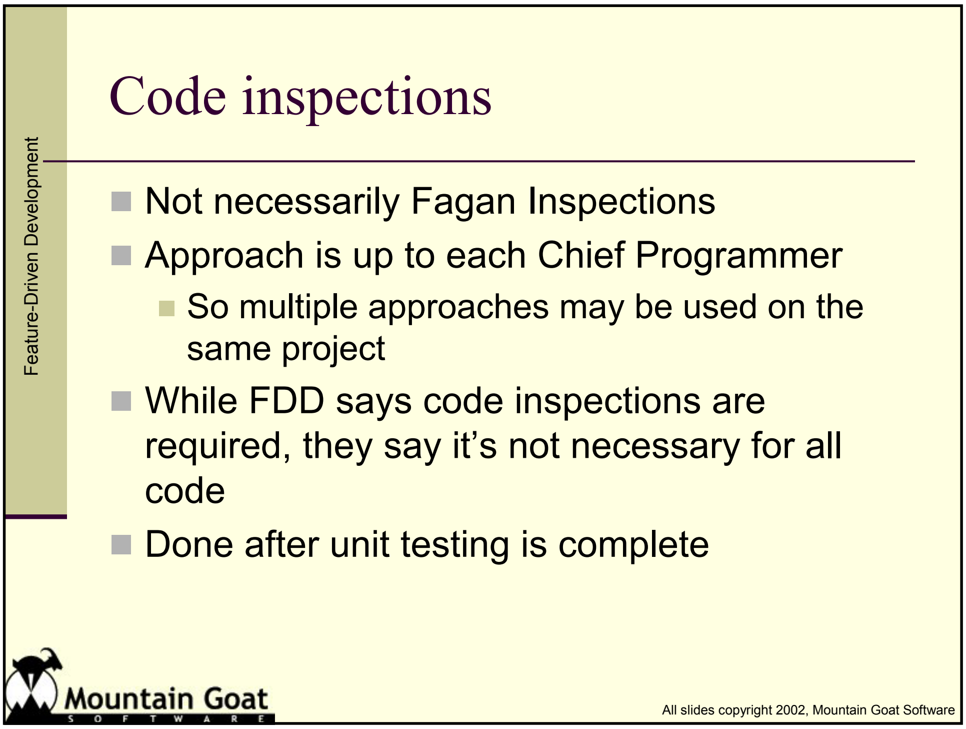
- The “Build by Feature” process does require unit testing
- Approach is left up to the Chief Programmers
  - Can be very different on projects with multiple Chief Programmers
- FDD requires “regular” builds
  - Not necessarily continuous builds

Mountain Goat SOFTWARE



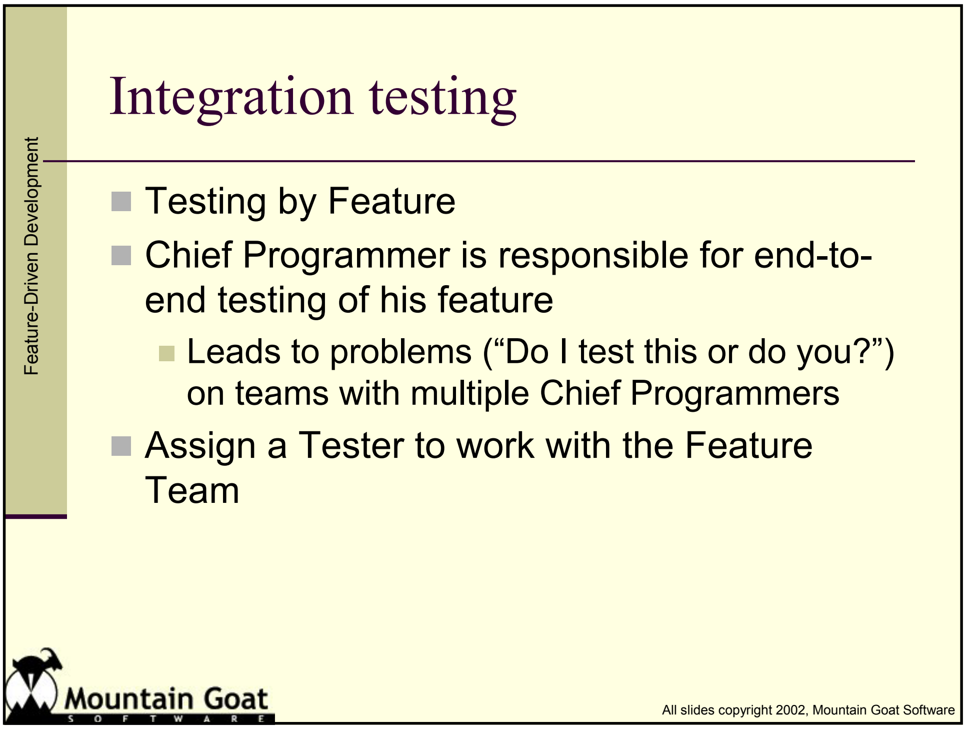
Design inspections

- Held during “Design by Feature” process for each feature set
- Full team (of one Chief Programmer) participates
- Other Chief Programmers may be invited



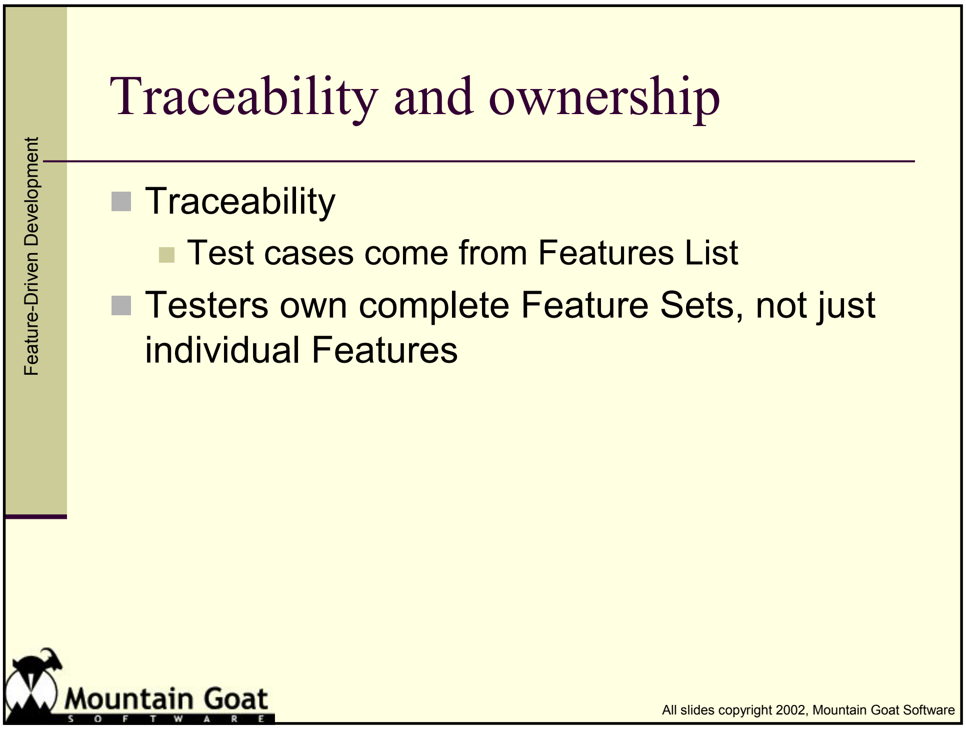
Code inspections

- Not necessarily Fagan Inspections
- Approach is up to each Chief Programmer
  - So multiple approaches may be used on the same project
- While FDD says code inspections are required, they say it’s not necessary for all code
- Done after unit testing is complete



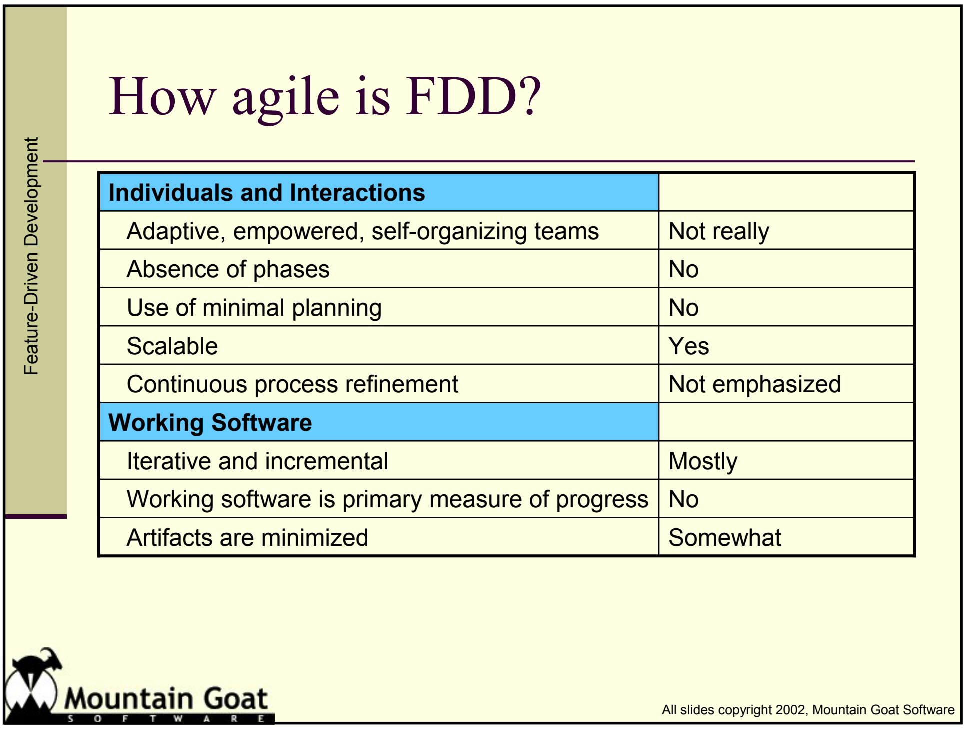
Integration testing

- Testing by Feature
- Chief Programmer is responsible for end-to-end testing of his feature
  - Leads to problems (“Do I test this or do you?”) on teams with multiple Chief Programmers
- Assign a Tester to work with the Feature Team



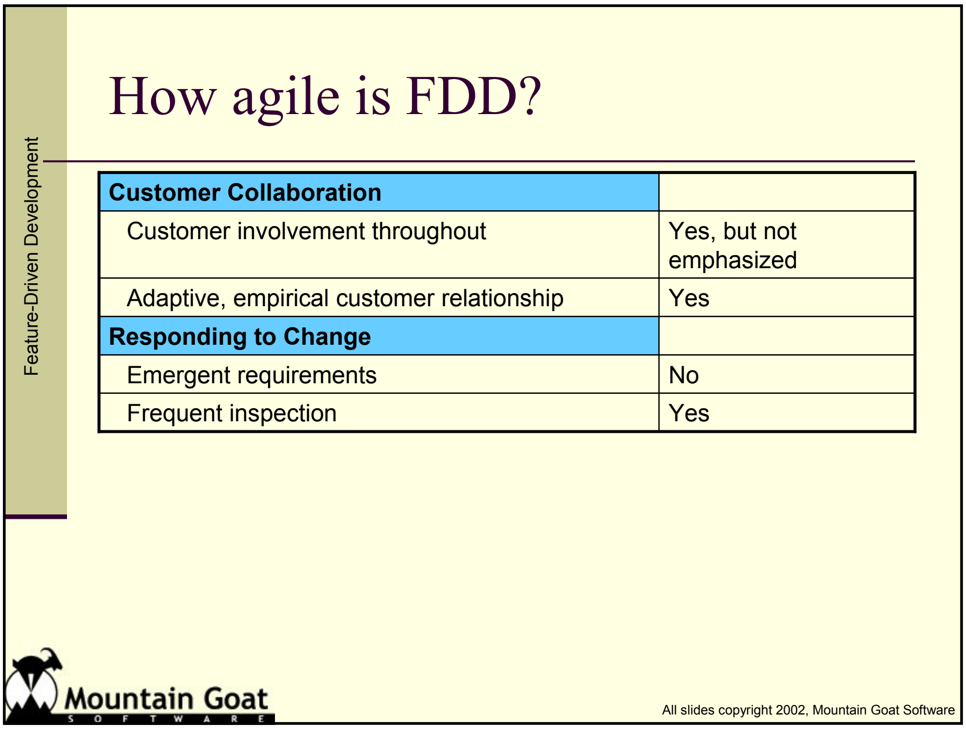
Traceability and ownership

- Traceability
  - Test cases come from Features List
- Testers own complete Feature Sets, not just individual Features



How agile is FDD?

Individuals and Interactions	
Adaptive, empowered, self-organizing teams	Not really
Absence of phases	No
Use of minimal planning	No
Scalable	Yes
Continuous process refinement	Not emphasized
Working Software	
Iterative and incremental	Mostly
Working software is primary measure of progress	No
Artifacts are minimized	Somewhat



How agile is FDD?

Customer Collaboration	
Customer involvement throughout	Yes, but not emphasized
Adaptive, empirical customer relationship	Yes
Responding to Change	
Emergent requirements	No
Frequent inspection	Yes

Scrum

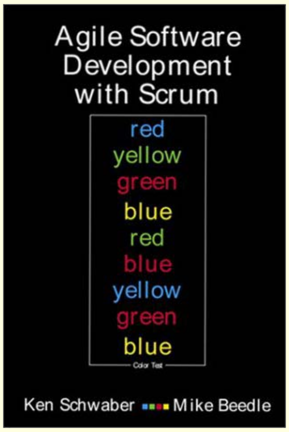
Scrum

- "The New New Product Development Game" in *Harvard Business Review*, 1986.
  - "The… 'relay race' approach to product development…may conflict with the goals of maximum speed and flexibility. Instead a holistic or 'rugby' approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today's competitive requirements."
- *Wicked Problems, Righteous Solutions* by DeGrace and Stahl, 1990.
  - This is where Scrum was first mentioned in a software context.

