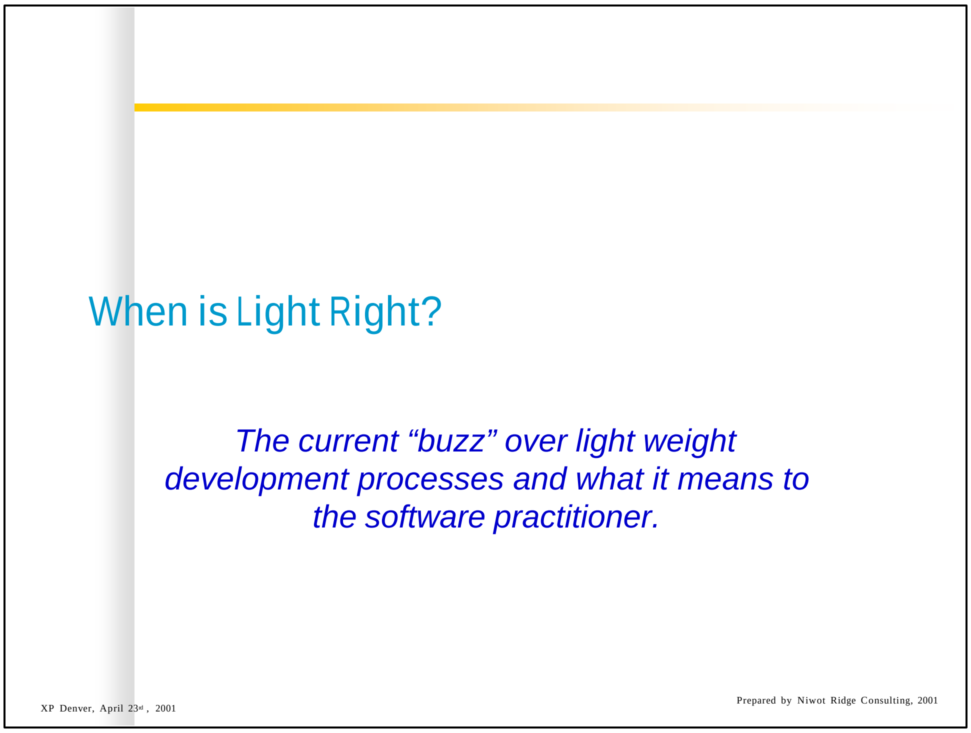


When is Light Right?

The current “buzz” over light weight development processes and what it means to the software practitioner.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The current buzz on the street is that XP is a revolution and is going to be the ***Next New Thing***. DeMarco’s forward to the “Green Book,” claims that XP “is the most important movement in our field today. I predict that it will be as essential to the present generations as the SEI and its CMM was to the last.”

I’m not here to dispute DeMarco or the claims of XP, nor am I qualified to.

But from my experience, there are larger questions to be asked.

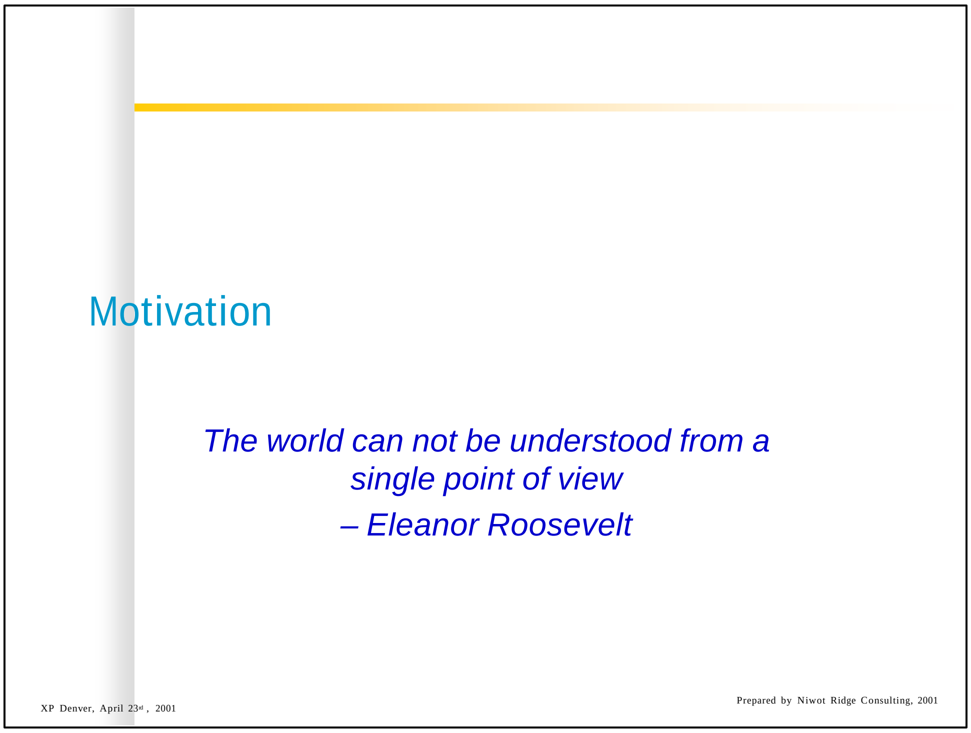
We need to ask the question, “what is the appropriate development methodology for our specific project, the domain, the team, our stakeholders, our managers, and our stock holders?

These are hard questions, which sometimes produce unpleasant answers.

My intention is to show you the XP is embedded in a class of methodologies that are “good engineering practices.” That the majority of the XP practices are derived from processes developed long ago in software development and other disciplines.

That the XP practices can be found in other methods.

That XP fulfills the KPA’s of CMM in the same way other methods do, it does not replace it.



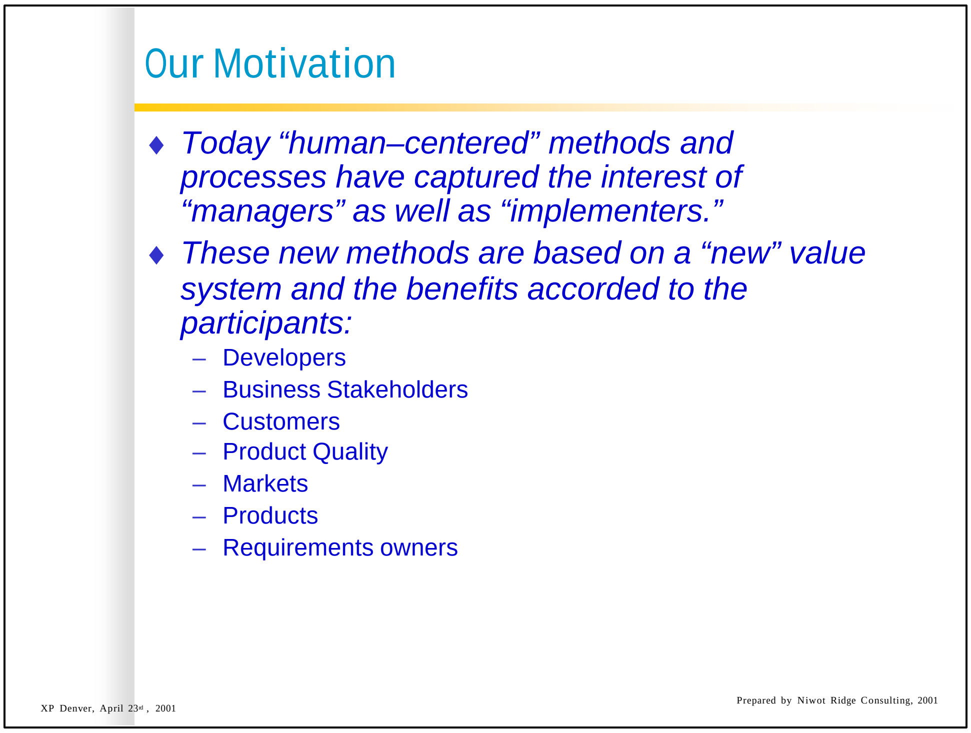
Motivation

The world can not be understood from a single point of view

– Eleanor Roosevelt

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001



Our Motivation

- *Today "human–centered" methods and processes have captured the interest of "managers" as well as "implementers."*
- *These new methods are based on a "new" value system and the benefits accorded to the participants:*
  - Developers
  - Business Stakeholders
  - Customers
  - Product Quality
  - Markets
  - Products
  - Requirements owners

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

DeMarco's comment may be a bit of "hype," but the concept of using Lightweight Processes is a dramatic shift from the previous generation of product and project development methods.

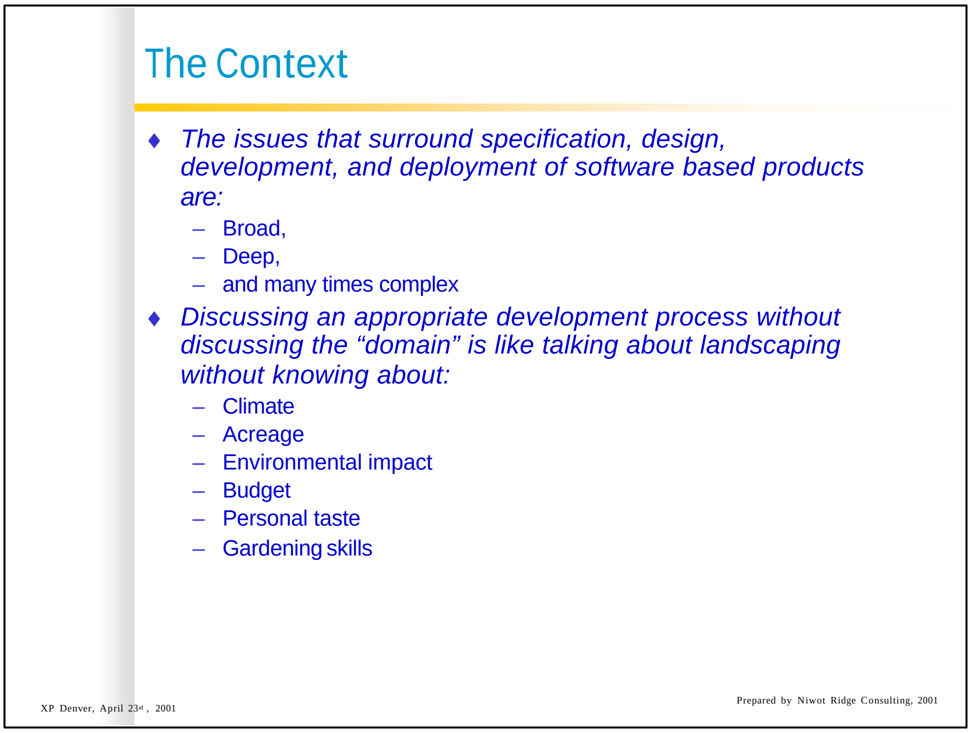
Like most "progressive movements," the principles underlying the "new" activities are obvious once they are applied.

The "human-centered" approach is not new. The identification of the participates is not new. Nothing is really "new" here, just the re-packaging of "good engineering practices," into a new form and putting light on to old problems in a new way.

So why all the "buzz?" The media is to blame a bit. Since I've work primarily in the media business over the past few years, I can say this with some authority. New stories are needed everyday to fill white space around advertising.

This cynical view of the press does not diminish the importance of XP or any lightweight approach, but we need to add some constraints to the situation, things like:

- Where are light weight processes appropriate?
- What practices of a particular process are useable in a specific situation?
- How are the practices of a particular process viewed by the different stakeholders?



The Context

- *The issues that surround specification, design, development, and deployment of software based products are:*
  - Broad,
  - Deep,
  - and many times complex
- *Discussing an appropriate development process without discussing the "domain" is like talking about landscaping without knowing about:*
  - Climate
  - Acreage
  - Environmental impact
  - Budget
  - Personal taste
  - Gardening skills

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Using a "patterns" approach let's first talk about the "Context" of Lightweight Processes. This discussion is sometimes not performed, not because it is unimportant, but because it is assumed to have already taken place.

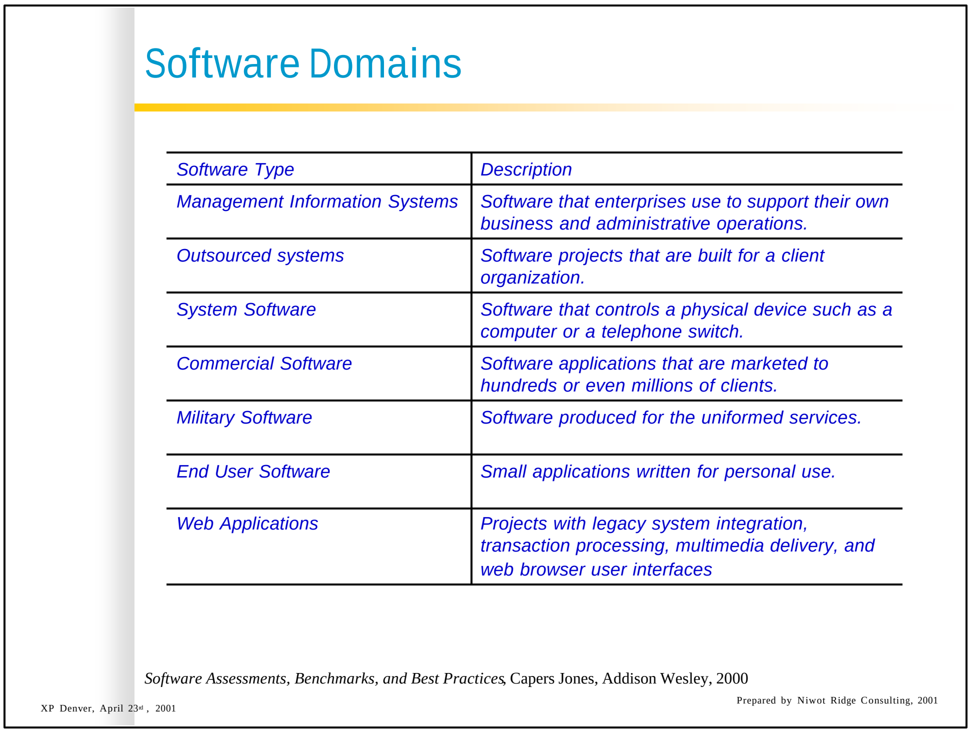
Much like the "patterns" community books, there is a basic assumption that the reader or participant is fully knowledgeable of all that has come before.

The first question is - what environment are light weight processes" appropriate for?

Are they appropriate for all domains, problem spaces, teams, business environments?

Good questions I think.

Not asked as often as maybe they should be.



Software Domains

Software Type	*Description*
Management Information Systems	*Software that enterprises use to support their own business and administrative operations.*
Outsourced systems	*Software projects that are built for a client organization.*
System Software	*Software that controls a physical device such as a computer or a telephone switch.*
Commercial Software	*Software applications that are marketed to hundreds or even millions of clients.*
Military Software	*Software produced for the uniformed services.*
End User Software	*Small applications written for personal use.*
Web Applications	*Projects with legacy system integration, transaction processing, multimedia delivery, and web browser user interfaces*

Software Assessments, Benchmarks, and Best Practices, Capers Jones, Addison Wesley, 2000

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

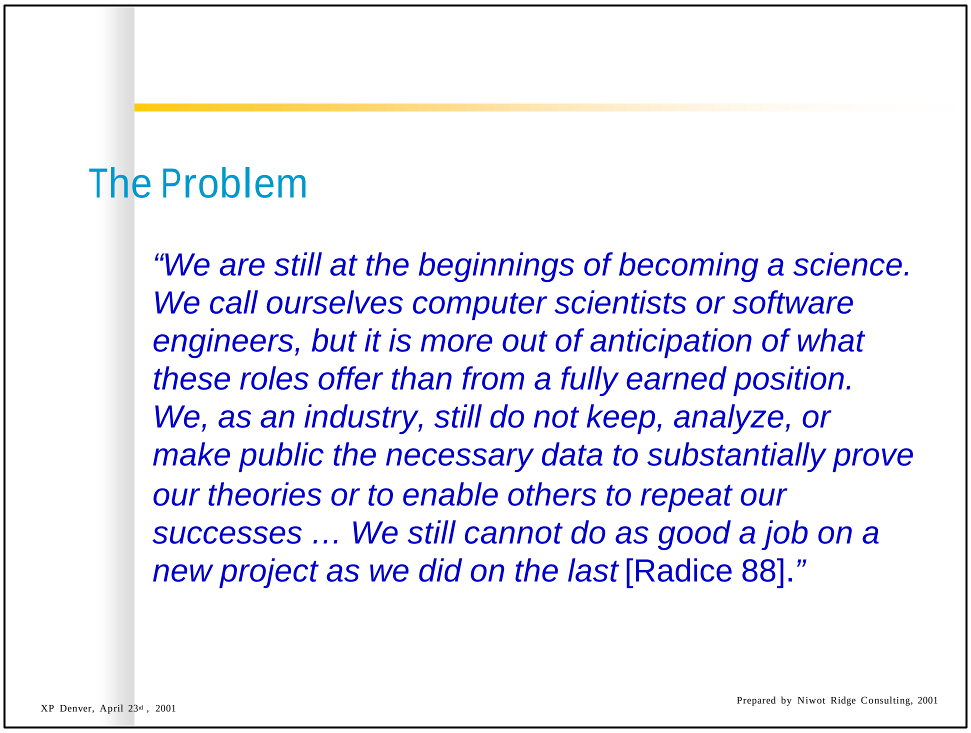
One approach, and there are many, to partitioning software systems is provided by Capers Jones.

In the Jones taxonomy the end application of the software determines the partitions. In this way the details of each system is hidden in the business domain and the attributes of the business domain becomes the important factor.

As you can see there are a variety of different business domains.

The software systems in each domain may be further subdivided, but that’s too much detail for now.

If I were asked to pick out some domains where Light Weight processes excel, I actually would have a hard time. Not because I don’t want to, or I’m opposed to Light Weight processes. But simply that the specific attributes of the stakeholders must also be considered, not just the business domain, or the software application attributes.



The Problem

"We are still at the beginnings of becoming a science. We call ourselves computer scientists or software engineers, but it is more out of anticipation of what these roles offer than from a fully earned position. We, as an industry, still do not keep, analyze, or make public the necessary data to substantially prove our theories or to enable others to repeat our successes … We still cannot do as good a job on a new project as we did on the last [Radice 88]."

XP Denver, April 23rd, 2001

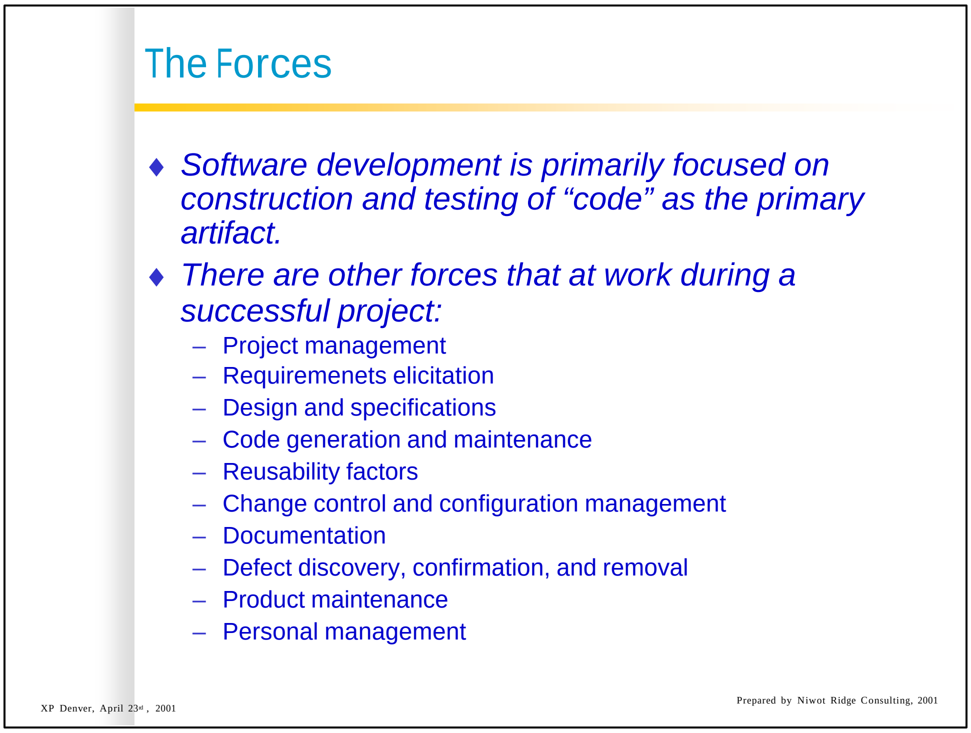
Prepared by Niwot Ridge Consulting, 2001

One core problem we are faced with is that software process "theories," and the study of software processes is still an immature subject.

Much of the information coming from participants is mostly anecdotal. Rather than laboratory data.there have been some experiments in the light weight domain, but they are confined to very controlled groups.

On the other hand processes based on CMM have a head start on data gathering.

Unfortunately much of this data is inappropriate to the problems faced by those interested in Light Weight processes.