Scrum

Scrum origins

- Jeff Sutherland
  - Initial Scrums at Easel Corp in 1993
  - IDX and nearly 600 people doing Scrum
  - Not just for trivial projects
    - FDA-approved, life-critical software for x-rays and MRIs
- Ken Schwaber
  - ADM
  - Initial definitions of Scrum at OOPSLA 96 with Sutherland
- Mike Beedle
  - Scrum patterns in PLOPD4



Agile Software Development with Scrum

red
yellow
green
blue
red
blue
yellow
green
blue

Color Test

Ken Schwaber Mike Beedle

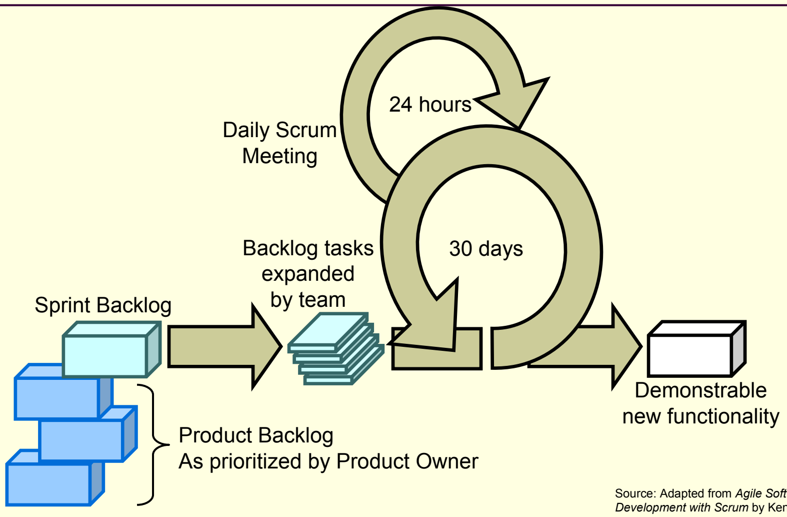
Scrum

Characteristics

- Self-organizing teams
- Product progresses in a series of month-long “sprints”
- Requirements are captured as items in a list of “product backlog”
- No specific engineering practices prescribed
- Uses generative rules to create an agile environment for delivering projects

Mountain Goat SOFTWARE

Overview

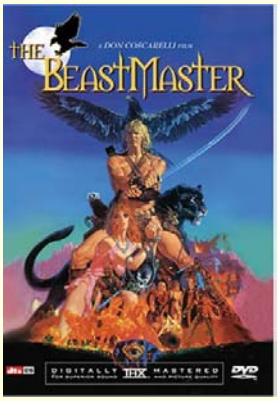

24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Demonstrable new functionality
Product Backlog
As prioritized by Product Owner

Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Mountain Goat SOFTWARE

The Scrum Master

- Represents management to the project
- Typically filled by a Project Manager or Team Leader
- Responsible for enacting Scrum values and practices
- Main job is to remove impediments



The Scrum Team

- Typically 5-10 people
- Cross-functional
  - QA, Programmers, UI Designers, etc.
- Members should be full-time
  - May be exceptions (e.g., System Admin, etc.)
- Teams are self-organizing
  - What to do if a team self-organizes someone off the team??
  - No titles
- Membership can change only between sprints

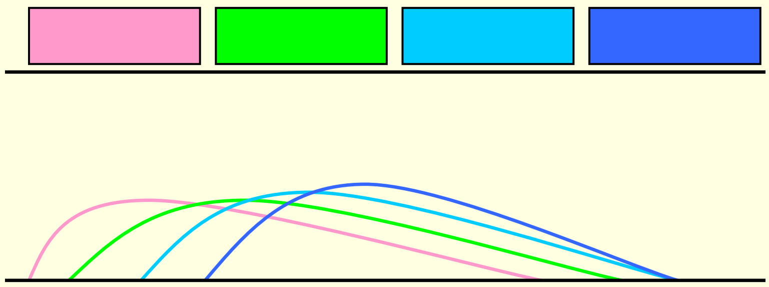
Scrum

Sprints

- Scrum projects make progress in a series of "sprints"
  - Analogous to XP iterations
- Target duration is one month
  - +/- a week or two
- Product is designed, coded, and tested during the sprint

Scrum

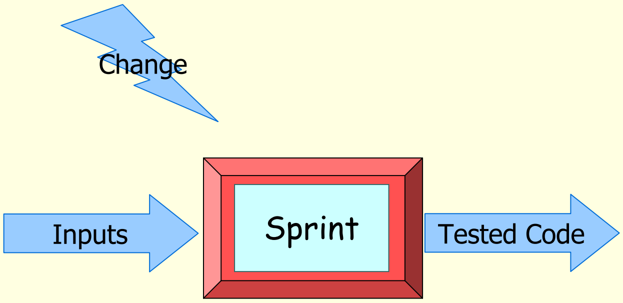
Sequential vs. Overlapping Development



Source: "The New New Product Development Game", Hirotaka Takeuchi and Ikujiro Nonaka, *Harvard Business Review*, January 1986.

Scrum

No changes during the sprint



- Plan sprint durations around how long you can commit to keeping change out of the sprint

Mountain Goat
SOFTWARE

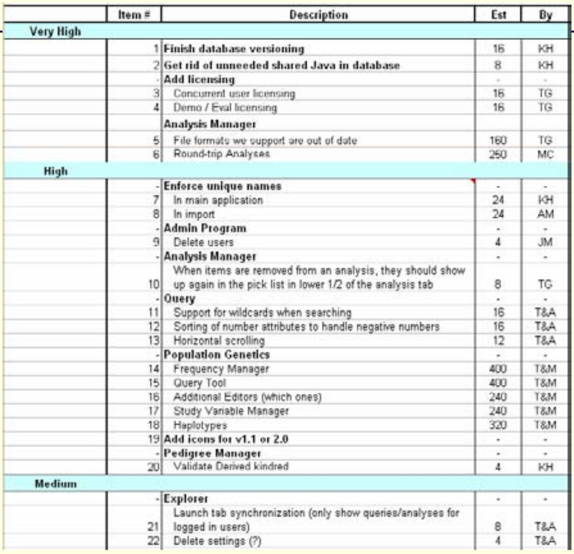
Scrum

Product Backlog

- A list of all desired work on the project
  - Usually a combination of
    - story-based work (“let user search and replace”)
    - task-based work (“improve exception handling”)
- List is prioritized by the Product Owner
  - Typically a Product Manager, Marketing, Internal Customer, etc.

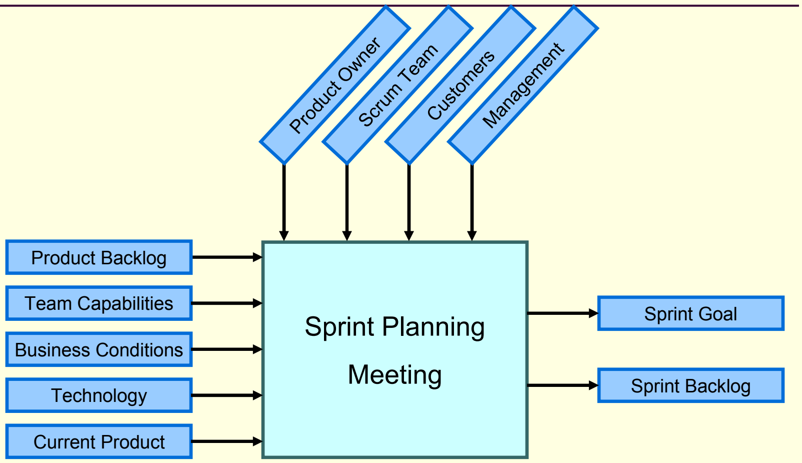
Mountain Goat
SOFTWARE

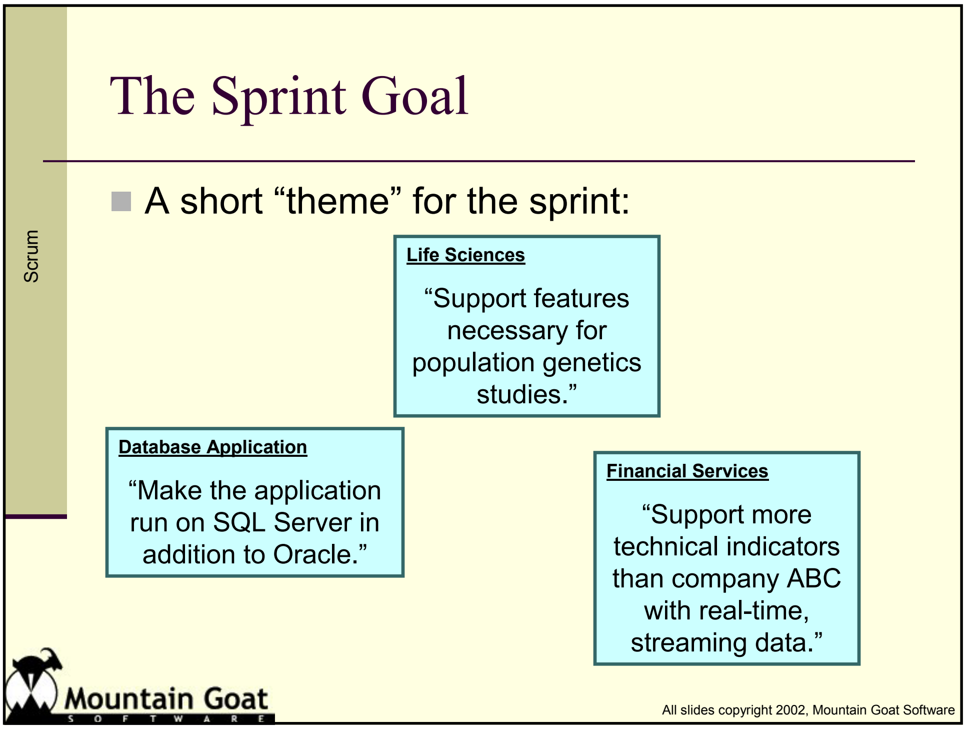
Sample Product Backlog



	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Sprint Planning Meeting





The Sprint Goal

- A short "theme" for the sprint:

Life Sciences

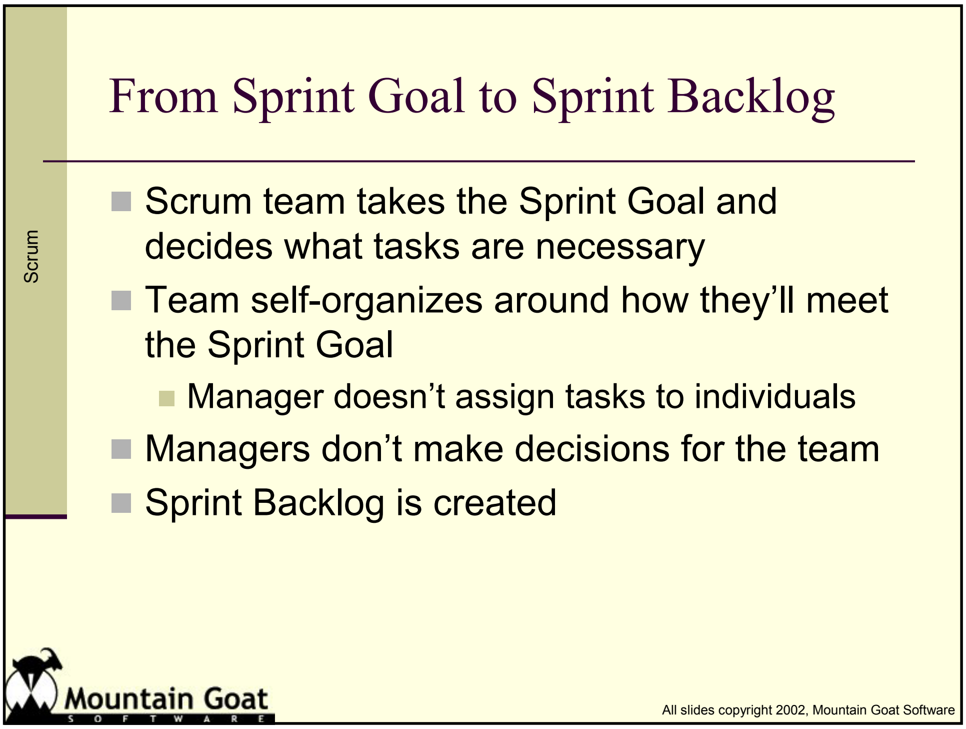
"Support features necessary for population genetics studies."

Database Application

"Make the application run on SQL Server in addition to Oracle."

Financial Services

"Support more technical indicators than company ABC with real-time, streaming data."

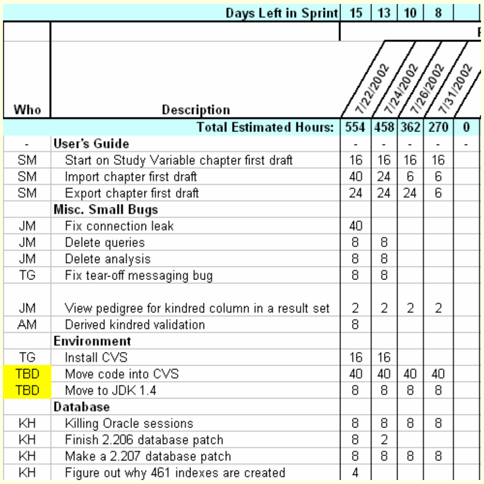


From Sprint Goal to Sprint Backlog

- Scrum team takes the Sprint Goal and decides what tasks are necessary
- Team self-organizes around how they'll meet the Sprint Goal
  - Manager doesn't assign tasks to individuals
- Managers don't make decisions for the team
- Sprint Backlog is created

Scrum

Sample Sprint Backlog



	Days Left in Sprint	15	13	10	8	
Who	Description	7/22/2002	7/24/2002	7/26/2002	7/31/2002	
	Total Estimated Hours:	554	458	362	270	0
-	**User's Guide**	-	-	-	-	-
SM	Start on Study Variable chapter first draft	16	16	16	16	
SM	Import chapter first draft	40	24	6	6	
SM	Export chapter first draft	24	24	24	6	
	Misc. Small Bugs					
JM	Fix connection leak	40				
JM	Delete queries	8	8			
JM	Delete analysis	8	8			
TG	Fix tear-off messaging bug	8	8			
JM	View pedigree for kindred column in a result set	2	2	2	2	
AM	Derived kindred validation	8				
	Environment					
TG	Install CVS	16	16			
TBD	Move code into CVS	40	40	40	40	
TBD	Move to JDK 1.4	8	8	8	8	
	Database					
KH	Killing Oracle sessions	8	8	8	8	
KH	Finish 2.206 database patch	8	2			
KH	Make a 2.207 database patch	8	8	8	8	
KH	Figure out why 461 indexes are created	4				

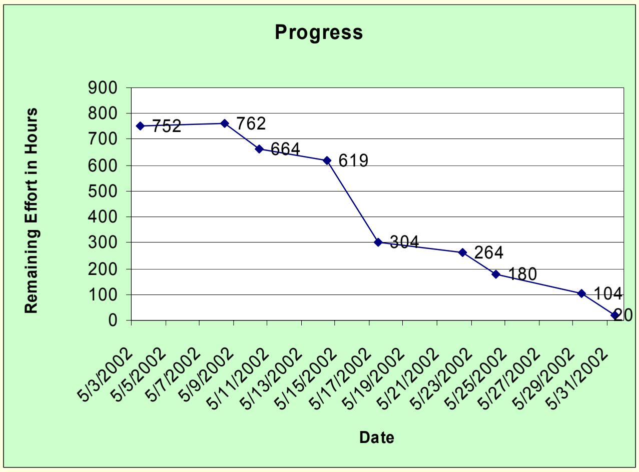
Scrum

Sprint Backlog during the Sprint

- Changes
  - Team adds new tasks whenever they need to in order to meet the Sprint Goal
  - Team can remove unnecessary tasks
  - But: Sprint Backlog can only be updated by the team
- Estimates are updated whenever there's new information

Scrum

Sprint Burndown Chart

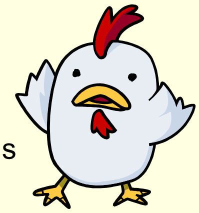


Mountain Goat SOFTWARE

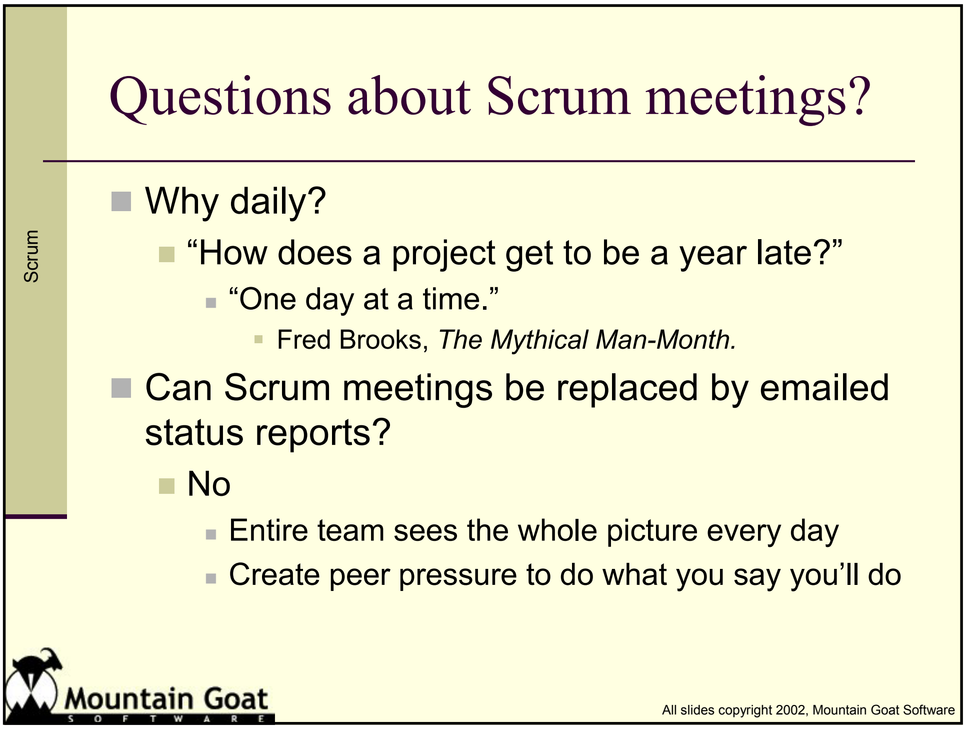
Scrum

Daily Scrum meetings

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
  - Not for problem solving
- Three questions:
  1. What did you do yesterday
  2. What will you do today?
  3. What obstacles are in your way?
- Chickens and pigs are invited
  - Help avoid other unnecessary meetings
- Only pigs can talk

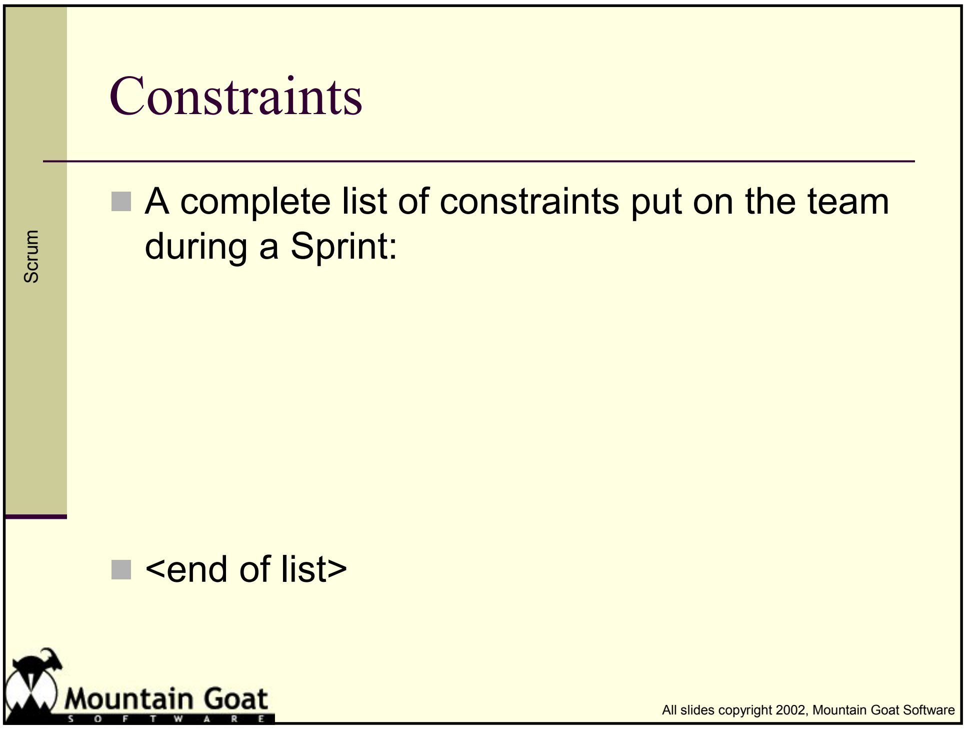


Mountain Goat SOFTWARE



Questions about Scrum meetings?

- Why daily?
  - "How does a project get to be a year late?"
    - "One day at a time."
      - Fred Brooks, *The Mythical Man-Month.*
- Can Scrum meetings be replaced by emailed status reports?
  - No
    - Entire team sees the whole picture every day
    - Create peer pressure to do what you say you'll do



Constraints

- A complete list of constraints put on the team during a Sprint:

- <end of list>

Sprint Review Meeting

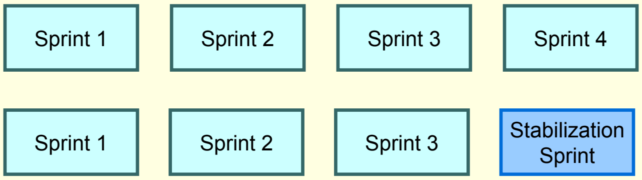
- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
- Participants
  - Customers
  - Management
  - Product Owner
  - Other engineers

Testing & Scrum

- Scrum doesn’t specify any specific engineering practices
- However, each sprint is required to produce ready-to-use code
  - Heavy in-sprint testing is usually applied
  - Some teams have dedicated testers
    - Others have programmers test everything
- Other engineering practices are up to you
  - Automation, code inspection, pair programming, static analysis tools, etc.

Scrum

Stabilization Sprints



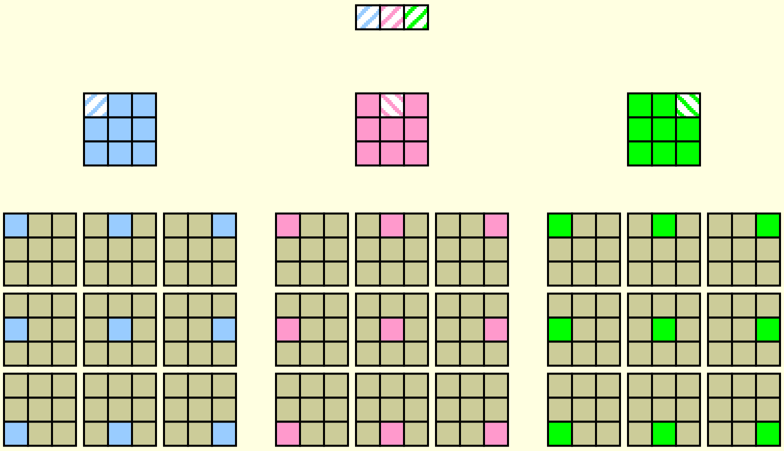
- Team focuses entirely on defects
  - Prepares a product for release
  - Useful during
    - active beta periods
    - when transitioning a team to Scrum
    - if quality isn’t quite where it should be on an initial release
- Not a part of standard Scrum, just something I’ve found useful

Scrum

Scalability of Scrum

- Typical Scrum team is 5-10 people
- Sutherland used Scrum in groups of 600+
- I’ve used in groups 100+

Scrum of Scrums / Meta-Scrum



How agile is Scrum?

Individuals and Interactions	
Adaptive, empowered, self-organizing teams	Yes
Absence of phases	Yes
Use of minimal planning	Yes
Scalable	Yes
Continuous process refinement	Yes
Working Software	
Iterative and incremental	Yes
Working software is primary measure of progress	Yes
Artifacts are minimized	Yes

How agile is Scrum?

Customer Collaboration	
Customer involvement throughout	Yes
Adaptive, empirical customer relationship	Yes
Responding to Change	
Emergent requirements	Yes
Frequent inspection	Yes

Extreme Programming (XP)

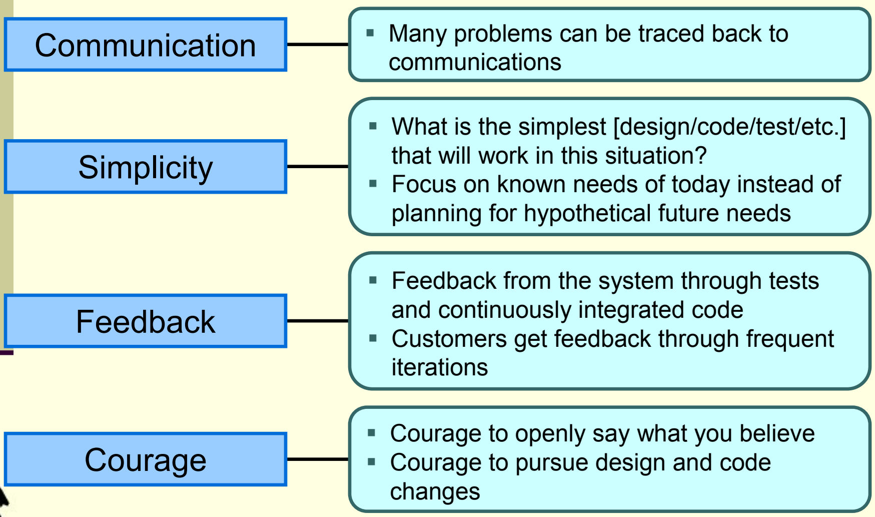
- The Three Extremos
  - Kent Beck
  - Ward Cunningham
  - Ron Jeffries
- C3 Project

extreme Programming explained
EMBRACE CHANGE
Kent Beck
Foreword by Erich Gamma
Extreme Programming Installed
Ron Jeffries
Questioning Extreme Programming
The XP Series
Pete McBreen
Foreword by Kent Beck
A Practical Guide to eXtreme programming
David Astels · Granville Miller
Miroslav Novak
THE COAD SERIES
Planning Extreme Programming
Kent Beck
Martin Fowler

Characteristics

- “Turning all the dials up to 10”
- 1-3 week iterations
- Stories
- On-site customer
- Heavy, heavy emphasis on unit testing
- Do the simplest thing possible
- You Aren’t Gonna Need It (YAGNI)

Core values



Communication
- Many problems can be traced back to communications

Simplicity
- What is the simplest [design/code/test/etc.] that will work in this situation?
- Focus on known needs of today instead of planning for hypothetical future needs

Feedback
- Feedback from the system through tests and continuously integrated code
- Customers get feedback through frequent iterations

Courage
- Courage to openly say what you believe
- Courage to pursue design and code changes

~~12~~ 13 Practices

- Whole Team (On-site customer)
- Small releases
- The Planning Game
- Simple design
- Pair programming
- Test Driven Development
- Customer Tests
- Refactoring (Design Improvement)
- Collective code ownership
- Coding standard
- Continuous integration
- Metaphor
- Sustainable Pace

Practice 1

Whole Team / On-site customer

- Everyone sits together in one room
- A real customer sits with the development team
  - May be a customer proxy when a real customer isn't available (e.g., ISV)
- If the business can't spare a customer, is the project worth doing?
- The customer
  - Writes stories
  - Writes acceptance tests

Stories

- Method for expressing functionality in XP
  - Analogous to use cases or requirements
- Also used for tracking progress

View an existing reservation

Present the customer with a list of reservations he's made.

Track preferences

Keep track of the types of hotel (e.g., Marriott, 4-star, etc.) that a customer stays at.

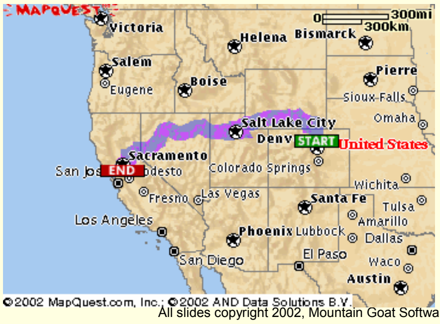
Sort hotels

Allow the customer to sort hotels by various attributes (e.g., class, price, name).

Practice 2

Small releases

- Plan only as far in advance as you can see
- Adjust the plan as necessary
- Each release is as small as possible to actually deliver something of value
  - Typically 1-3 weeks
- Do not need to deploy

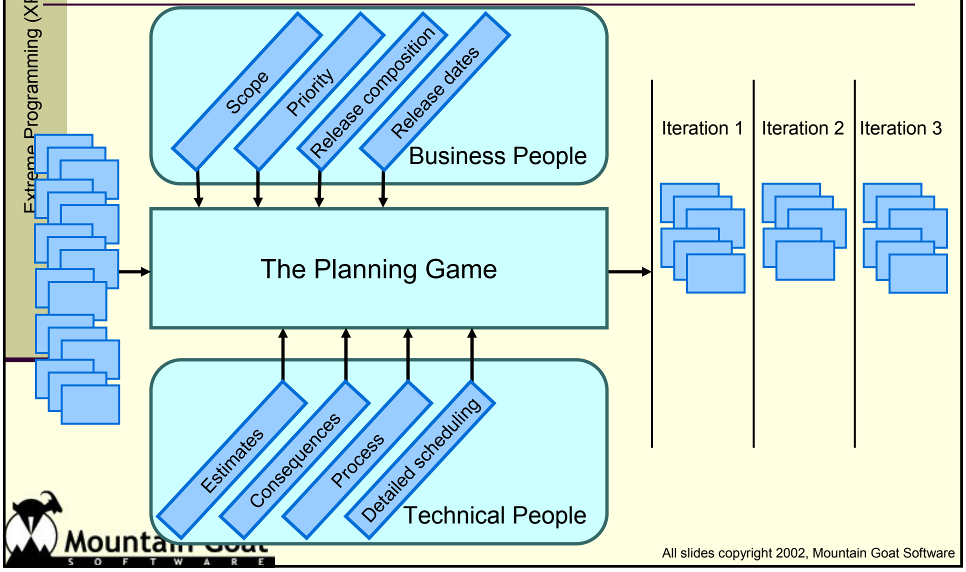


©2002 MapQuest.com, Inc.; ©2002 AND Data Solutions B.V.

Extreme Programming (XP)

Practice 3

The Planning Game



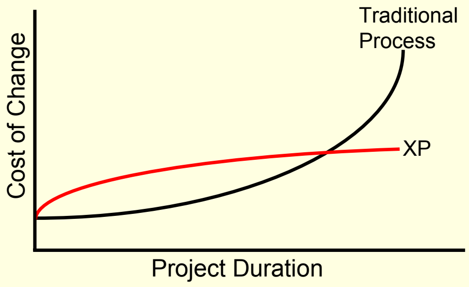
Extreme Programming (XP)

The Cost of Change

- "The error [is] typically 100 times more expensive to correct in the maintenance phase than in the requirements phase."
  - *Software Engineering Economics*, Barry Boehm, 1981, p. 40.

Extreme Programming (XP)

The Cost of Change



Extreme Programming (XP)

Practice 4

Simple design

- Design only for today
- If the future is uncertain, don’t code for it today
- Do not add infrastructure *in this iteration* for stories coming *in future iterations*
  - Upcoming stories could be cancelled or lowered in priority
- YAGNI
- Do the simplest thing that can possibly work

Extreme Programming (XP)

Practice 5

Pair programming

- Two programmers at one computer
  - The driver
    - has the keyboard
    - focuses on the tactical aspects of writing the code
  - Partner
    - Watches the forest, not the trees
    - Thinks about missing tests, integration issues, etc.
- Keep each other “honest”
  - A lot of XP requires great discipline
- Programming is far more than typing
- Pairs constantly shift

Mountain Goat SOFTWARE

Extreme Programming (XP)

Practice 6

Test-Driven Development (TDD)

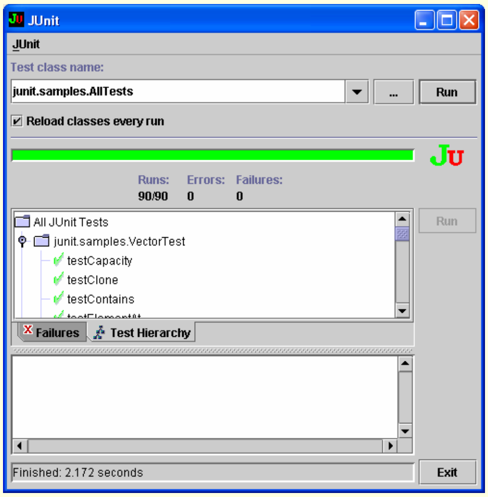
- Write the unit tests first, then write the code
- “Any program feature without an automated test simply doesn’t exist.”
  - —Kent Beck

Mountain Goat SOFTWARE

JUnit

- A framework for automated unit testing
- Programmers write tests in their Java code
  - JUnit executes TestCases and TestSuites
  - Provides instant feedback on whether the code works
- If each programmer writes JUnit TestCases…
- Details are at: www.junit.org
- Other xUnit test frameworks exist (VB, http, etc.)