The Forces

- *Software development is primarily focused on construction and testing of "code" as the primary artifact.*
- *There are other forces that at work during a successful project:*
  - Project management
  - Requiremenets elicitation
  - Design and specifications
  - Code generation and maintenance
  - Reusability factors
  - Change control and configuration management
  - Documentation
  - Defect discovery, confirmation, and removal
  - Product maintenance
  - Personal management

XP Denver, April 23rd, 2001

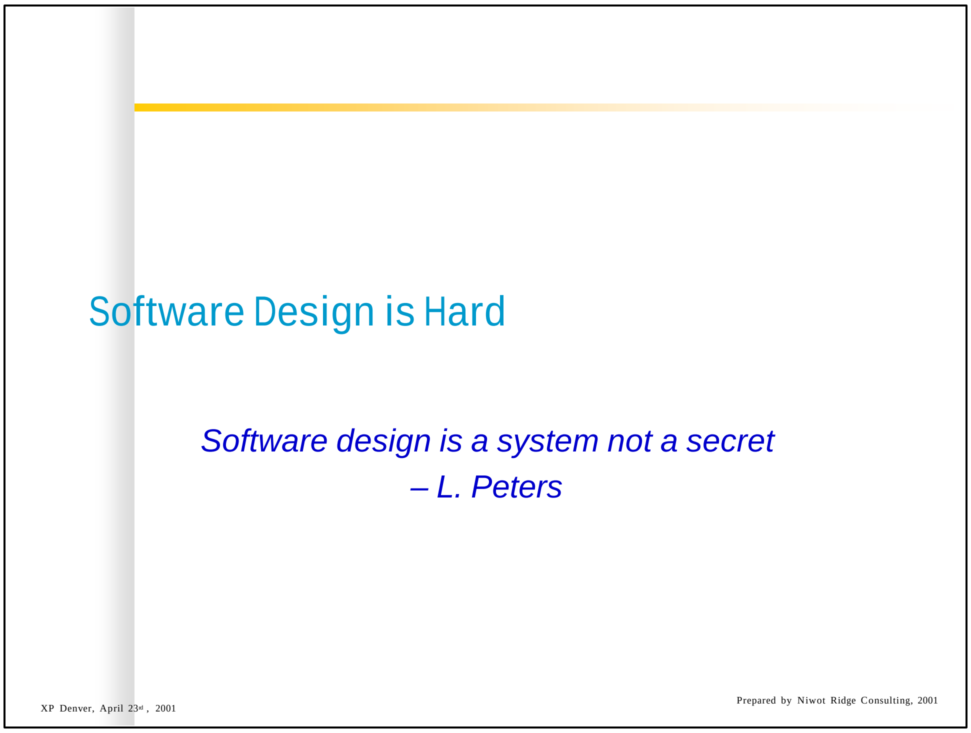
Prepared by Niwot Ridge Consulting, 2001

The primary forces driving Light Weight process, at least in the XP instance, is the development of code. It is argued that code is all there is. That all other artifacts and processes support the development of code. Code is in fact the "end deliverable" of the programming effort.

I'll grant that these arguments have a good point, at least in the domain of "coding." But there is more to design, building and "selling" software than just writing the code.

The previous generation of process analyst have claimed that coding was a small part of the problem. The "smallness" assigned to coding was probably too small, in the same way the current generation emphasizes coding as the large part of development.

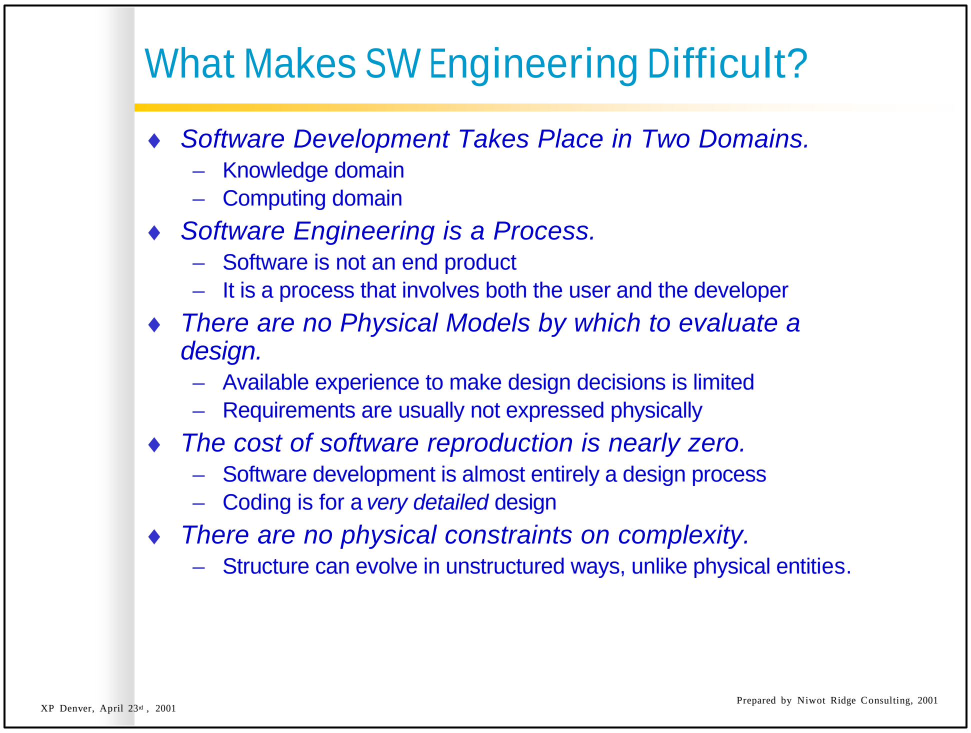
Without addressing the coding issue, here are some other accepted forces driving the development of software.



Software Design is Hard

Software design is a system not a secret
– L. Peters

Niwot Ridge Consulting
4347 Pebble Beach Drive
Niwot, Colorado 80503



What Makes SW Engineering Difficult?

- *Software Development Takes Place in Two Domains.*
  - Knowledge domain
  - Computing domain
- *Software Engineering is a Process.*
  - Software is not an end product
  - It is a process that involves both the user and the developer
- *There are no Physical Models by which to evaluate a design.*
  - Available experience to make design decisions is limited
  - Requirements are usually not expressed physically
- *The cost of software reproduction is nearly zero.*
  - Software development is almost entirely a design process
  - Coding is for a *very detailed* design
- *There are no physical constraints on complexity.*
  - Structure can evolve in unstructured ways, unlike physical entities.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

David Parnas said in 1985 “Software is hard to build because it is inherently and necessarily complex.”

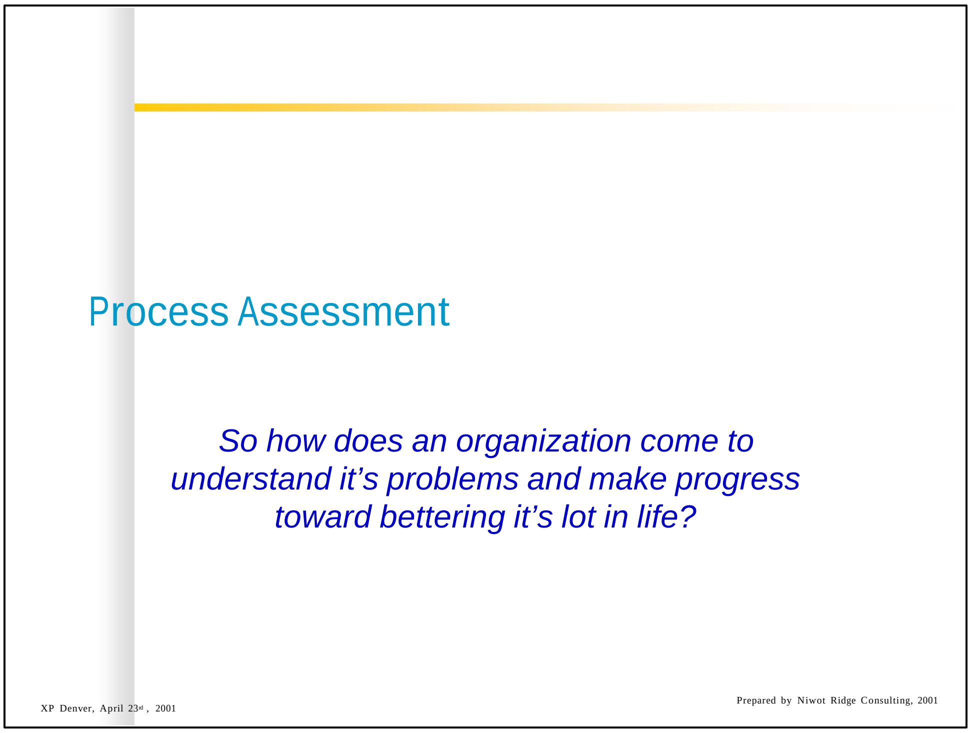
There are some other aspects of software that make it hard.

The concurrent domain issues are encountered every day. A customer wants something but can’t really describe the details of the implementation. The developers hear what the customer wants in terms of code. But the subtle aspects of the translation are lost.

This issue is addressed by one of the powerful aspects of XP – an on site customer. The rub is of course that without an on site customer the problem now has no clever solution, and traditional requirements elicitation processes need to be deployed.

That engineering is a process is well understood in the engineering disciplines. Unfortunately it is not well understood in the sales, marketing, and finance departments.

With no real constraints on complexity, many engineering disciplines have started to adopt heuristics to reduce complexity. Software development is not unique here.



Process Assessment

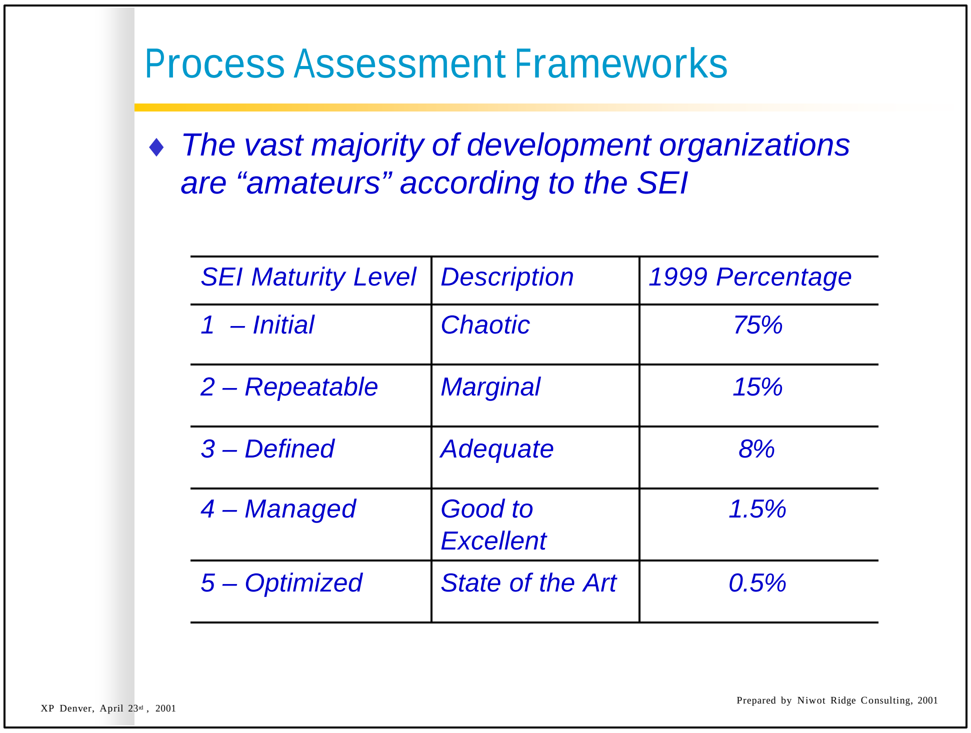
So how does an organization come to understand it's problems and make progress toward bettering it's lot in life?

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

One approach to answering the question, "what process should I use?" can be answered by looking or assessing the current situation and determining what changes need to take place.

Consultants call this a "gap analysis," since we're looking for gaps in the current process to be filled by some better process.