JUnit



Test class name: junit.samples.AllTests

☑ Reload classes every run

Runs:	Errors:	Failures:
90/90	0	0

All JUnit Tests
- junit.samples.VectorTest
  - ✓ testCapacity
  - ✓ testClone
  - ✓ testContains

Finished: 2.172 seconds

Practice 7

Customer tests

- While programmers are programming:
  - Customer writes an acceptance test for each story
- Ideally, a tester is available to automate the test

View an existing reservation

Present the customer with a list of reservations he's made.

Front

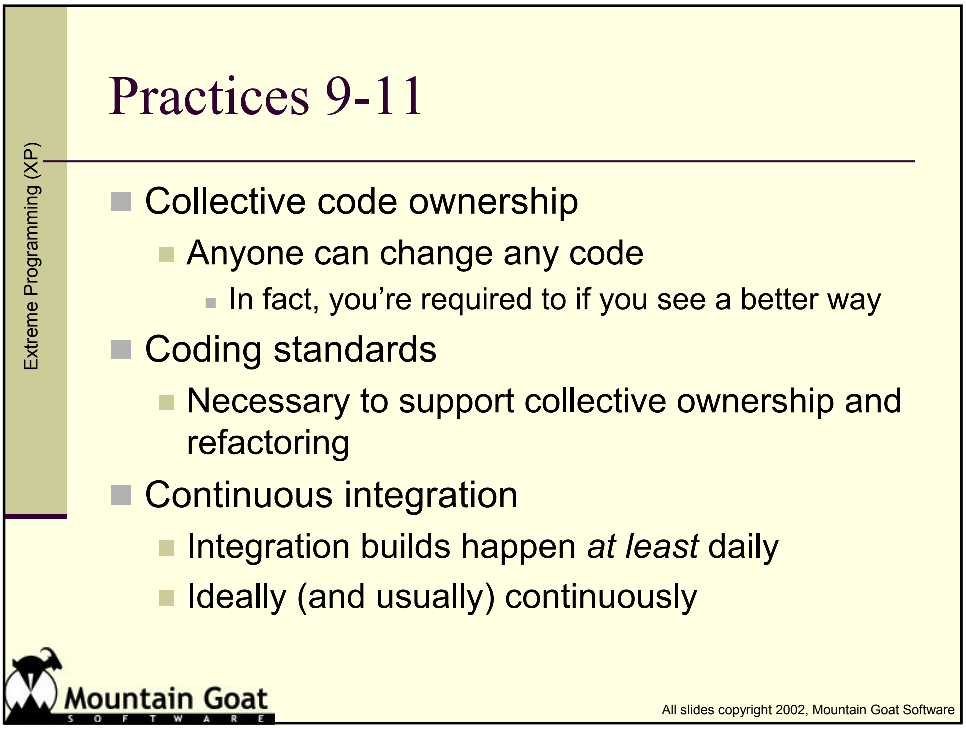
1) Test with a customer with one reservation in the past and two in the future.
2) Test with a customer with no reservations.

Back

Practice 8

Refactoring (Design Improvement)

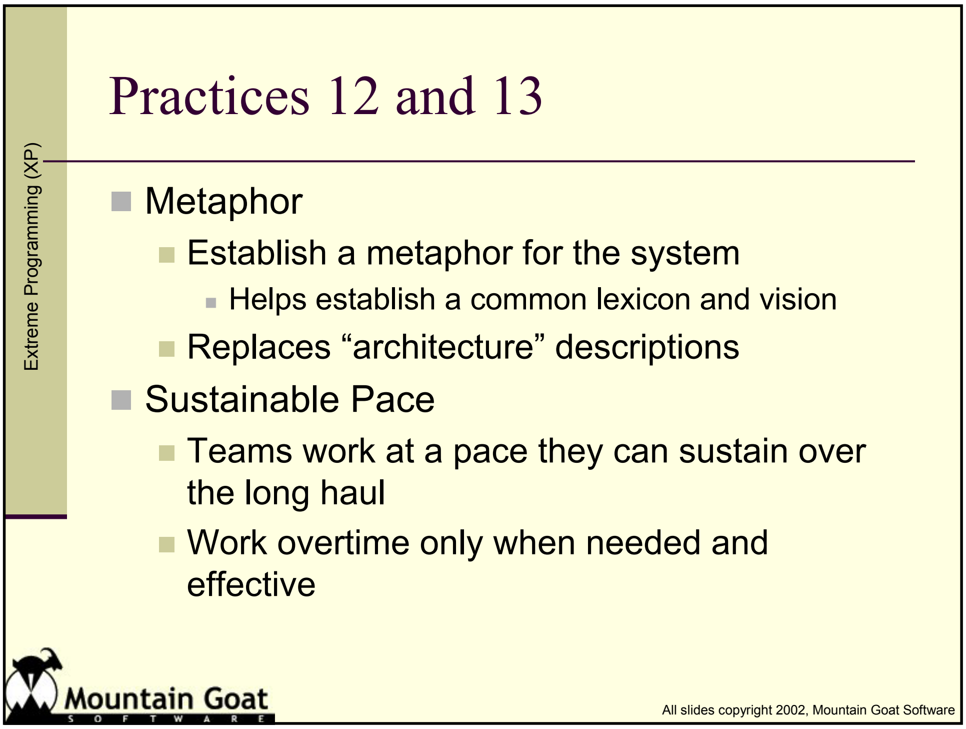
- Refactoring
  - Simplifying or improving the code without changing its behavior
- Automated unit tests ensure nothing breaks
  - Allows programmers to refactor with confidence
- "Always leave the code cleaner than you found it."



Extreme Programming (XP)

Practices 9-11

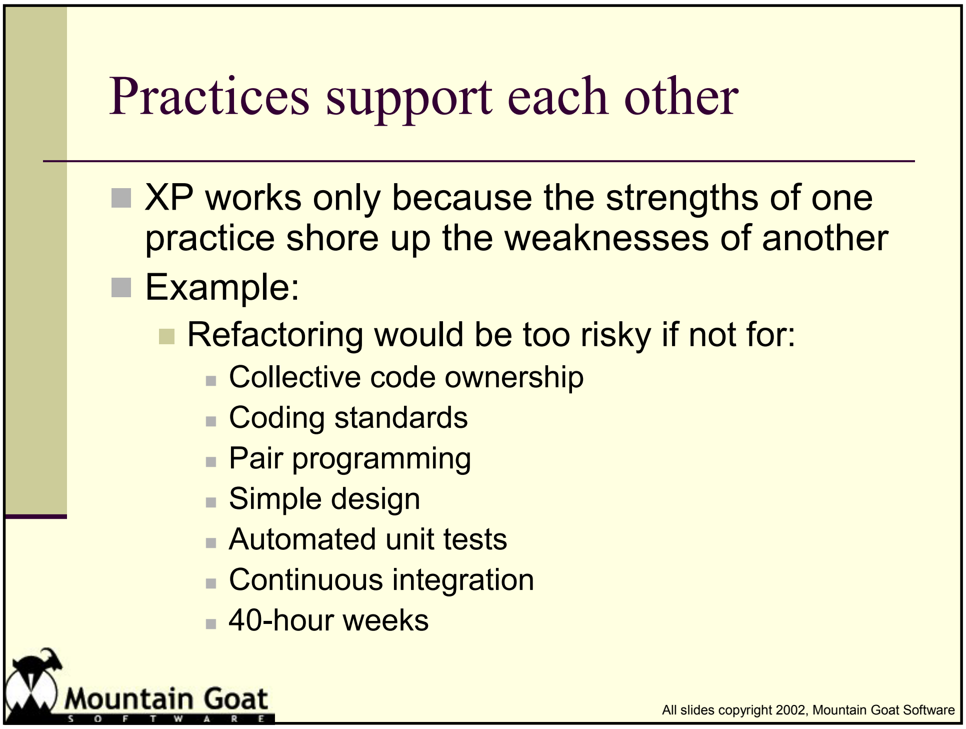
- Collective code ownership
  - Anyone can change any code
    - In fact, you’re required to if you see a better way
- Coding standards
  - Necessary to support collective ownership and refactoring
- Continuous integration
  - Integration builds happen *at least* daily
  - Ideally (and usually) continuously



Extreme Programming (XP)

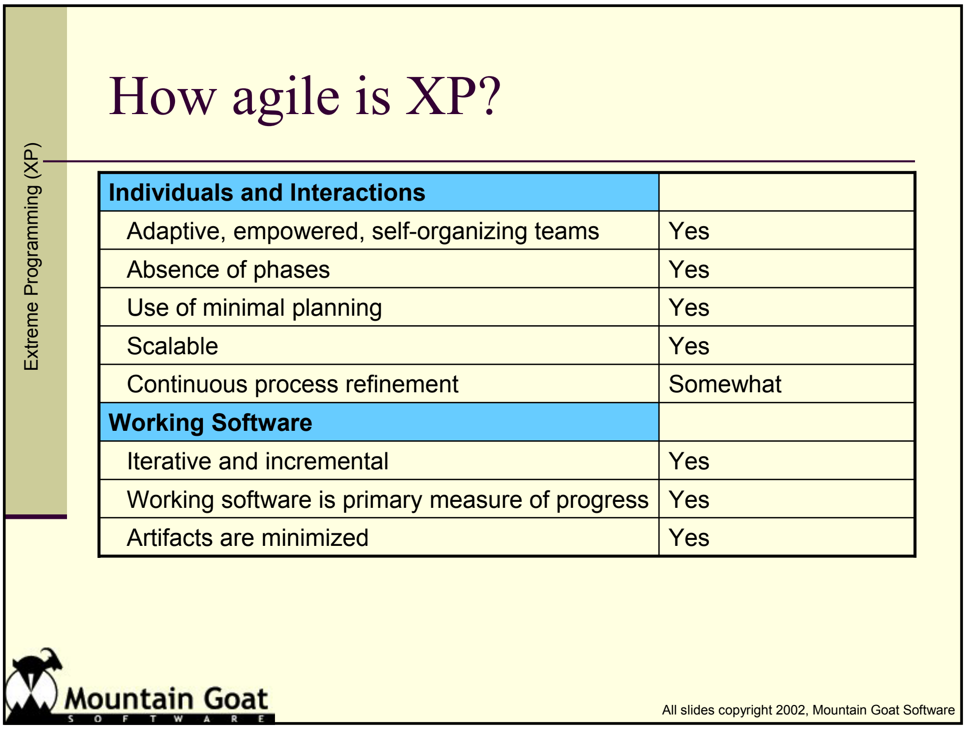
Practices 12 and 13

- Metaphor
  - Establish a metaphor for the system
    - Helps establish a common lexicon and vision
  - Replaces “architecture” descriptions
- Sustainable Pace
  - Teams work at a pace they can sustain over the long haul
  - Work overtime only when needed and effective



Practices support each other

- XP works only because the strengths of one practice shore up the weaknesses of another
- Example:
  - Refactoring would be too risky if not for:
    - Collective code ownership
    - Coding standards
    - Pair programming
    - Simple design
    - Automated unit tests
    - Continuous integration
    - 40-hour weeks



Extreme Programming (XP)

How agile is XP?

Individuals and Interactions	
Adaptive, empowered, self-organizing teams	Yes
Absence of phases	Yes
Use of minimal planning	Yes
Scalable	Yes
Continuous process refinement	Somewhat
Working Software	
Iterative and incremental	Yes
Working software is primary measure of progress	Yes
Artifacts are minimized	Yes

Extreme Programming (XP)

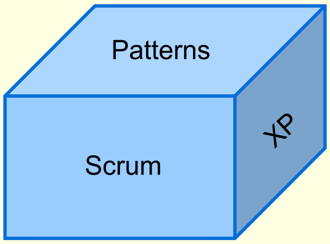
How agile is XP?

Customer Collaboration	
Customer involvement throughout	Yes
Adaptive, empirical customer relationship	Yes
Responding to Change	
Emergent requirements	Yes
Frequent inspection	Yes

Mountain Goat SOFTWARE

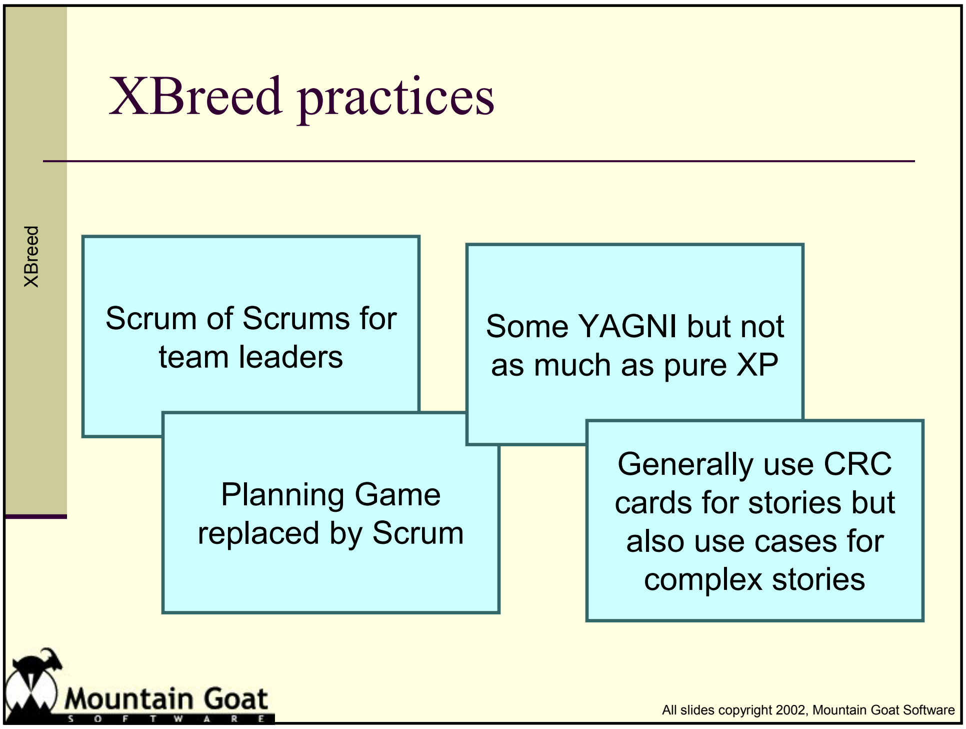
XBreed

XBreed



- Mike Beedle's combination of Scrum, XP and Patterns

Mountain Goat SOFTWARE



XBreed

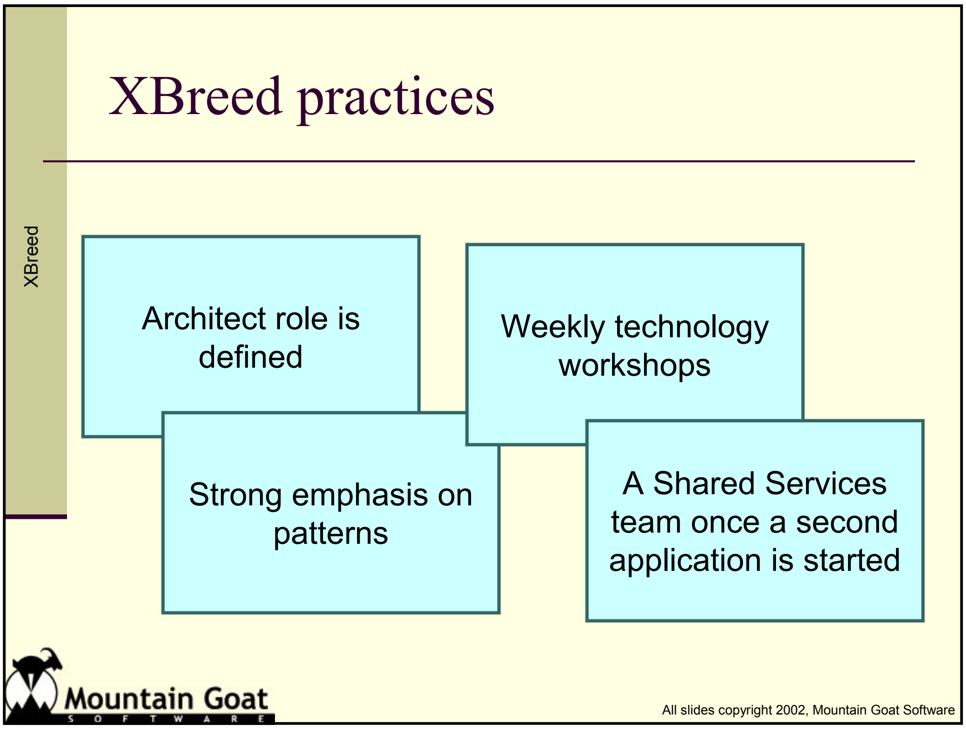
XBreed practices

Scrum of Scrums for team leaders

Planning Game replaced by Scrum

Some YAGNI but not as much as pure XP

Generally use CRC cards for stories but also use cases for complex stories



XBreed

XBreed practices

Architect role is defined

Strong emphasis on patterns

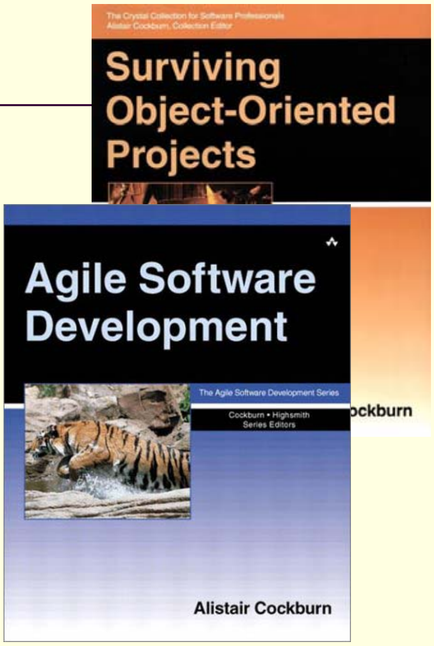
Weekly technology workshops

A Shared Services team once a second application is started

Crystal

Crystal

- Alistair Cockburn
  - Project anthropologist
  - Interviews project teams around the world
- “Software development is a cooperative game of invention and communication.”
  - —Alistair Cockburn