Process Assessment Frameworks

- The vast majority of development organizations are “amateurs” according to the SEI

SEI Maturity Level	Description	1999 Percentage
1 – Initial	Chaotic	75%
2 – Repeatable	Marginal	15%
3 – Defined	Adequate	8%
4 – Managed	Good to Excellent	1.5%
5 – Optimized	State of the Art	0.5%

XP Denver, April 23rd, 2001

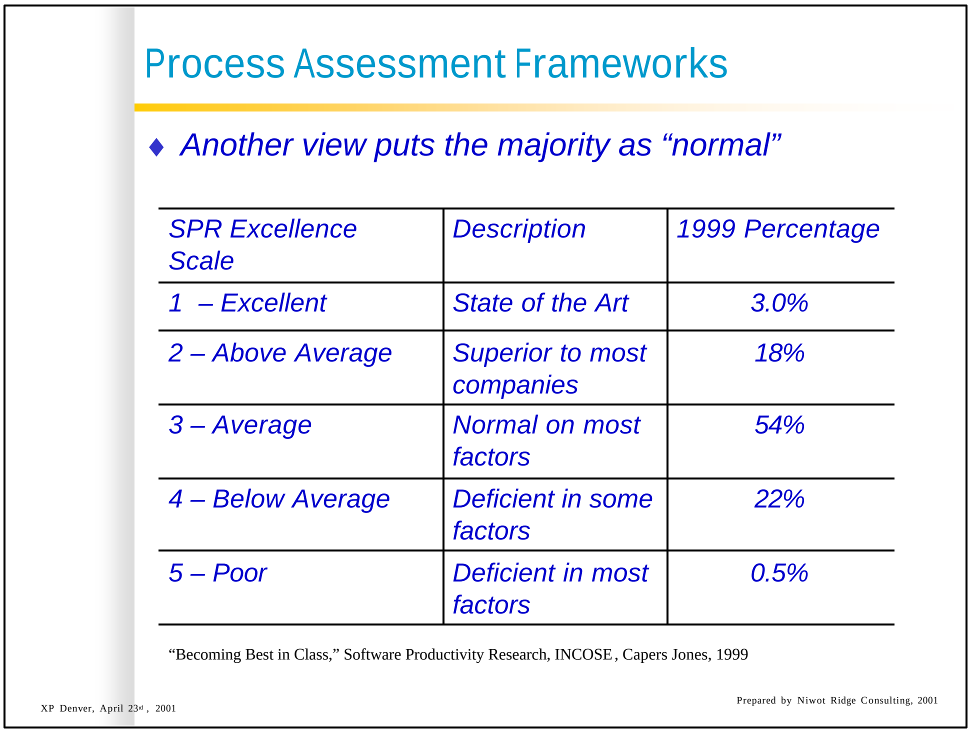
Prepared by Niwot Ridge Consulting, 2001

The “boggie man” of process assessments is the Capability Maturity Model of the Software Engineering Institute.

In one sense this reputation is deserved. The SEI has not come to the defense of CMM in the face of light weight processes, other than PSP and some other derivations of CMM.

Another issue is the over reliance on the artifacts associated with the Key Process Areas. Using the SEI numbers the vast majority of software development organizations are marginal or chaotic.

This does not seem to correspond with the reality of the products being produced by the software community.



Process Assessment Frameworks

- *Another view puts the majority as "normal"*

SPR Excellence Scale	*Description*	*1999 Percentage*
1 – Excellent	*State of the Art*	*3.0%*
2 – Above Average	*Superior to most companies*	*18%*
3 – Average	*Normal on most factors*	*54%*
4 – Below Average	*Deficient in some factors*	*22%*
5 – Poor	*Deficient in most factors*	*0.5%*

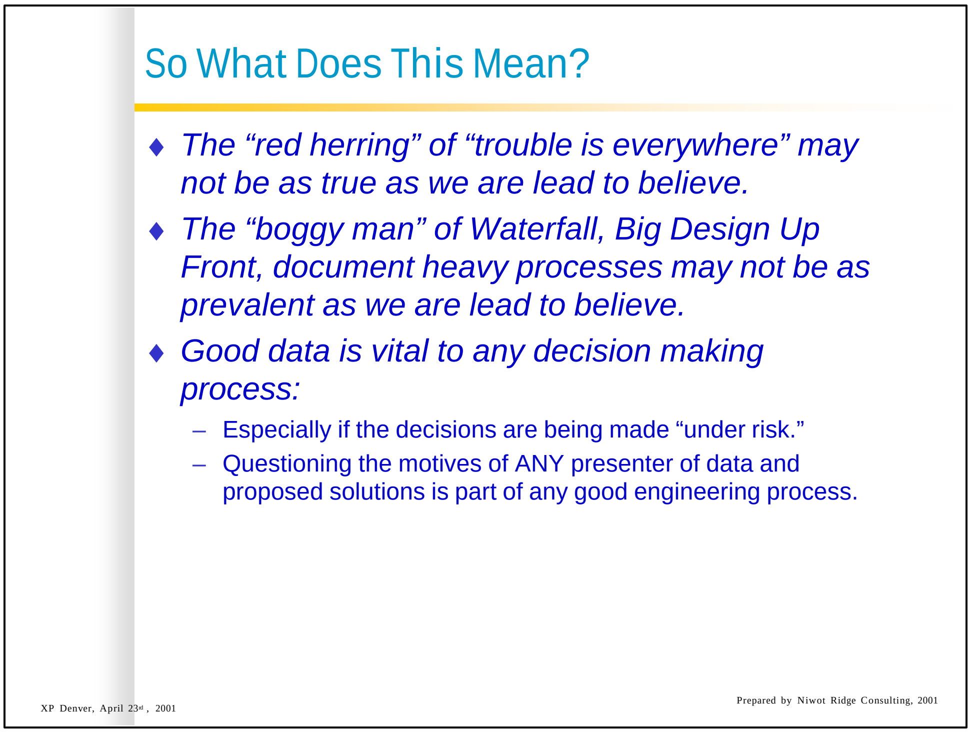
"Becoming Best in Class," Software Productivity Research, INCOSE, Capers Jones, 1999

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Another view of the maturity of the software industry is derived from a Capers Jones report.

In this view the majority of the development organizations are average with a second large percentage above average.



So What Does This Mean?

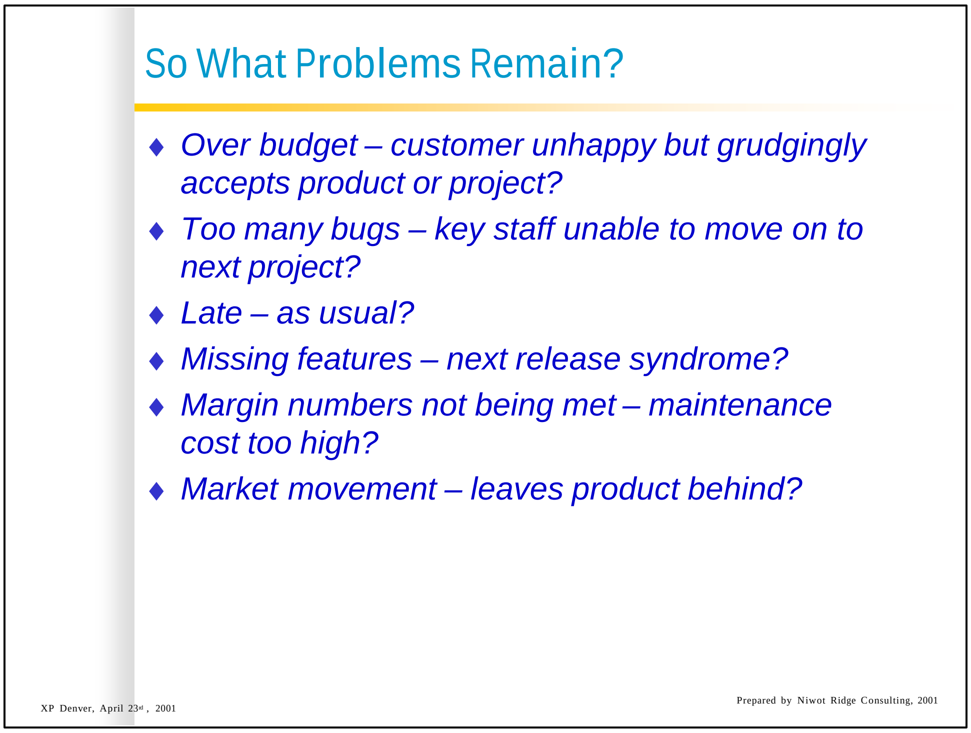
- *The “red herring” of “trouble is everywhere” may not be as true as we are lead to believe.*
- *The “boggy man” of Waterfall, Big Design Up Front, document heavy processes may not be as prevalent as we are lead to believe.*
- *Good data is vital to any decision making process:*
  - Especially if the decisions are being made “under risk.”
  - Questioning the motives of ANY presenter of data and proposed solutions is part of any good engineering process.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The CMM numbers would lead us to believe we’re all a bunch of losers and need to look for work in some other industry.

The Jones numbers would have us believe that everything is just fine and the problems are only at the fringes.



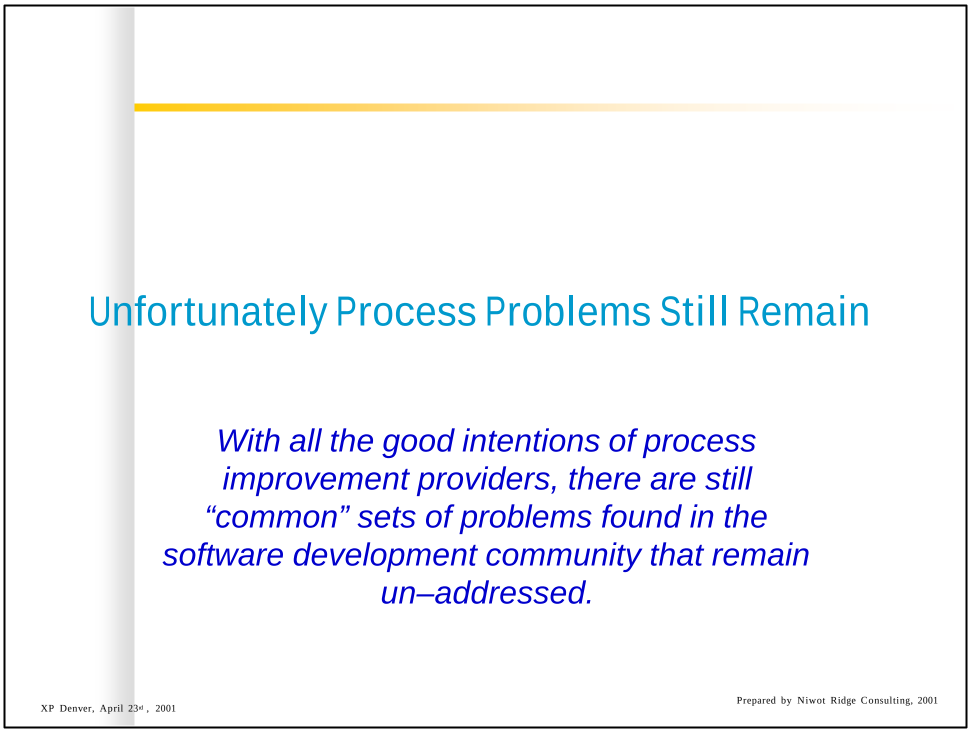
So What Problems Remain?

- *Over budget – customer unhappy but grudgingly accepts product or project?*
- *Too many bugs – key staff unable to move on to next project?*
- *Late – as usual?*
- *Missing features – next release syndrome?*
- *Margin numbers not being met – maintenance cost too high?*
- *Market movement – leaves product behind?*

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

We all know though that reality is somewhere in between. I doubt anyone in this room doesn't have at least one of these problems on their current project. At least if not now, then on a project in the past, and surely will encounter one of these problems in the future.



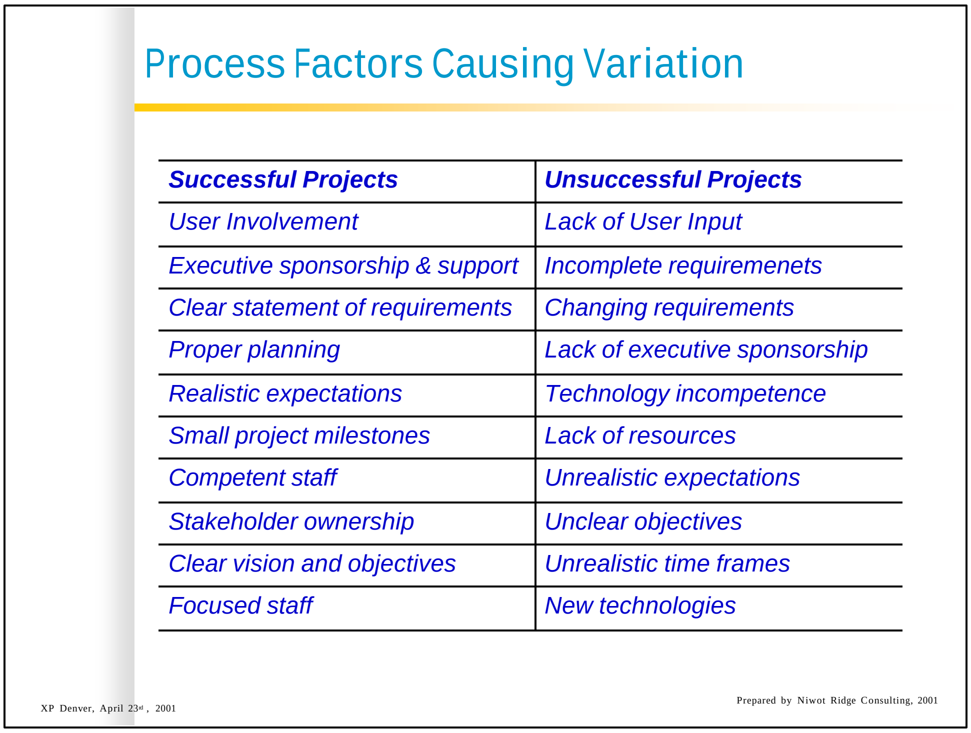
Unfortunately Process Problems Still Remain

With all the good intentions of process improvement providers, there are still "common" sets of problems found in the software development community that remain un–addressed.

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

The previous slide is simply restating the obvious … a tautology.



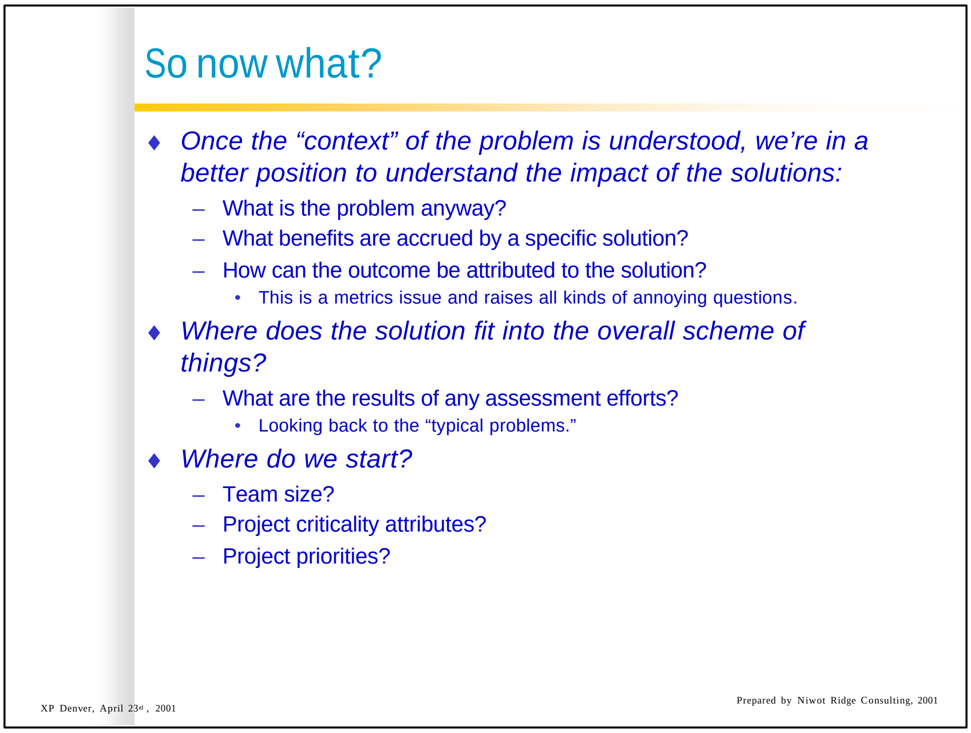
Process Factors Causing Variation

Successful Projects	***Unsuccessful Projects***
User Involvement	*Lack of User Input*
Executive sponsorship & support	*Incomplete requiremenets*
Clear statement of requirements	*Changing requirements*
Proper planning	*Lack of executive sponsorship*
Realistic expectations	*Technology incompetence*
Small project milestones	*Lack of resources*
Competent staff	*Unrealistic expectations*
Stakeholder ownership	*Unclear objectives*
Clear vision and objectives	*Unrealistic time frames*
Focused staff	*New technologies*

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

Here are some attributes of successful projects and projects that are, how shall we saying in a PC manner, challenged.



So now what?

- *Once the "context" of the problem is understood, we're in a better position to understand the impact of the solutions:*
  - What is the problem anyway?
  - What benefits are accrued by a specific solution?
  - How can the outcome be attributed to the solution?
    - This is a metrics issue and raises all kinds of annoying questions.
- *Where does the solution fit into the overall scheme of things?*
  - What are the results of any assessment efforts?
    - Looking back to the "typical problems."
- *Where do we start?*
  - Team size?
  - Project criticality attributes?
  - Project priorities?

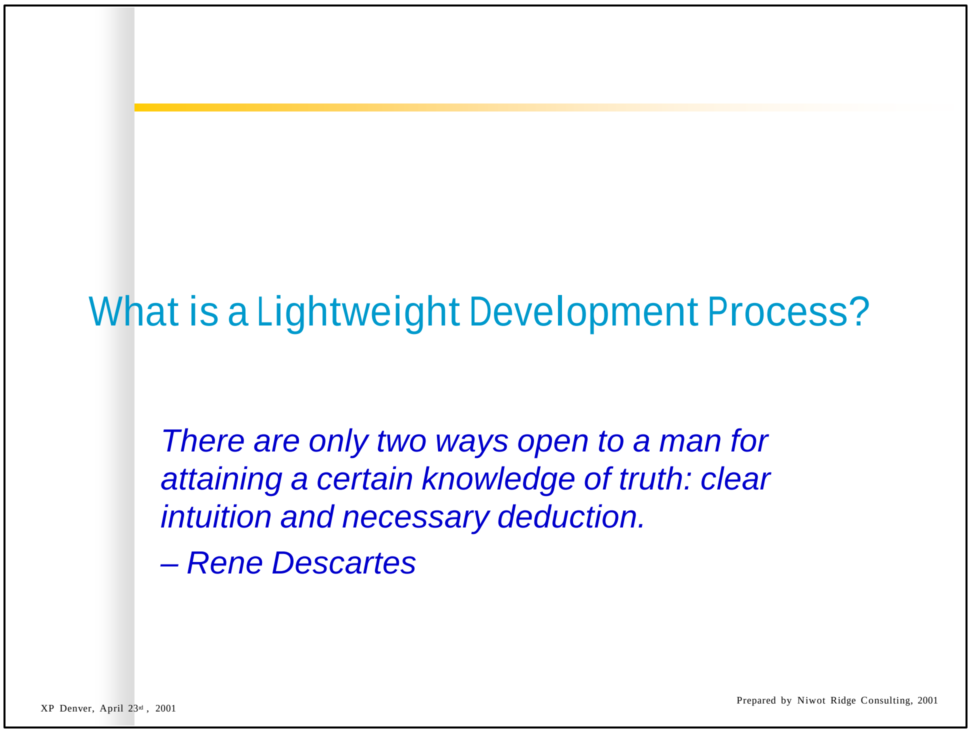
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

By first asking what the problem domain is, then asking what are the set of appropriate solutions to the problem, the focus of the development process moves from "let's find situations where this will work" (on site customer, PP, etc) to "let's find the best practices for the situation at hand."

This is approach has been suggested by many light weight process advocates. Alistair Cockburn's paper "Balancing Lightness with Sufficiency," is describes a similar approach.

We're now on the other side of the XP position. This may be uncomfortable for some and is not the approach taken by the XP advocates, but the reality of software development is there are "situational" aspects of projects that are out of our control.



What is a Lightweight Development Process?

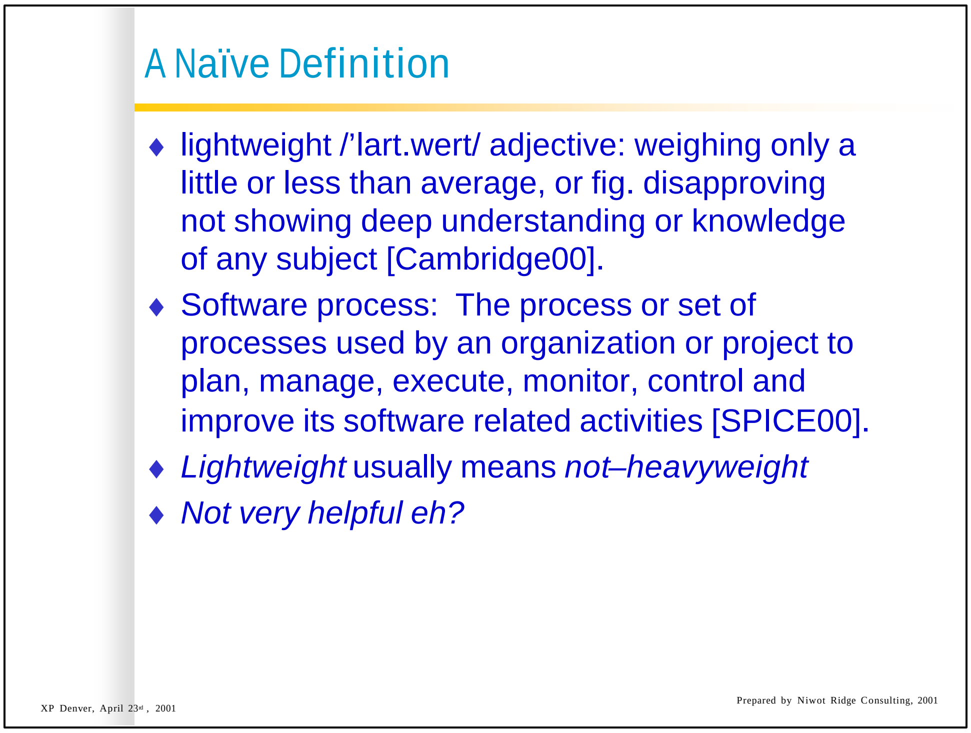
There are only two ways open to a man for attaining a certain knowledge of truth: clear intuition and necessary deduction.