The Crystal Collection for Software Professionals
Alistair Cockburn, Collection Editor
Surviving Object-Oriented Projects
ockburn
Agile Software Development
The Agile Software Development Series
Cockburn • Highsmith Series Editors
Alistair Cockburn

Crystal

Two values

- People- and communication-centric
  - Tools, artifacts, and processes exist only to support the people on the project
- Highly tolerant
  - High or low ceremony
  - High or low discipline

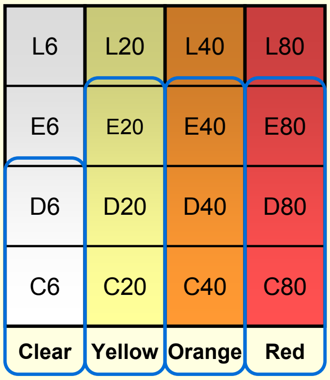
Two rules

- Project must use incremental development
  - Increments cannot exceed four months
- Team must hold pre- and post-increment workshops
  - Reflect on successes and failures of the process
  - Mid-increment workshops encouraged as well

Additional characteristics

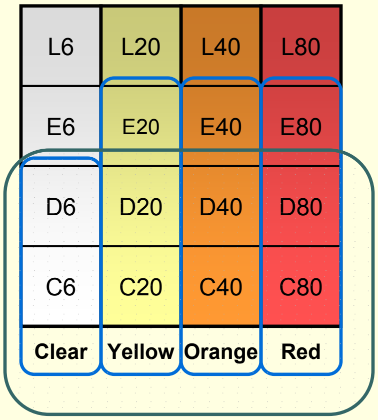
- Only for collocated teams
- Different projects need to be run differently
  - There can never be one process
  - Use heavier methodologies for larger teams
- Fiddling with the process is a Critical Success Factor
- Two most important CSFs:
  - Communication
  - Community

The Crystal family



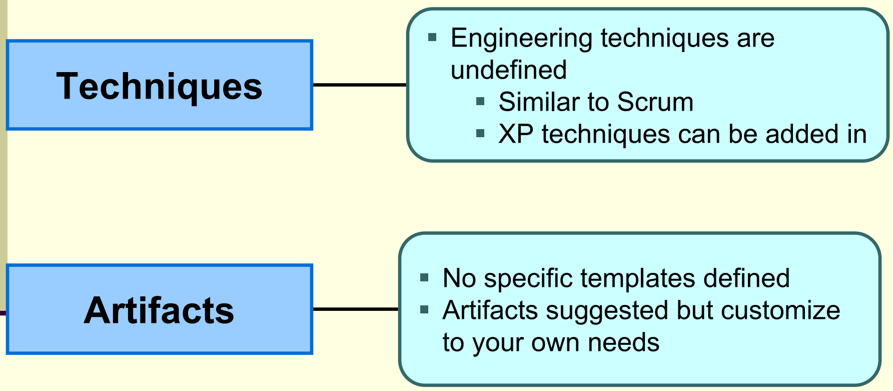
Life (L)	L6	L20	L40	L80
Essential Money (E)	E6	E20	E40	E80
Discretionary Money (D)	D6	D20	D40	D80
Comfort (C)	C6	C20	C40	C80
	Clear	**Yellow**	**Orange**	**Red**

Where Cockburn thinks agile works



Life (L)	L6	L20	L40	L80
Essential Money (E)	E6	E20	E40	E80
Discretionary Money (D)	D6	D20	D40	D80
Comfort (C)	C6	C20	C40	C80
	Clear	**Yellow**	**Orange**	**Red**

Techniques and Artifacts



Techniques

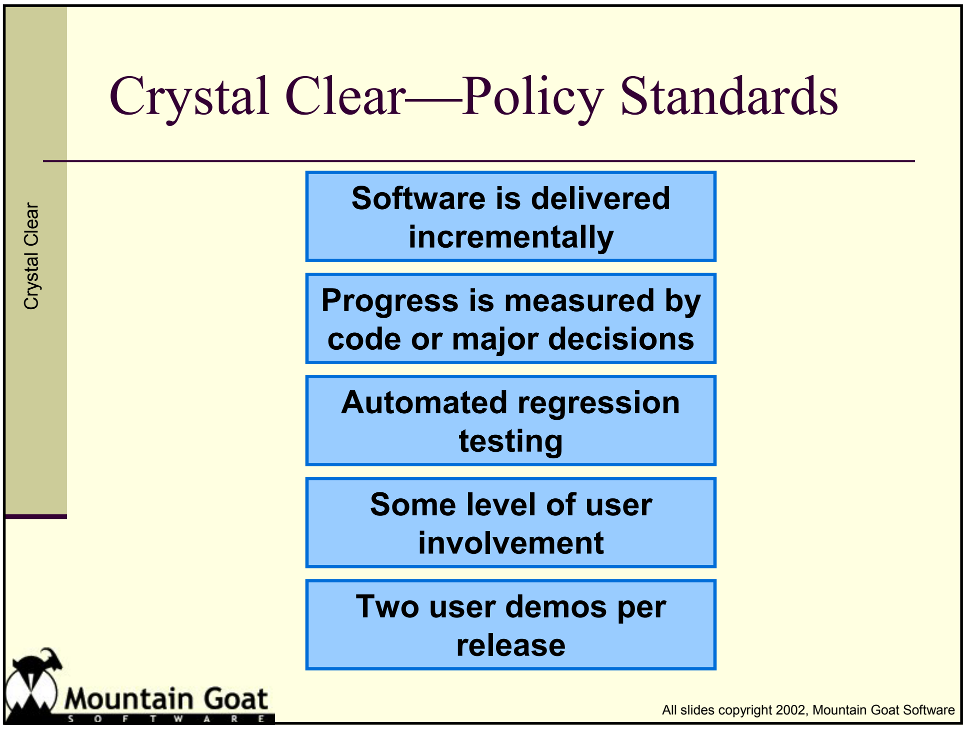
- Engineering techniques are undefined
  - Similar to Scrum
  - XP techniques can be added in

Artifacts

- No specific templates defined
- Artifacts suggested but customize to your own needs

Crystal Clear

- Targeted at D6
  - But works up to E8 or D10
- One team, one office
- Roles
  - Sponsor
  - Senior Designer / Programmer
  - Designer / Programmer
  - User (possibly part-time)

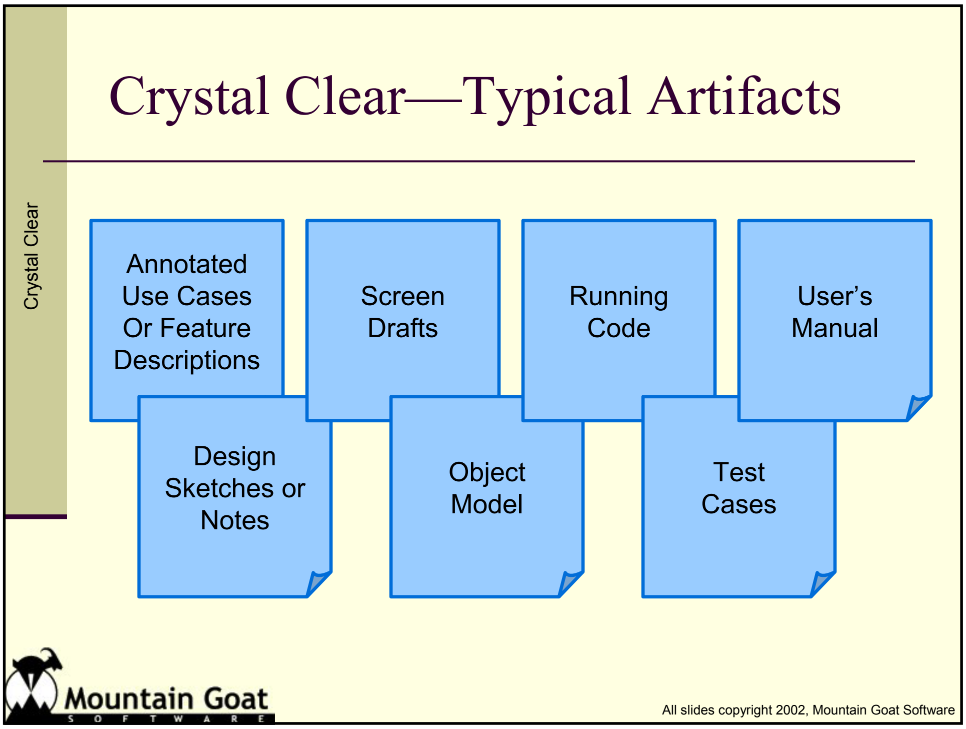


Crystal Clear

Crystal Clear—Policy Standards

- Software is delivered incrementally
- Progress is measured by code or major decisions
- Automated regression testing
- Some level of user involvement
- Two user demos per release

Mountain Goat Software



Crystal Clear

Crystal Clear—Typical Artifacts

- Annotated Use Cases Or Feature Descriptions
- Screen Drafts
- Running Code
- User’s Manual
- Design Sketches or Notes
- Object Model
- Test Cases

Mountain Goat Software

Crystal Orange

Crystal Orange

- 10-40 people
- Project duration of 1-2 years
- Time-to-market is critical
- Project is not life critical
- Desire to communicate with future staff
  - But while minimizing time and cost of doing so

Mountain Goat SOFTWARE

Crystal Orange

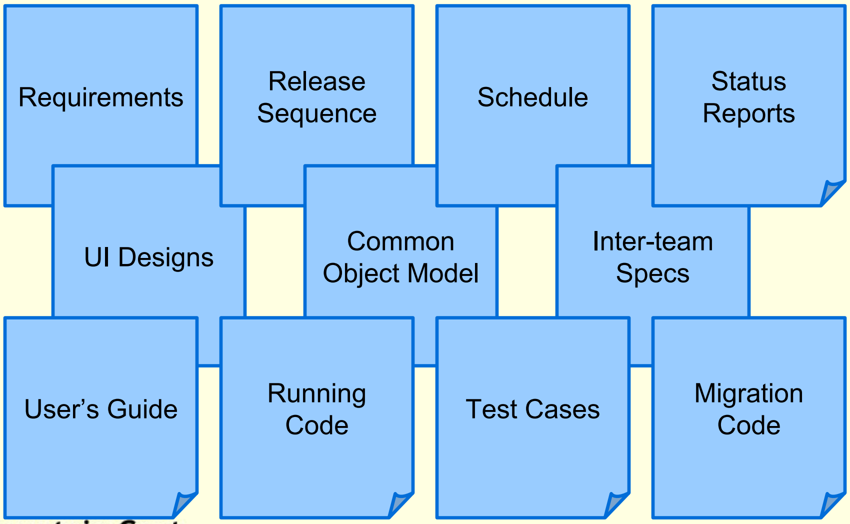
Crystal Orange—Roles

- Sponsor
- Business Expert
- Usage expert
- Technical facilitator
- Business analyst/designer
- Project Manager
- Architect
- Tester
- Design mentor
- Lead designer /programmer
- Other designers / programmers
- UI designer
- Reuse point
- Writer

Mountain Goat SOFTWARE

Crystal Orange

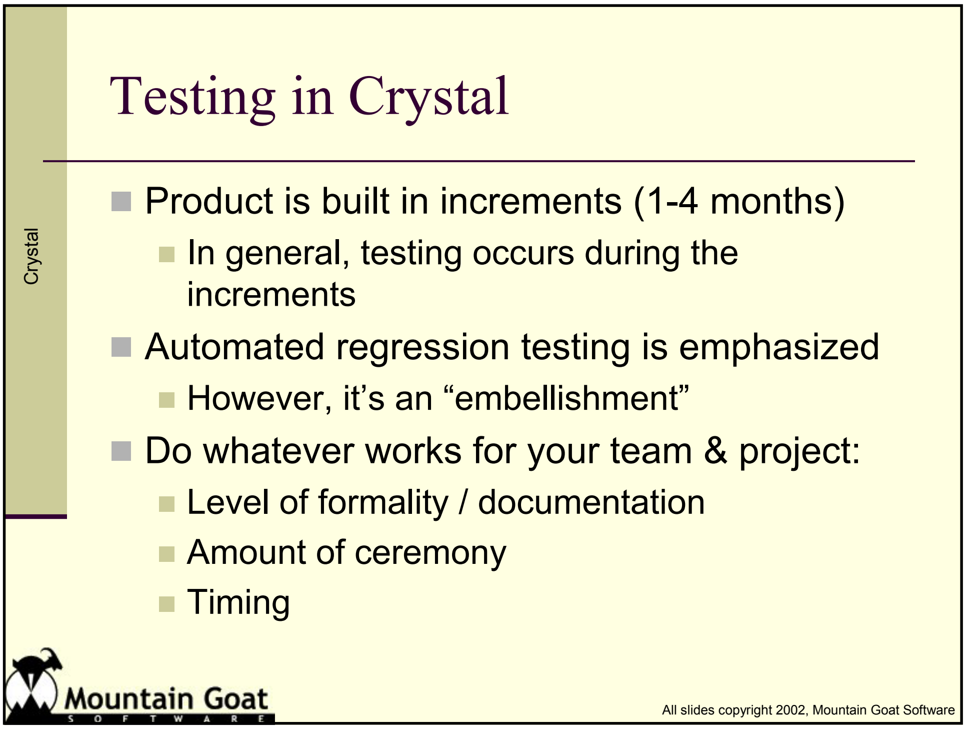
Crystal Orange—Typical Artifacts



Crystal

So how do I “do Crystal?”

- Hold a two-day workshop to develop policy statements for your project
- Start with one of the documented variants
  - Crystal Clear, Orange and Orange-Web
- Do 2-4 month increments
- Constantly adjust process to be “barely sufficient”
- Reflect at middle and end of each increment

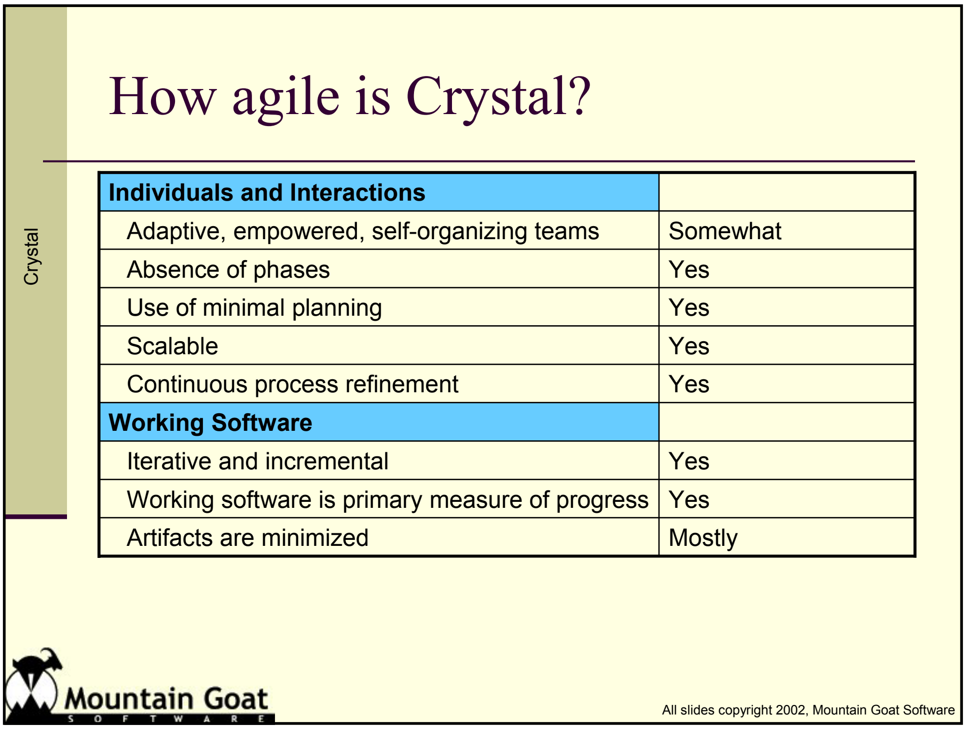


Crystal

Testing in Crystal

- Product is built in increments (1-4 months)
  - In general, testing occurs during the increments
- Automated regression testing is emphasized
  - However, it's an "embellishment"
- Do whatever works for your team & project:
  - Level of formality / documentation
  - Amount of ceremony
  - Timing

Mountain Goat SOFTWARE



Crystal

How agile is Crystal?

Individuals and Interactions	
Adaptive, empowered, self-organizing teams	Somewhat
Absence of phases	Yes
Use of minimal planning	Yes
Scalable	Yes
Continuous process refinement	Yes
Working Software	
Iterative and incremental	Yes
Working software is primary measure of progress	Yes
Artifacts are minimized	Mostly

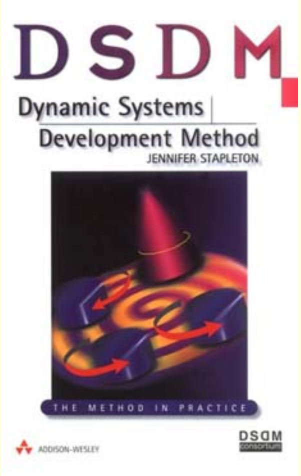
Mountain Goat SOFTWARE

How agile is Crystal?

Customer Collaboration	
Customer involvement throughout	Yes
Adaptive, empirical customer relationship	Yes
Responding to Change	
Emergent requirements	For C and D projects; less so for E and no for L
Frequent inspection	Yes

DSDM

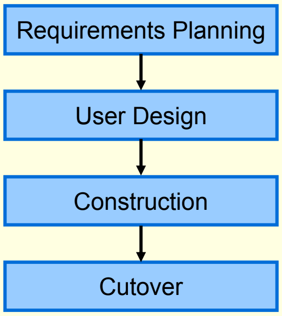
Dynamic
Systems
Development
Method



DSDM

Origins

- James Martin’s *Rapid Application Development* book in 1991
- DSDM Consortium formed in 1994
  - Put out a collection of best practices that hadn’t yet been tried together
  - 220 organizations in Europe



Characteristics

- Highly iterative
- Strong emphasis on prototyping
- Uses timeboxes to control scope
- Strong focus on business value

Current State

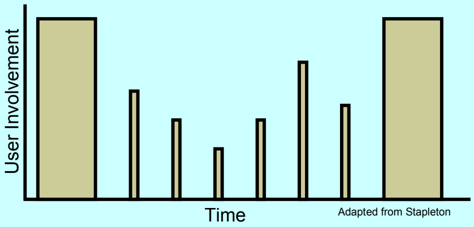
- DSDM 4.1 is currently released
- DSDM 4.2 anticipated November/December
- Members "own" the process
  - Must join the consortium and can then vote

Principles

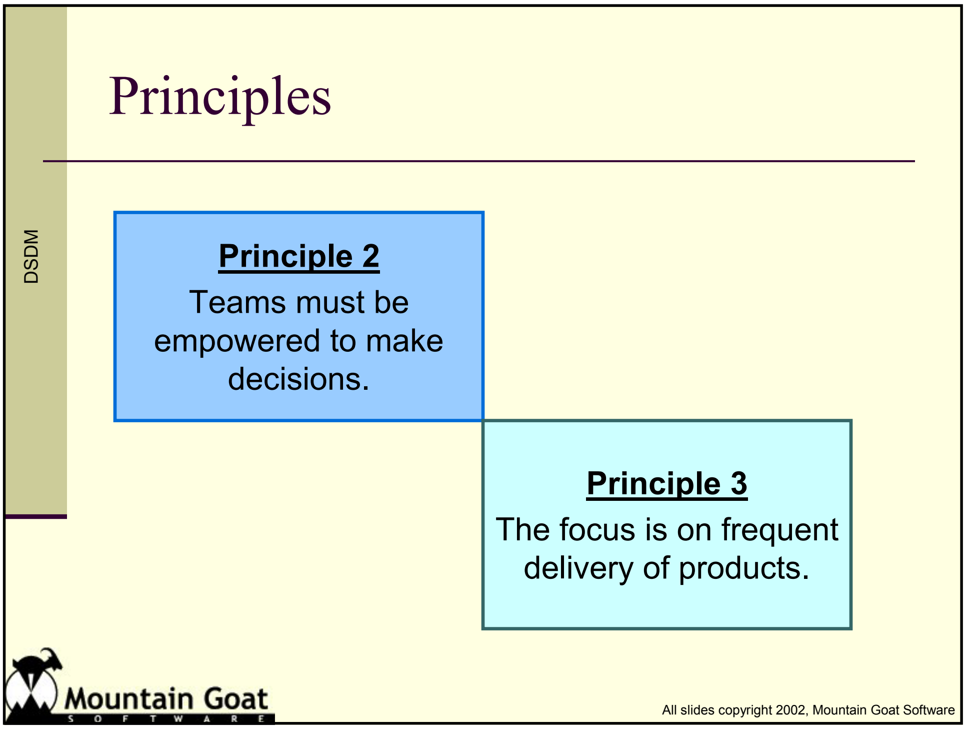
Principle 1

Active user involvement is imperative.

- Avoids the "spiky sofa" curve



Source: *Dynamic Systems Development Method*, Jennifer Stapleton.



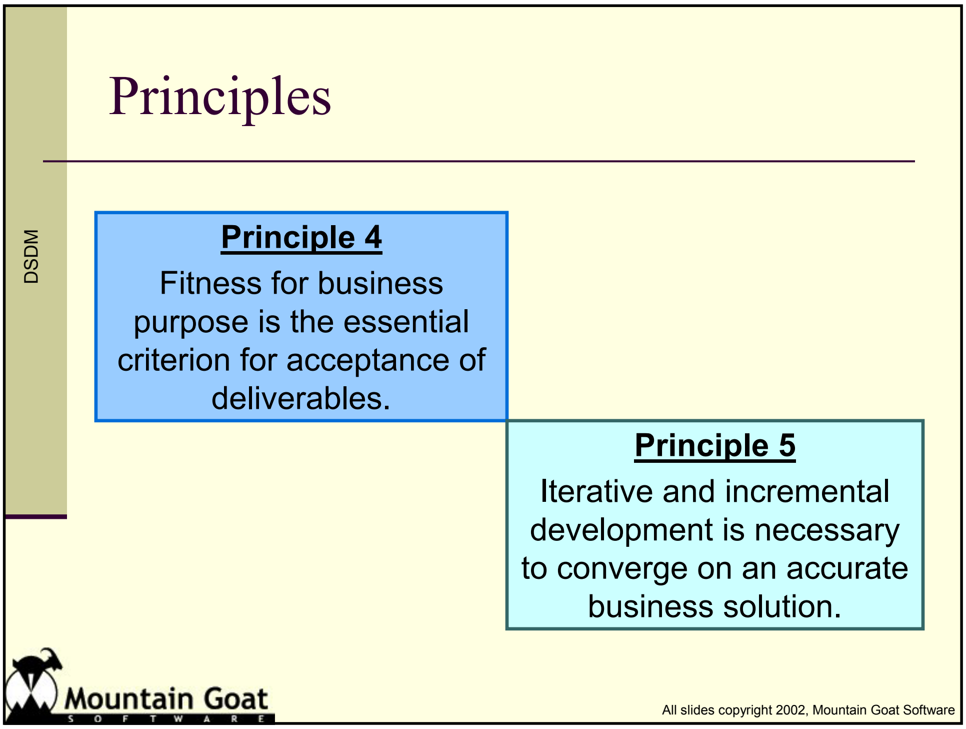
Principles

Principle 2

Teams must be empowered to make decisions.

Principle 3

The focus is on frequent delivery of products.



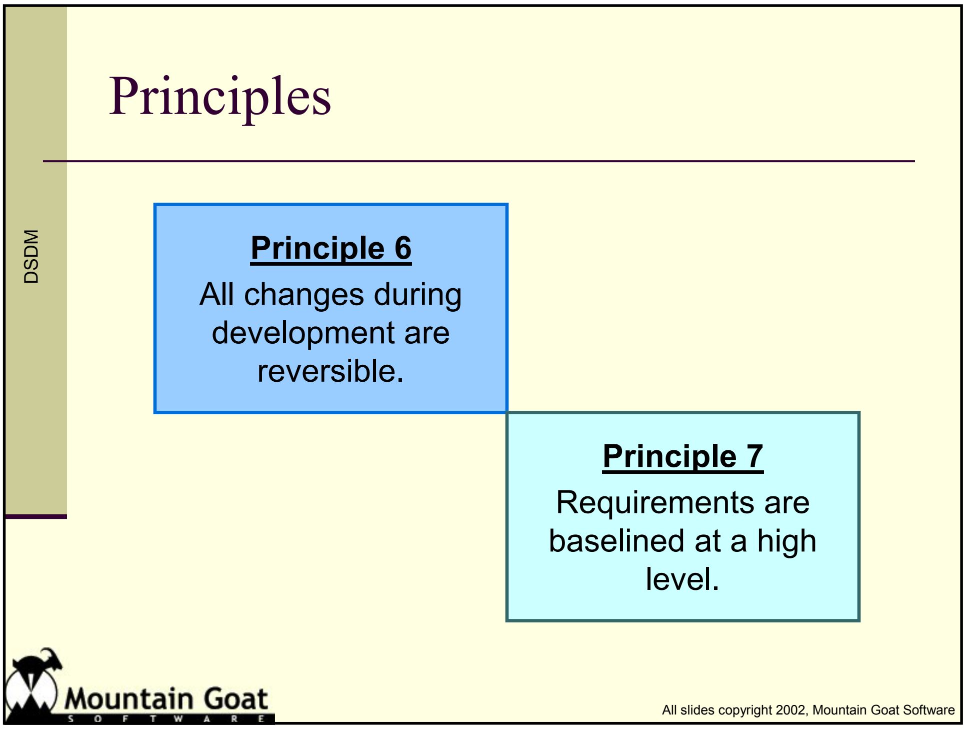
Principles

Principle 4

Fitness for business purpose is the essential criterion for acceptance of deliverables.

Principle 5

Iterative and incremental development is necessary to converge on an accurate business solution.



DSDM

Principles

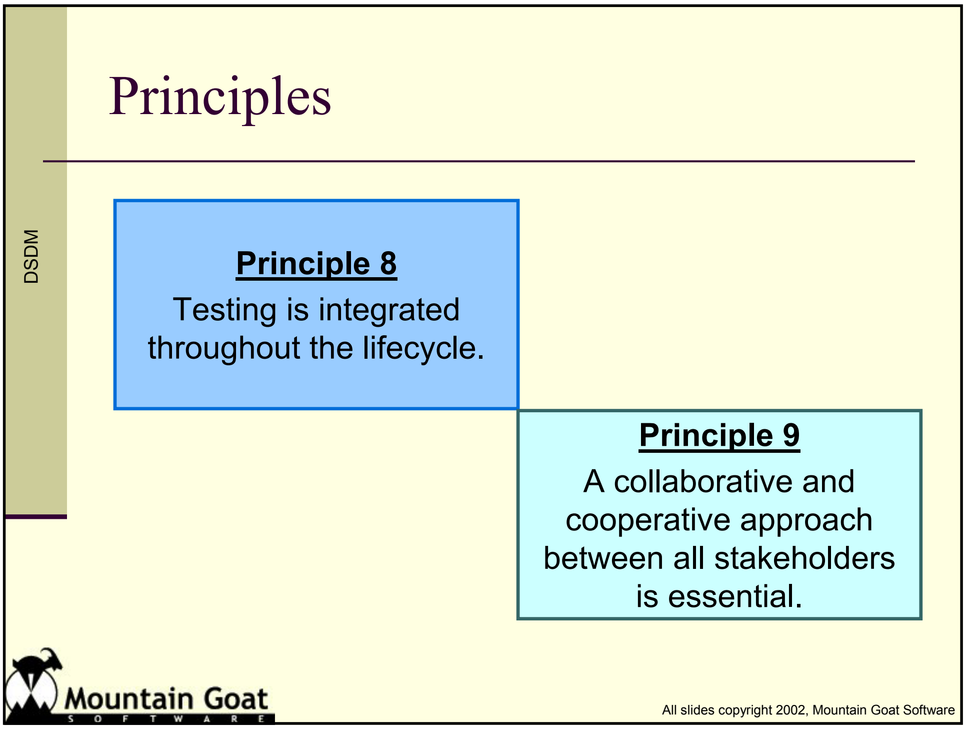
Principle 6

All changes during development are reversible.

Principle 7

Requirements are baselined at a high level.