– Rene Descartes

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Let's revisit the original question.



A Naïve Definition

- lightweight /'lart.wert/ adjective: weighing only a little or less than average, or fig. disapproving not showing deep understanding or knowledge of any subject [Cambridge00].
- Software process: The process or set of processes used by an organization or project to plan, manage, execute, monitor, control and improve its software related activities [SPICE00].
- *Lightweight* usually means *not–heavyweight*
- *Not very helpful eh?*

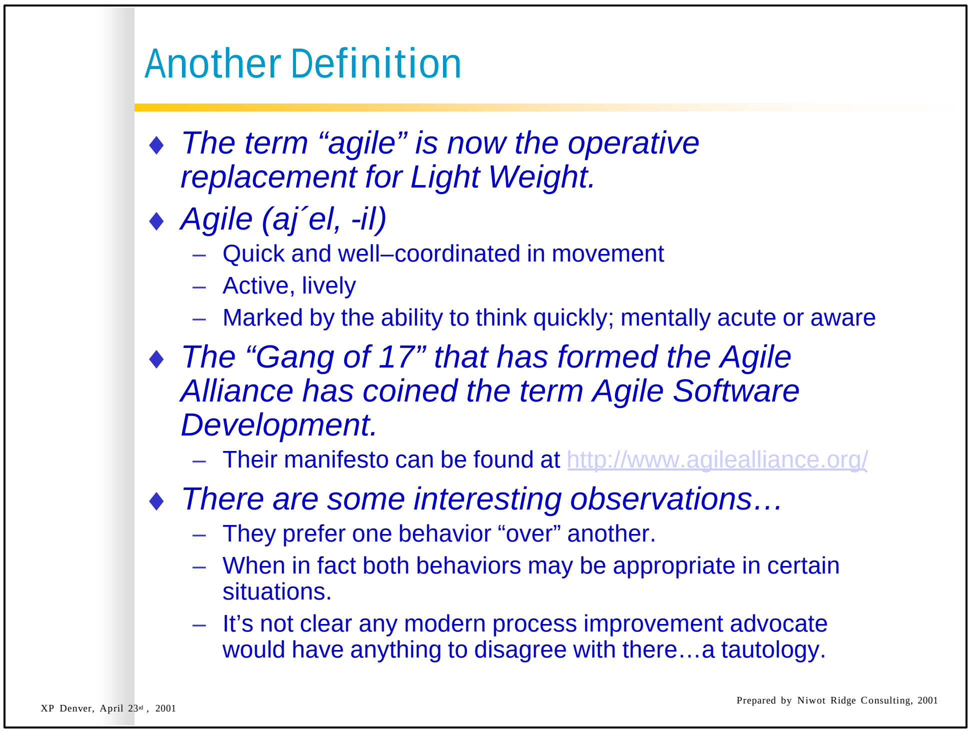
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The textbook definition of a lightweight process is actually useless in this case. In the end lightweight is a situational definition and means many things.

Is XP lightweight, yes. Is RUP lightweight, well compared to some heavy weight CMM Level 5 weapons systems development process, yes. How about ASD, SCRUM, Crystal, etc.

All these processes have some aspect of “lightness,” but what those aspects are cannot be dramatically different from other processes.



Another Definition

- *The term "agile" is now the operative replacement for Light Weight.*
- *Agile (aj´el, -il)*
  - Quick and well–coordinated in movement
  - Active, lively
  - Marked by the ability to think quickly; mentally acute or aware
- *The "Gang of 17" that has formed the Agile Alliance has coined the term Agile Software Development.*
  - Their manifesto can be found at http://www.agilealliance.org/
- *There are some interesting observations…*
  - They prefer one behavior "over" another.
  - When in fact both behaviors may be appropriate in certain situations.
  - It's not clear any modern process improvement advocate would have anything to disagree with there…a tautology.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

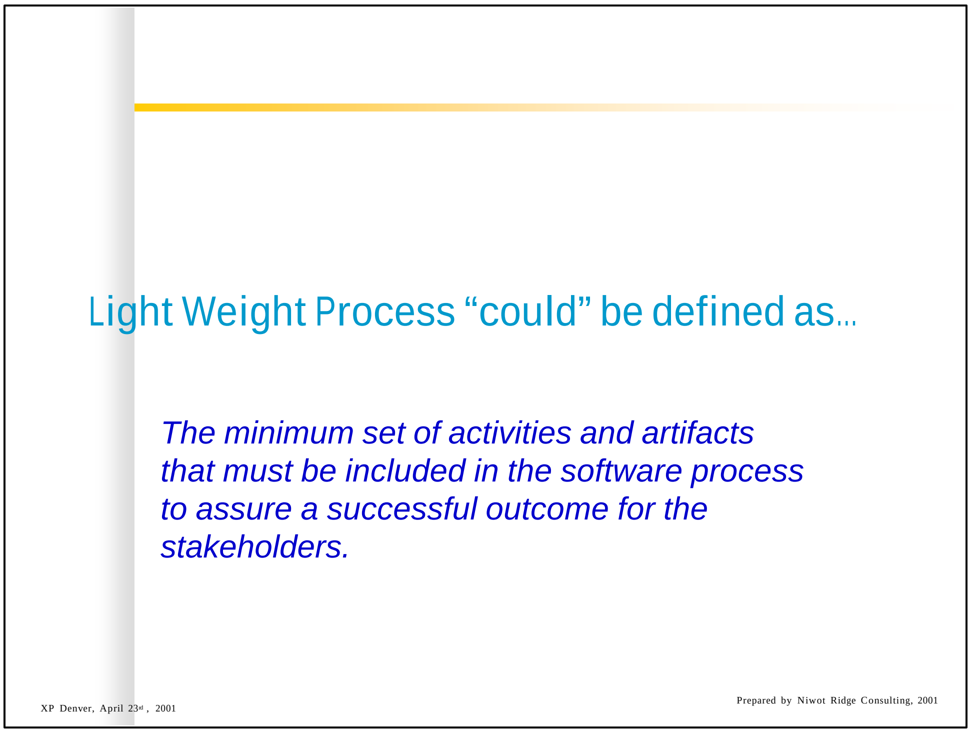
The words used in today's world have become more important than the meaning of the words. Light Weight has been replaced by "Agile."

The Agile methods approach doesn't really say much about the weightiness of the process, it's artifacts or the participants view of the burden of these artifacts.

The Agile Alliance Manifesto states some very nice behaviors for the management of projects. I would find it difficult to disagree with the manifesto. But the there so obvious "red herrings" in the manifesto.

- Individuals and actions must operate within some process framework using tools. Individuals without frameworks are mobs.
- Working software is a deliverable. In some cases the documentation is also a deliverable and may actually be more important than the software, since without it there is no business value.
- Customer collaboration is typical defined in some way. Even a verbal commitment is a contract in the eyes of the court.
- Good planning includes adapting to change. This is part of any professional project managers skill set.

So when the Gang of 17 say's "we value the left more than the right," they don't say by how much 1%, 1000%? If there is not large differentiation, then it's just nice words, but that's about it nice words and who can argue with nice words?



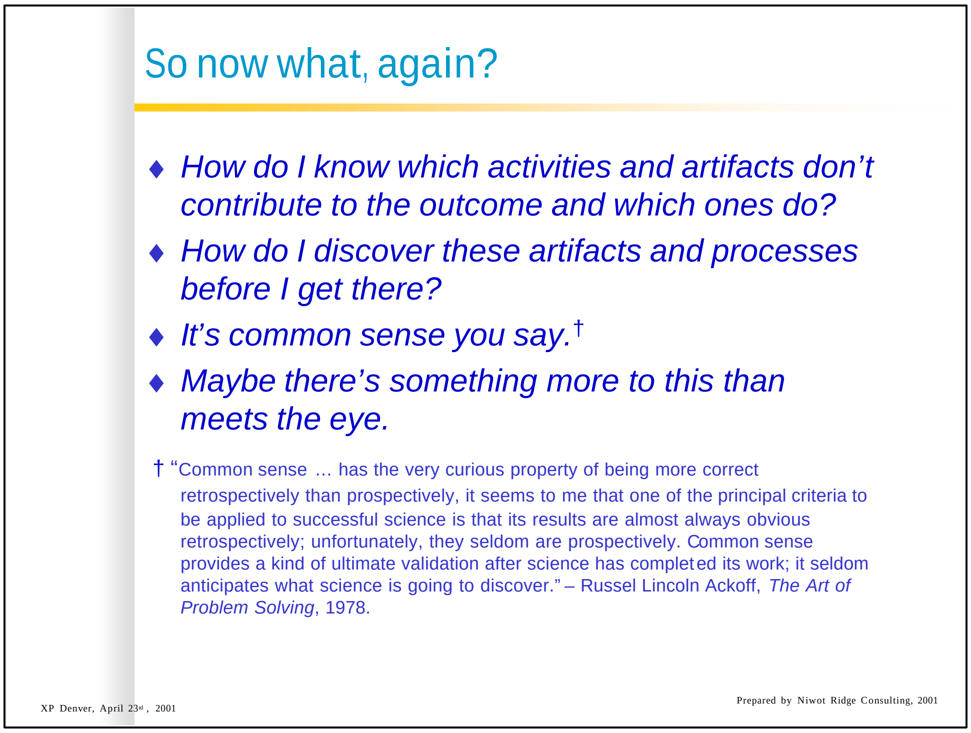
Light Weight Process "could" be defined as…

The minimum set of activities and artifacts that must be included in the software process to assure a successful outcome for the stakeholders.

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

A heuristic definition of a Light Weight Process might be.



So now what, again?

- *How do I know which activities and artifacts don't contribute to the outcome and which ones do?*
- *How do I discover these artifacts and processes before I get there?*
- *It's common sense you say.*†
- *Maybe there's something more to this than meets the eye.*

† "Common sense … has the very curious property of being more correct retrospectively than prospectively, it seems to me that one of the principal criteria to be applied to successful science is that its results are almost always obvious retrospectively; unfortunately, they seldom are prospectively. Common sense provides a kind of ultimate validation after science has completed its work; it seldom anticipates what science is going to discover." – Russel Lincoln Ackoff, *The Art of Problem Solving*, 1978.