Mountain Goat SOFTWARE



DSDM

Principles

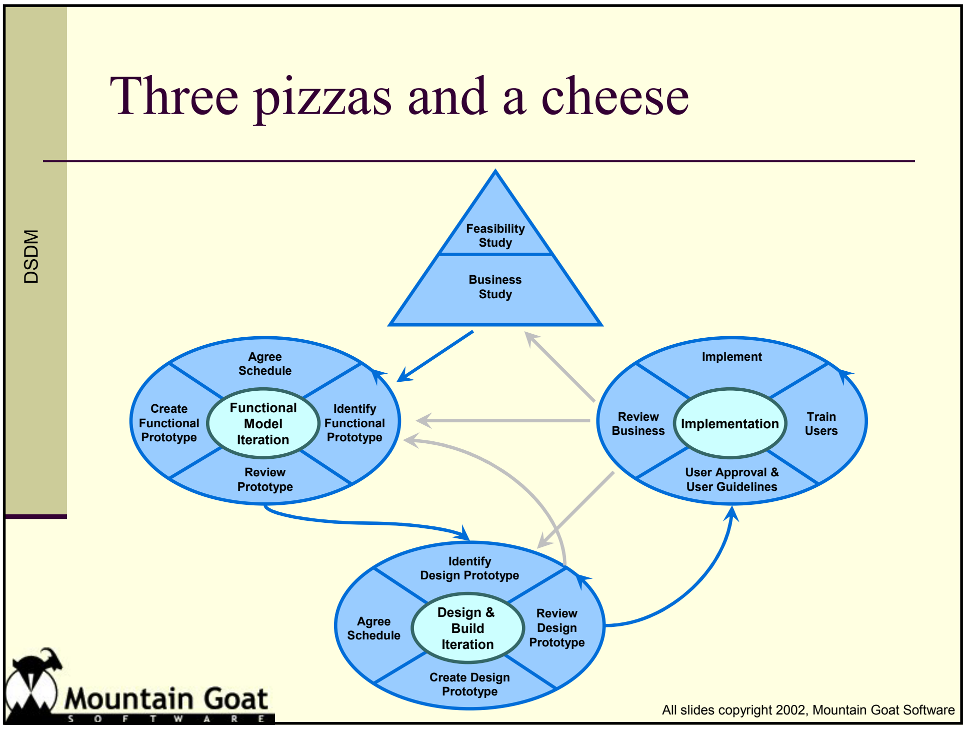
Principle 8

Testing is integrated throughout the lifecycle.

Principle 9

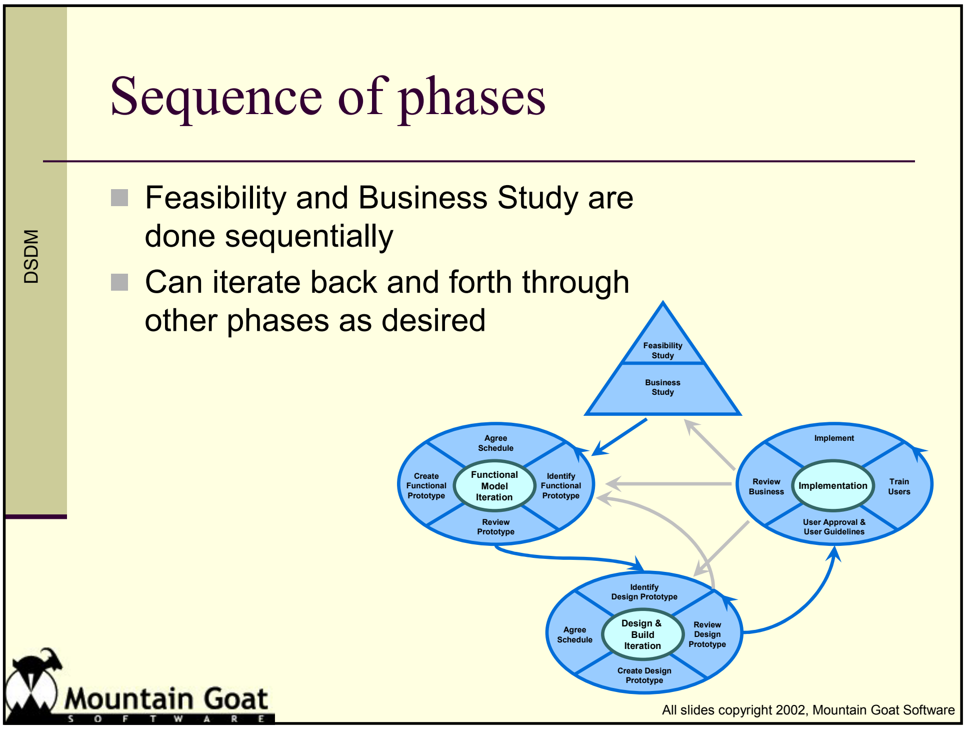
A collaborative and cooperative approach between all stakeholders is essential.

Mountain Goat SOFTWARE



DSDM

Three pizzas and a cheese



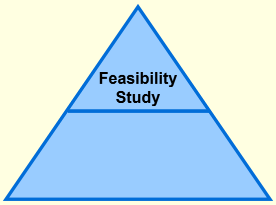
DSDM

Sequence of phases

- Feasibility and Business Study are done sequentially
- Can iterate back and forth through other phases as desired

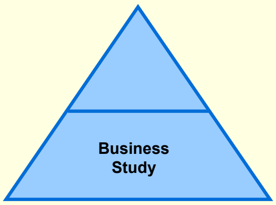
Feasibility Study

- Done to make sure DSDM is right approach for the project
  - Is the project urgent?
  - Is the project UI-intensive?
  - Are specs incomplete?
  - Are the users up for it?
- Produces
  - Outline Plan for Development
  - Prototype, if needed



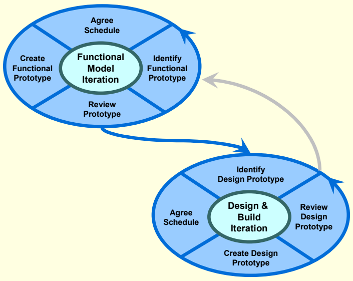
Business Study

- Gain an understanding of business processes
  - ER or class diagrams or ?
- Uses facilitated workshops to gain consensus
- Identify users who will participate throughout project
- *Outline Plan* is created

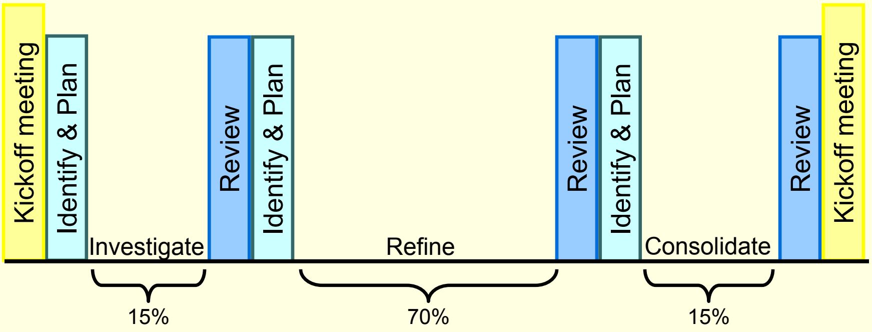


Functional Model & Design and Build Iterations

- Repetitive cycles of:
  - Identify
  - Agree
  - Do
  - Review
- Functional Model
  - Non-production quality code
  - Analysis artifacts
- Design and Build
  - Production quality code



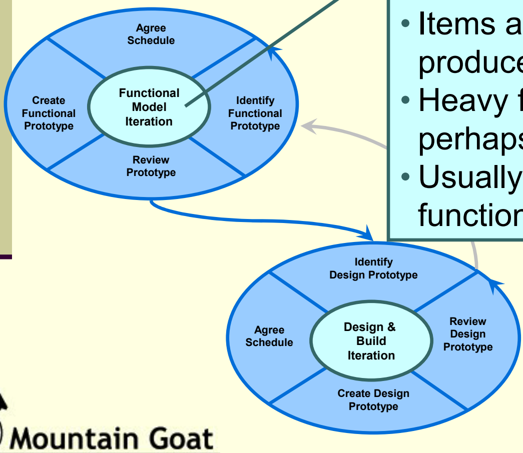
An idealized timebox



Timeboxing requires prioritization

- MoSCoW Rules
  - Must have
    - fundamental to the system
  - Should have
    - important requirement with short-term workaround, would normally be mandatory on a less time-constrained project
  - Could have
    - can be left out of this increment
  - Want to have but won’t have this time
    - Would like to have this increment but can wait for a future increment

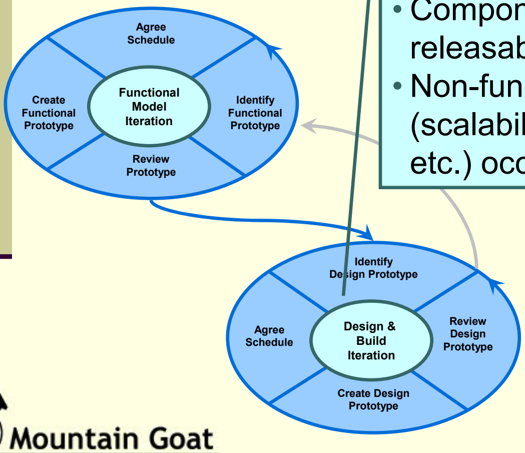
Testing during Functional Model Iterations



- Continuous testing
- Items are tested as they are produced
- Heavy focus on usability testing; perhaps even with an HF group
- Usually little emphasis on non-functional aspects

DSDM

Testing during Design & Build Iterations

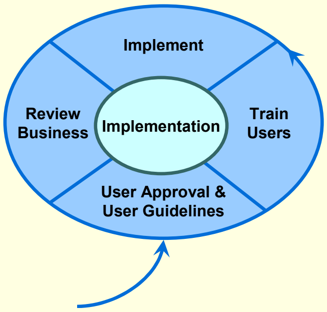


- Testing continues
- Components are driven to releasable quality
- Non-functional testing (scalability, performance, stress, etc.) occurs

DSDM

Implementation Phase

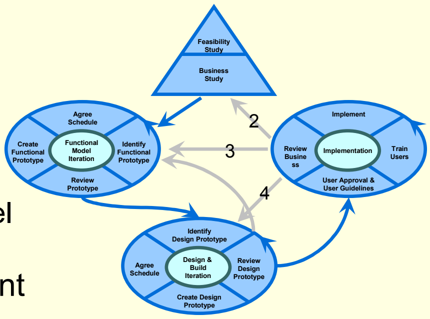
- Deployment of actual system into production environment



DSDM

At end of Implementation Phase

1. Done
2. New business needs are discovered
   - Back to Business Study
3. Low priority work was skipped
   - Back to Functional Model Iteration
4. Non-functional requirement only partially fulfilled
   - Back to Design and Build Iteration

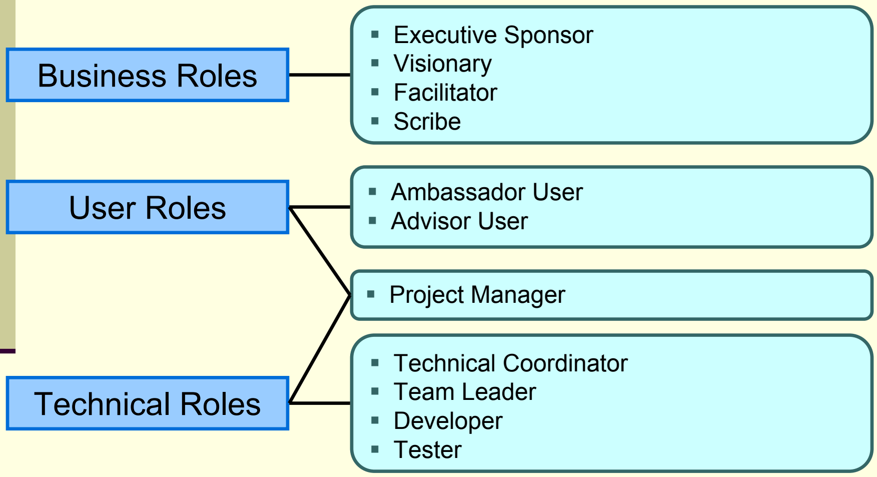

Feasibility Study
Business Study
Agree Schedule
Create Functional Prototype
Functional Model Iteration
Identify Functional Prototype
Review Prototype
Implement
Review Busine ss
Implementation
Train Users
User Approval & User Guidelines
Identify Design Prototype
Agree Schedule
Design & Build Iteration
Review Design Prototype
Create Design Prototype
2
3
4

DSDM

When to use DSDM

- Interactive, UI-intensive
- Clearly defined user group
- Either small projects or projects that can be made small by decomposing them
- Strong time constraints
- Requirements can be prioritized
- Requirements are not clear or change frequently

Roles in DSDM



Business Roles

- Executive Sponsor
- Visionary
- Facilitator
- Scribe

User Roles

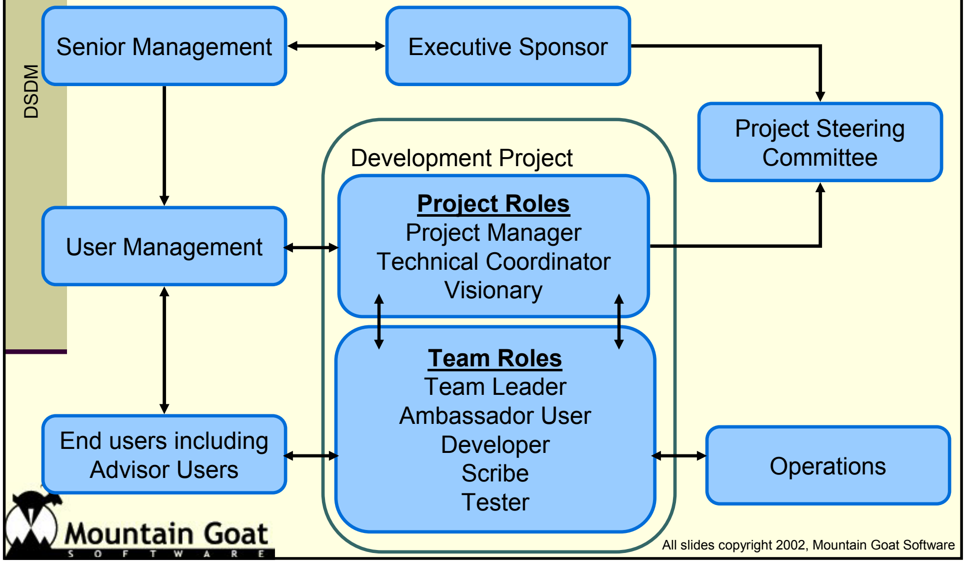
- Ambassador User
- Advisor User

- Project Manager

Technical Roles

- Technical Coordinator
- Team Leader
- Developer
- Tester

Teams



Senior Management

Executive Sponsor

Project Steering Committee

Development Project

Project Roles
Project Manager
Technical Coordinator
Visionary

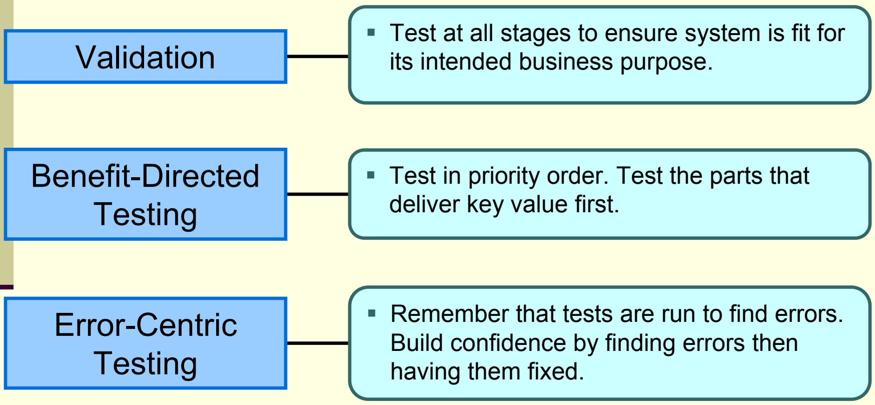
User Management

Team Roles
Team Leader
Ambassador User
Developer
Scribe
Tester

End users including Advisor Users

Operations

Testing principles



Validation

- Test at all stages to ensure system is fit for its intended business purpose.

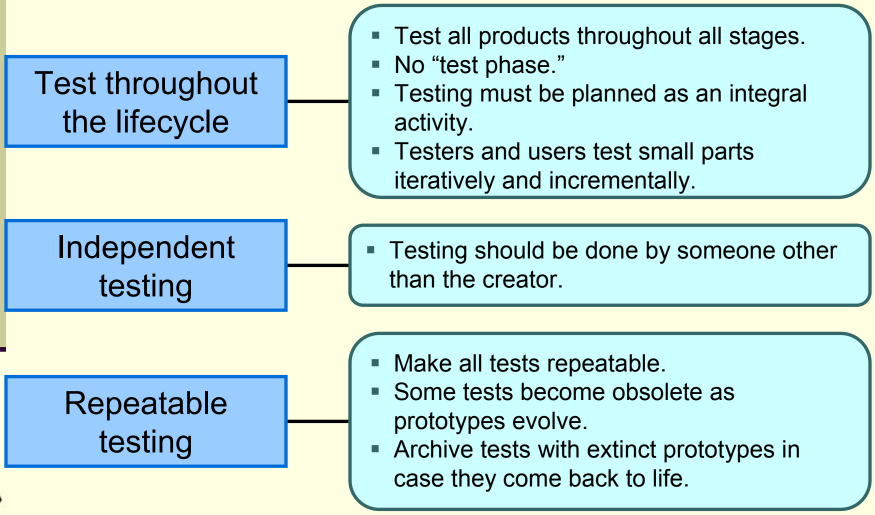
Benefit-Directed Testing

- Test in priority order. Test the parts that deliver key value first.

Error-Centric Testing

- Remember that tests are run to find errors. Build confidence by finding errors then having them fixed.

Testing principles



Test throughout the lifecycle

- Test all products throughout all stages.
- No “test phase.”
- Testing must be planned as an integral activity.
- Testers and users test small parts iteratively and incrementally.

Independent testing

- Testing should be done by someone other than the creator.

Repeatable testing

- Make all tests repeatable.
- Some tests become obsolete as prototypes evolve.
- Archive tests with extinct prototypes in case they come back to life.

DSDM

Testing against business goals

- Testing is against a hierarchy of business goals
  - Not truly against requirements
  - Each requirement supports one or more business goals to greater or lesser degree

Mountain Goat SOFTWARE

DSDM

Risk-based testing

- Typical project constraints force testing to be skipped in some areas
  - Time is critical so apply test time wisely, not necessarily evenly
- RBT says to plan for this upfront by identifying areas you can skip or test lightly
- Identify – Assess – Plan – Reduce Risk
- Done within each timebox so if timebox expires, most important tests have been performed.
- Unit testing performed system-wide

Mountain Goat SOFTWARE

Testing

- Level of testing formality is reduced
  - Normally no step-by-step test cases
  - Instead, a list of test conditions
  - Predicted results not listed, rely on tester's judgment
- A final system test (by technical team and business users) does occur
- Use of static code analyzers and dynamic analysis tools strongly encouraged
  - e.g., Jtest, BoundsChecker, etc.

How agile is DSDM?

Individuals and Interactions	
Adaptive, empowered, self-organizing teams	Partially
Absence of phases	No
Use of minimal planning	Partially
Scalable	Somewhat
Continuous process refinement	Yes
Working Software	
Iterative and incremental	Yes
Working software is primary measure of progress	Yes
Artifacts are minimized	Partially

DSDM

How agile is DSDM?

Customer Collaboration	
Customer involvement throughout	Yes
Adaptive, empirical customer relationship	Yes
Responding to Change	
Emergent requirements	Yes
Frequent inspection	Yes

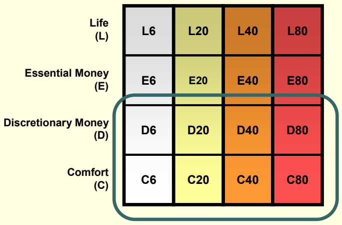
Summary

Summary

- The most agile processes are
  - XP
  - Scrum
  - XBreed
  - Crystal
- Less so
  - DSDM
  - FDD

But….

- “Being agile” is not necessarily the goal
- Delivering working software is the goal
  - Add your own sub-goals about:
    - Speed
    - Quality
    - Schedule predictability
    - Fun
    - Etc.

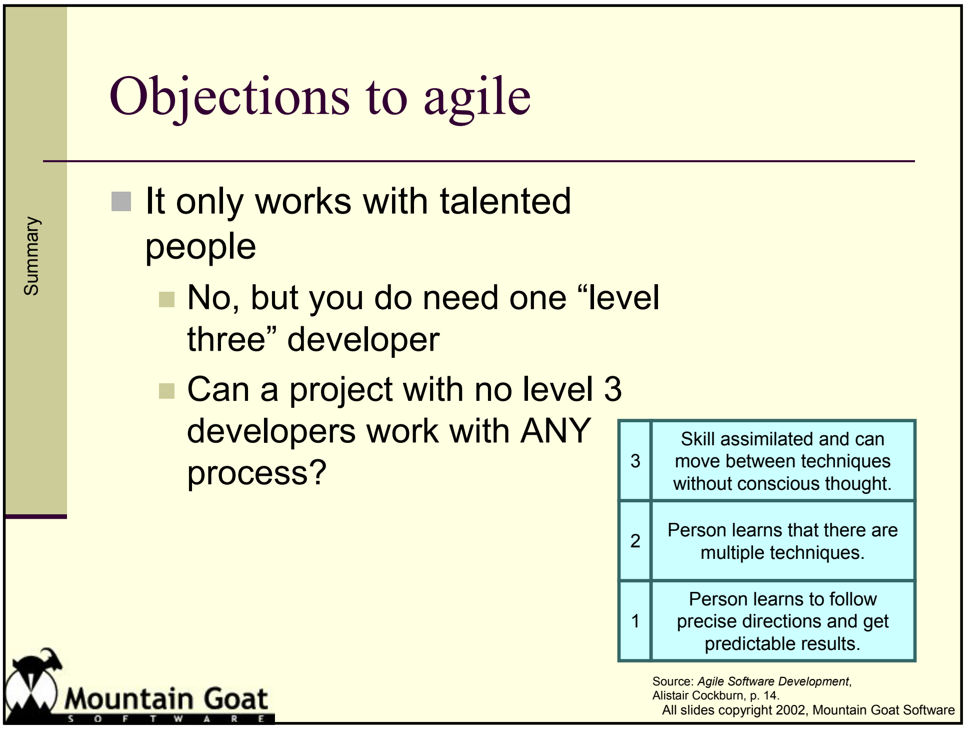


Life (L)	L6	L20	L40	L80
Essential Money (E)	E6	E20	E40	E80
Discretionary Money (D)	D6	D20	D40	D80
Comfort (C)	C6	C20	C40	C80

The Hawthorne Effect

- Western Electric Company, 1927-1932
- Impact of lighting of productivity:
  - With more lighting, productivity went up
  - With less lighting, productivity went up
  - With the same lighting, productivity went up
- “The team gave itself wholeheartedly and spontaneously to cooperation in the experiment.”
- **On important projects, the team owns the process.**

Source: *The Social Problems of an Industrial Civilization*, Mayo, 1945.



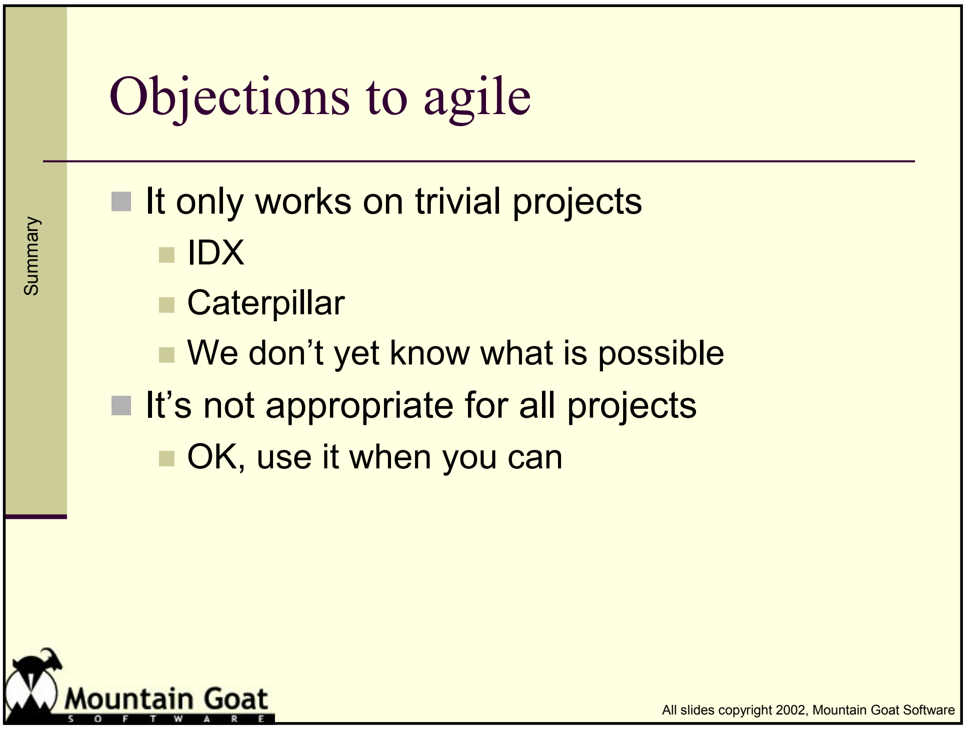
Summary

Objections to agile

- It only works with talented people
  - No, but you do need one “level three” developer
  - Can a project with no level 3 developers work with ANY process?

3	Skill assimilated and can move between techniques without conscious thought.
2	Person learns that there are multiple techniques.
1	Person learns to follow precise directions and get predictable results.

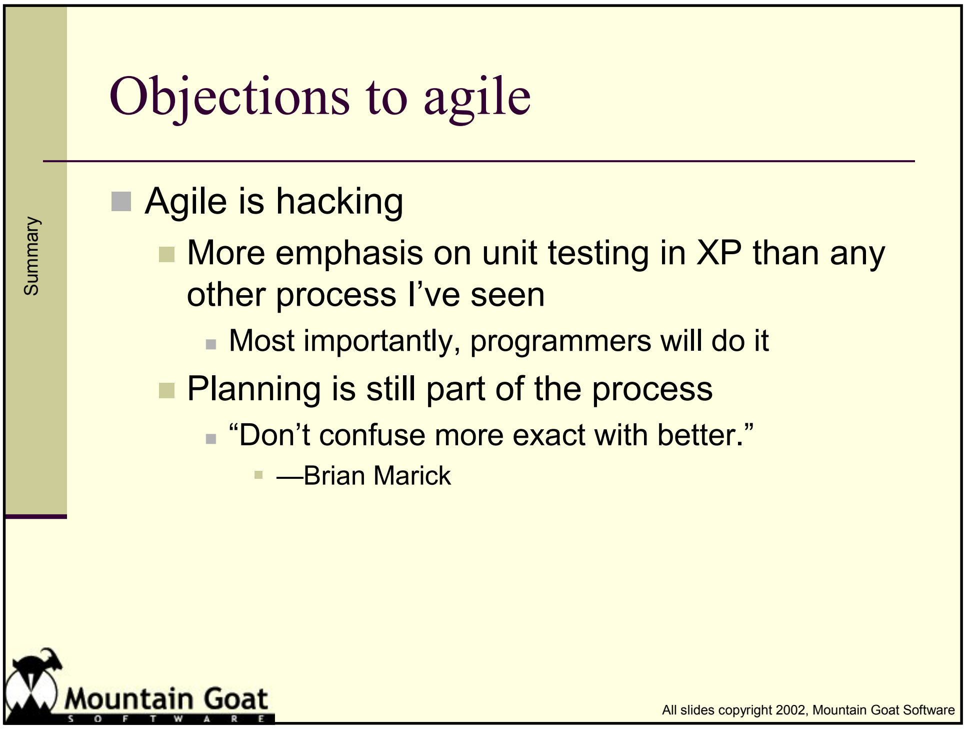
Source: *Agile Software Development*, Alistair Cockburn, p. 14.



Summary

Objections to agile

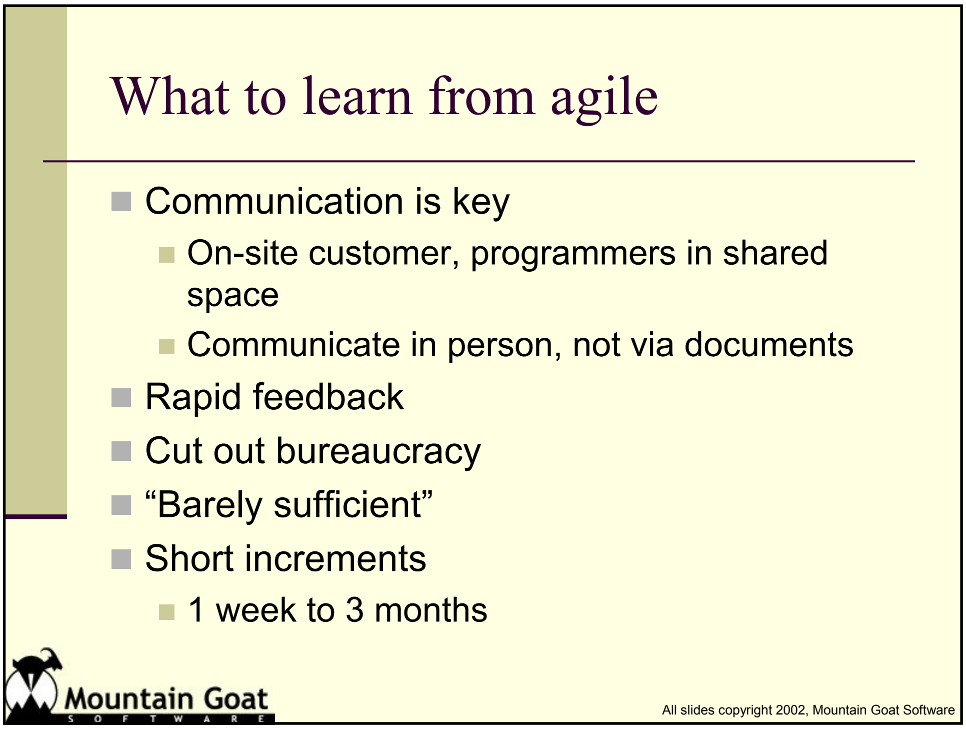
- It only works on trivial projects
  - IDX
  - Caterpillar
  - We don’t yet know what is possible
- It’s not appropriate for all projects
  - OK, use it when you can



Summary

Objections to agile

- Agile is hacking
  - More emphasis on unit testing in XP than any other process I've seen
    - Most importantly, programmers will do it
  - Planning is still part of the process
    - "Don't confuse more exact with better."
      - —Brian Marick



What to learn from agile

- Communication is key
  - On-site customer, programmers in shared space
  - Communicate in person, not via documents
- Rapid feedback
- Cut out bureaucracy
- "Barely sufficient"
- Short increments
  - 1 week to 3 months

What to learn from agile

- Measure progress only by working code
- Customize the process
- Acknowledge the rapidly decreasing precision of plans
- You Aren’t Gonna Need It (YAGNI)
  - Programmers won’t need all the architecture they design
  - Customers don’t need all the features
- Measure success with ROI not KLOC

Further Sources

Where to go next?

- General
  - www.agilealliance.com
  - www.mountaingoatsoftware.com
- Crystal
  - alistair.cockburn.us
  - *Agile Software Development* and *Surviving Object-Oriented Projects* by Alistair Cockburn
- DSDM
  - na.dsdm.org

Further Sources

Where to go next?

- Scrum
  - www.mountaingoatsoftware.com/scrum
  - www.controlchaos.com
  - scrumdevelopment@yahoogroups.com
  - *Agile Software Development with Scrum*
    - Ken Schwaber and Mike Beedle
- Testing
  - agile-testing@yahoogroups.com
  - www.xptester.org
  - www.junit.org

Further Sources

Where to go next?

- XP
  - www.xprogramming.com
  - http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap
  - extremeprogramming@yahoogroups.com
  - xpdenver@yahoogroups.com
  - http://www.extremeprogramming.org/
  - Addison-Wesley's XP Series of books
  - *A Practical Guide to Extreme Programming* by David Astels, Granville Miller, Miroslav Novak
- XBreed
  - www.xbreed.org

My contact information

- Email
  - mike@mountaingoatsoftware.com
- Websites
  - www.mountaingoatsoftware.com
  - www.userstories.com