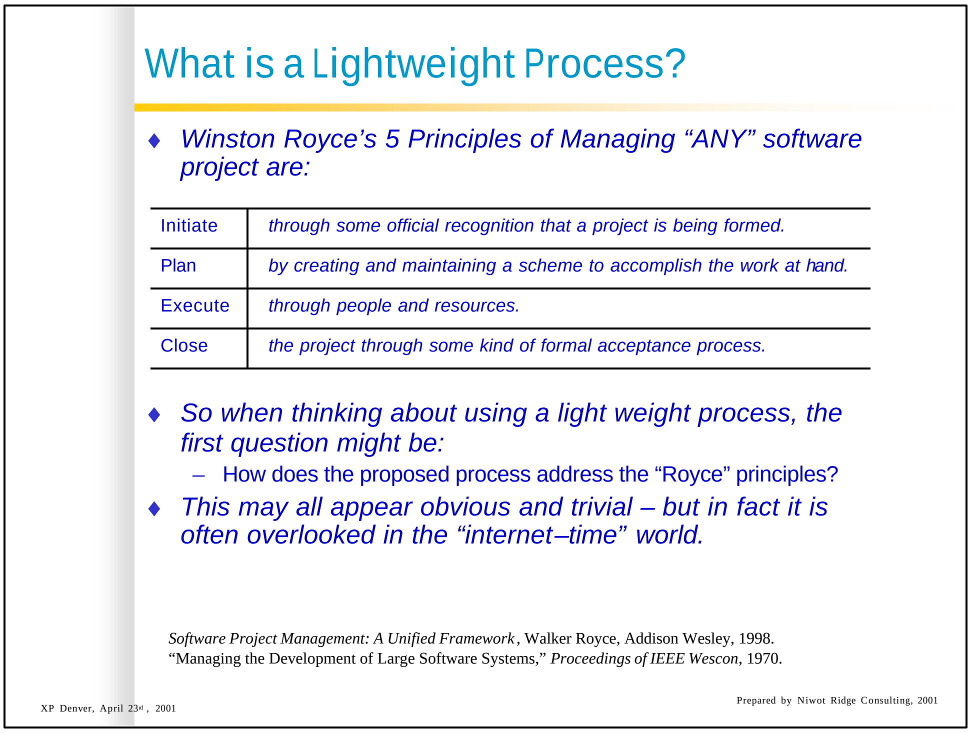
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

In the end the burden comes back to the development team.

What process to deploy is the responsibility of the team, the managers, the stakeholders in the outcome of the process.

Having all three participants "in the loop" creates a different situation then just picking a process and applying it to a problem.



What is a Lightweight Process?

- *Winston Royce's 5 Principles of Managing "ANY" software project are:*

Initiate	*through some official recognition that a project is being formed.*
Plan	*by creating and maintaining a scheme to accomplish the work at hand.*
Execute	*through people and resources.*
Close	*the project through some kind of formal acceptance process.*

- *So when thinking about using a light weight process, the first question might be:*
  - How does the proposed process address the "Royce" principles?
- *This may all appear obvious and trivial – but in fact it is often overlooked in the "internet–time" world.*

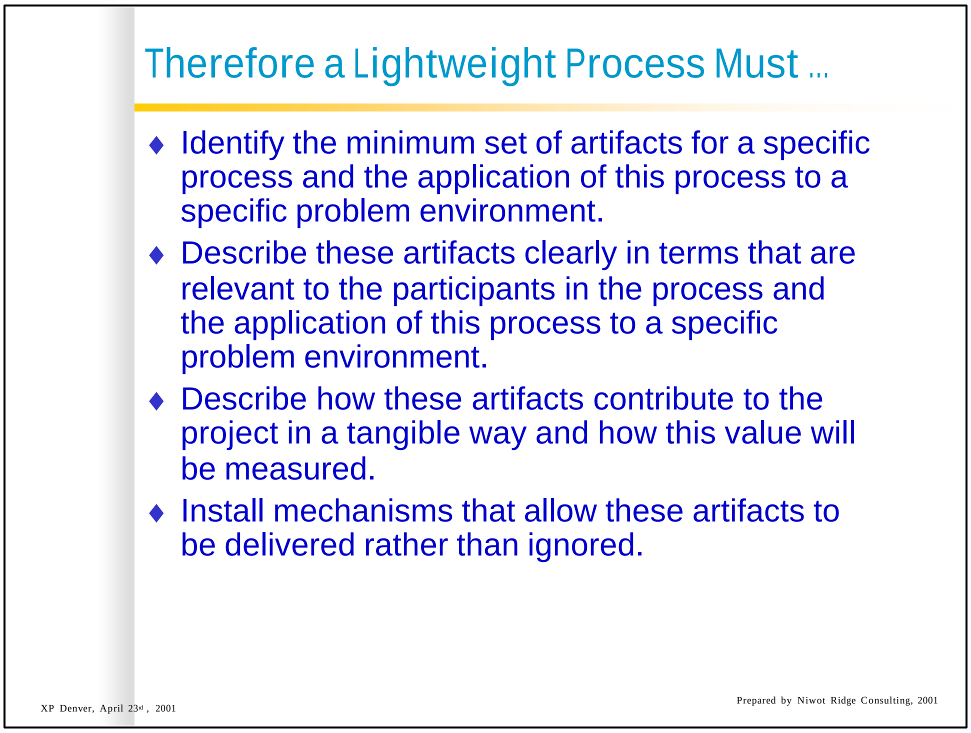
Software Project Management: A Unified Framework, Walker Royce, Addison Wesley, 1998.
"Managing the Development of Large Software Systems," *Proceedings of IEEE Wescon*, 1970.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

If we look back in time the problems that face development teams haven't changed "too" much. There are differences of course – internet time is one – but the core problems are nearly the same.

Winston Royce, the father of Walker Royce, described the changes that needed to take place within the development process to address these core problems. This was 1970.



Therefore a Lightweight Process Must …

- Identify the minimum set of artifacts for a specific process and the application of this process to a specific problem environment.
- Describe these artifacts clearly in terms that are relevant to the participants in the process and the application of this process to a specific problem environment.
- Describe how these artifacts contribute to the project in a tangible way and how this value will be measured.
- Install mechanisms that allow these artifacts to be delivered rather than ignored.

XP Denver, April 23rd, 2001

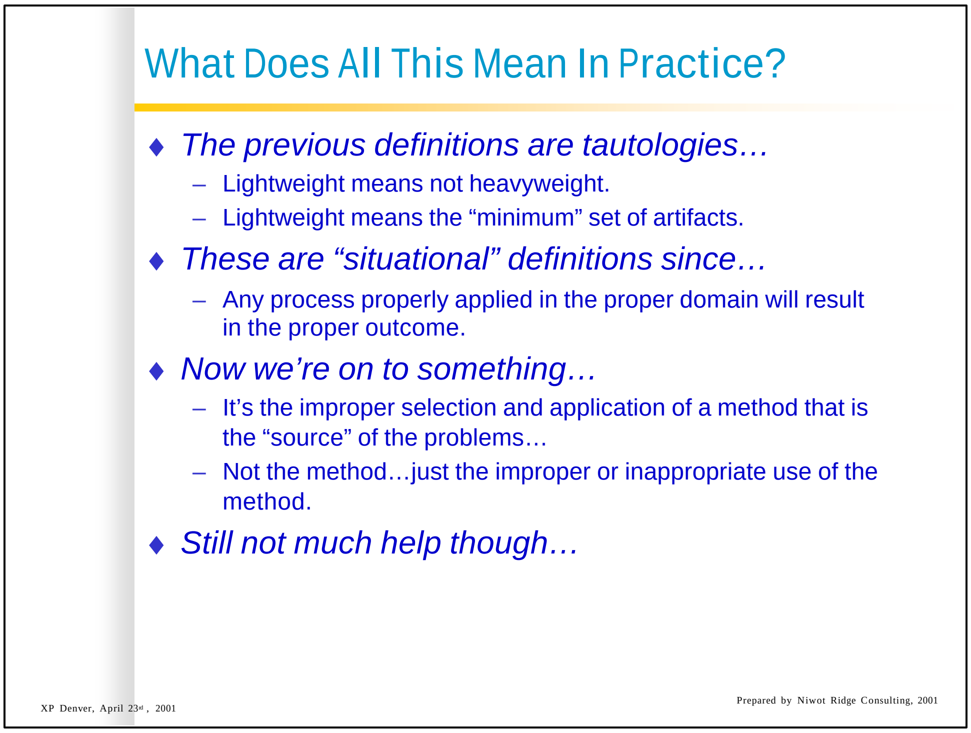
Prepared by Niwot Ridge Consulting, 2001

So here's a minimum set of attributes for a lightweight process.

Jim Highsmith would argue that "minimum" needs to be minimum plus a "little extra."

The description process should be appropriate for the project. If XP's CRC cards and User Stories are appropriate then use them. If something more persistent is needed then use it. This is a code word for documentation.

The connection between the practice and its benefit needs to be articulated "before" we start, not discovered in the end. Managers have a hard time going down the path of "we'll discover the benefits as we go along." It's just not in their nature.



What Does All This Mean In Practice?

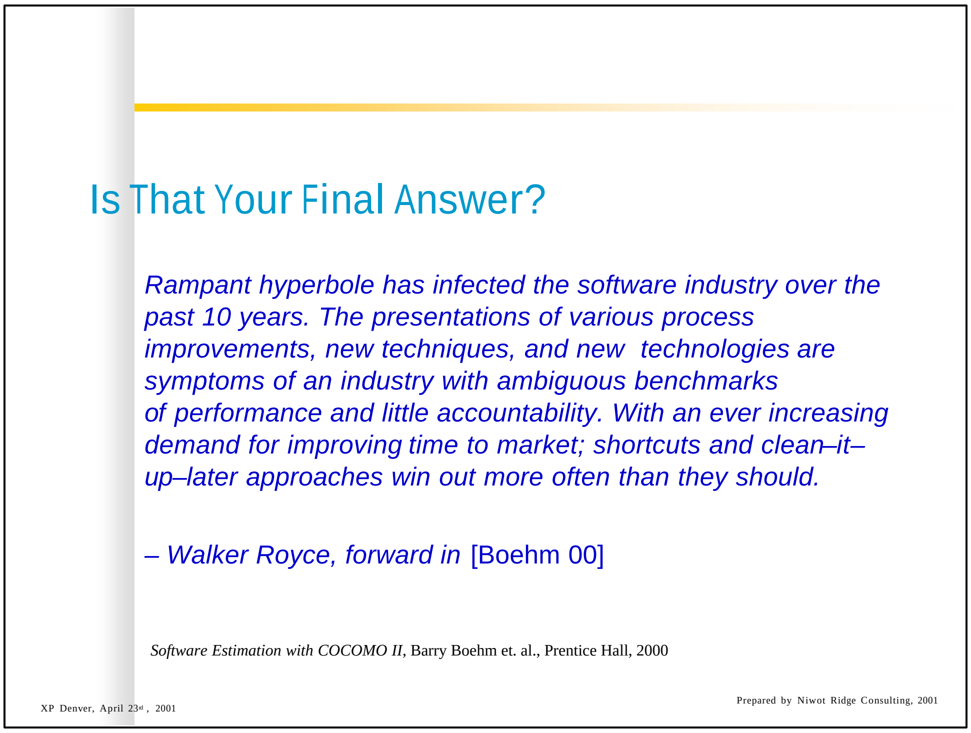
- *The previous definitions are tautologies…*
  - Lightweight means not heavyweight.
  - Lightweight means the "minimum" set of artifacts.
- *These are "situational" definitions since…*
  - Any process properly applied in the proper domain will result in the proper outcome.
- *Now we're on to something…*
  - It's the improper selection and application of a method that is the "source" of the problems…
  - Not the method…just the improper or inappropriate use of the method.
- *Still not much help though…*

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The real point here is to apply a process with proper expectations:

- Can I measure the benefits?
- Can all the participants agree on the outcomes?
- Are all the participants willing to behave as needed to reap the benefits?



Is That Your Final Answer?

Rampant hyperbole has infected the software industry over the past 10 years. The presentations of various process improvements, new techniques, and new technologies are symptoms of an industry with ambiguous benchmarks of performance and little accountability. With an ever increasing demand for improving time to market; shortcuts and clean–it–up–later approaches win out more often than they should.

– Walker Royce, forward in [Boehm 00]

Software Estimation with COCOMO II, Barry Boehm et. al., Prentice Hall, 2000

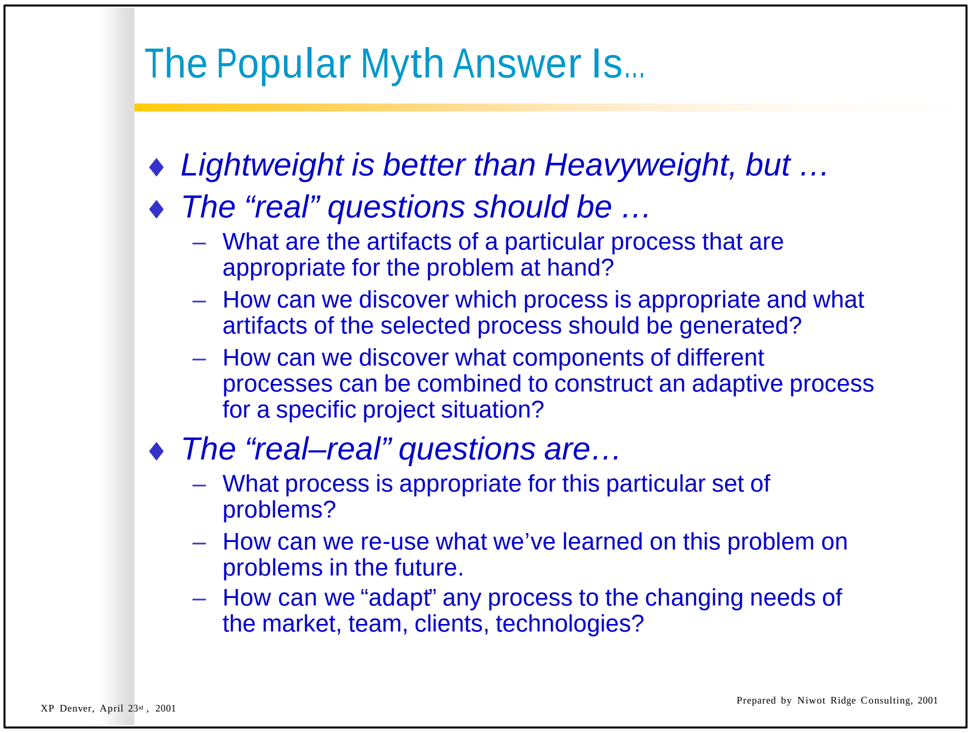
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

There is a theory in modern communications and evolutionary biology of "memes." Memes are viruses of the mind, ideas that are planted that take hold and change our view of the world. Advertisers use this concept to plant the seed for their products. News Editors use them to define the headlines of the day.

Lightweight processes has become a "meme" in the process improvement world.

I'm not saying there is a conspiracy out there, but the hyperbole of the "new new thing," as taken over our ability to analyze the situation and place ideas in the proper context.



The Popular Myth Answer Is…

- *Lightweight is better than Heavyweight, but …*
- *The "real" questions should be …*
  - What are the artifacts of a particular process that are appropriate for the problem at hand?
  - How can we discover which process is appropriate and what artifacts of the selected process should be generated?
  - How can we discover what components of different processes can be combined to construct an adaptive process for a specific project situation?
- *The "real–real" questions are…*
  - What process is appropriate for this particular set of problems?
  - How can we re-use what we've learned on this problem on problems in the future.
  - How can we "adapt" any process to the changing needs of the market, team, clients, technologies?

XP Denver, April 23rd , 2001

Prepared by Niwot Ridge Consulting, 2001

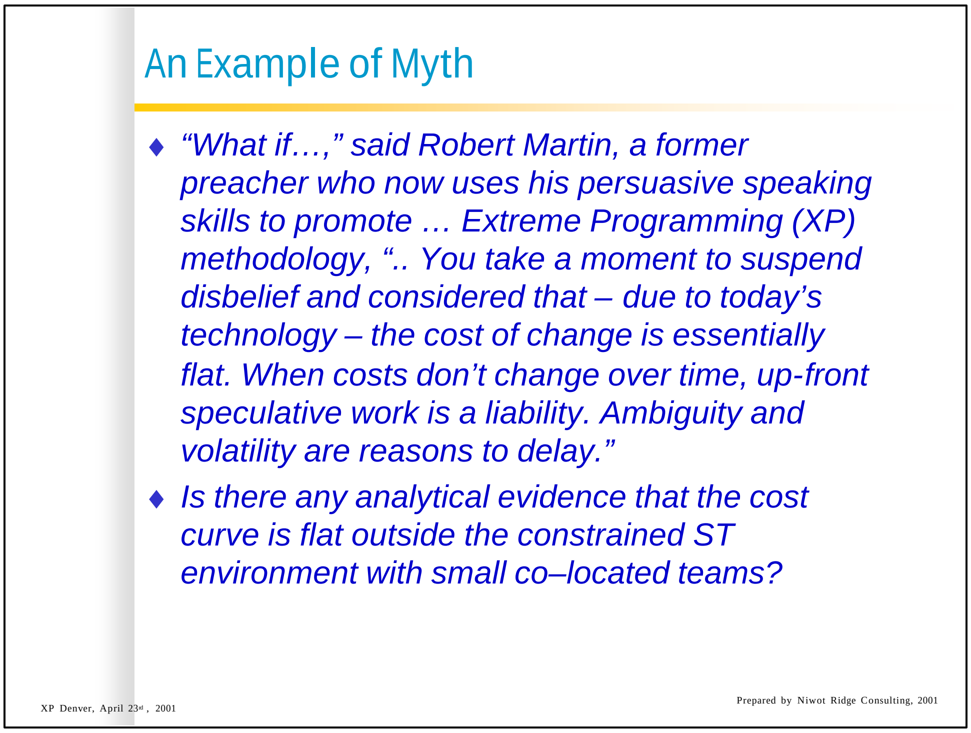
So let's ask the same set of questions in a slightly different way and try to discover what is "appropriate" for the situation at hand, not what process is better than another outside the problem domain.

Is XP appropriate when these situations are not present? The XP advocates provide different answers depending on the forum. So let's look in the back of the White Book. By the way this is the suggested way to read the White Book, since placing XP in context will prevent confusion when reading the practices.

XP advocates use the words, "I can't tell you where it won't work, so I'll assume it can work, until proven otherwise." This allows XP advocates to say "since I don't know it won't work - it must work…" The forums are full of circular arguments of applicability that have no basis in logic.

You shouldn't try XP when (pg. 156-157 of the White Book):

- Culture doesn't support it – this is a big loop hole.
- Team size – team size relates to project size BTW. Building SAP requires a minimum numbers of developers.
- Single thread integration – concurrent or distributed development creates problems XP can't deal with.
- Exponential cost curve – data on this curve is well developed and flat cost curve projects are actually rare.
- Long feed back loops – are the norm in organizations without on-site "everyone."



An Example of Myth

- *"What if…," said Robert Martin, a former preacher who now uses his persuasive speaking skills to promote … Extreme Programming (XP) methodology, ".. You take a moment to suspend disbelief and considered that – due to today's technology – the cost of change is essentially flat. When costs don't change over time, up-front speculative work is a liability. Ambiguity and volatility are reasons to delay."*
- *Is there any analytical evidence that the cost curve is flat outside the constrained ST environment with small co–located teams?*

XP Denver, April 23rd, 2001

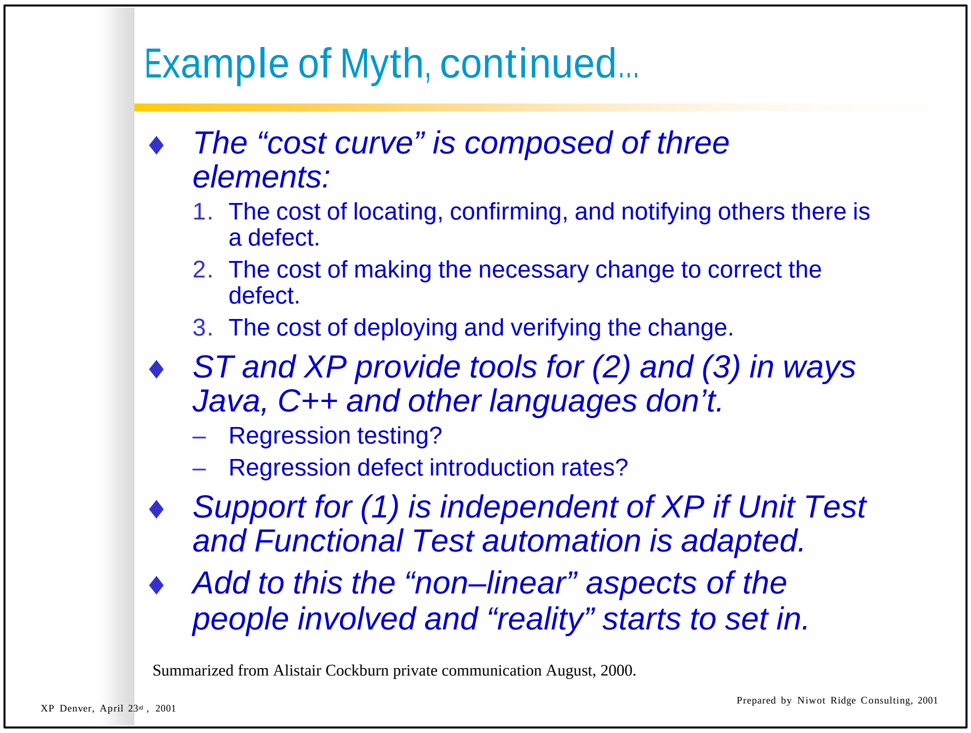
Prepared by Niwot Ridge Consulting, 2001

Martin's statement is interesting. But the context of the statement needs to be examined.

"Suspending ones belief," is not a natural process for the scientific minded among us.

However, driving the cost curve down using technology benefits everyone and every development process.

The question is "does any specific development process adapt better or worse to a flattening cost curve?" Obvious the XP advocates lay claim to being the best method to take advantage of this situation.



Example of Myth, continued…

- *The "cost curve" is composed of three elements:*
  1. The cost of locating, confirming, and notifying others there is a defect.
  2. The cost of making the necessary change to correct the defect.
  3. The cost of deploying and verifying the change.
- *ST and XP provide tools for (2) and (3) in ways Java, C++ and other languages don't.*
  - Regression testing?
  - Regression defect introduction rates?
- *Support for (1) is independent of XP if Unit Test and Functional Test automation is adapted.*
- *Add to this the "non–linear" aspects of the people involved and "reality" starts to set in.*

Summarized from Alistair Cockburn private communication August, 2000.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The "myth" of XP is that the cost curve is flat is based on several assumptions:

- Small teams
- Simple code
- On-site customer
- Refactored code base

All the principles of XP. This approach also assumes that all these situations are possible. In some cases they are, and in some cases they aren't.

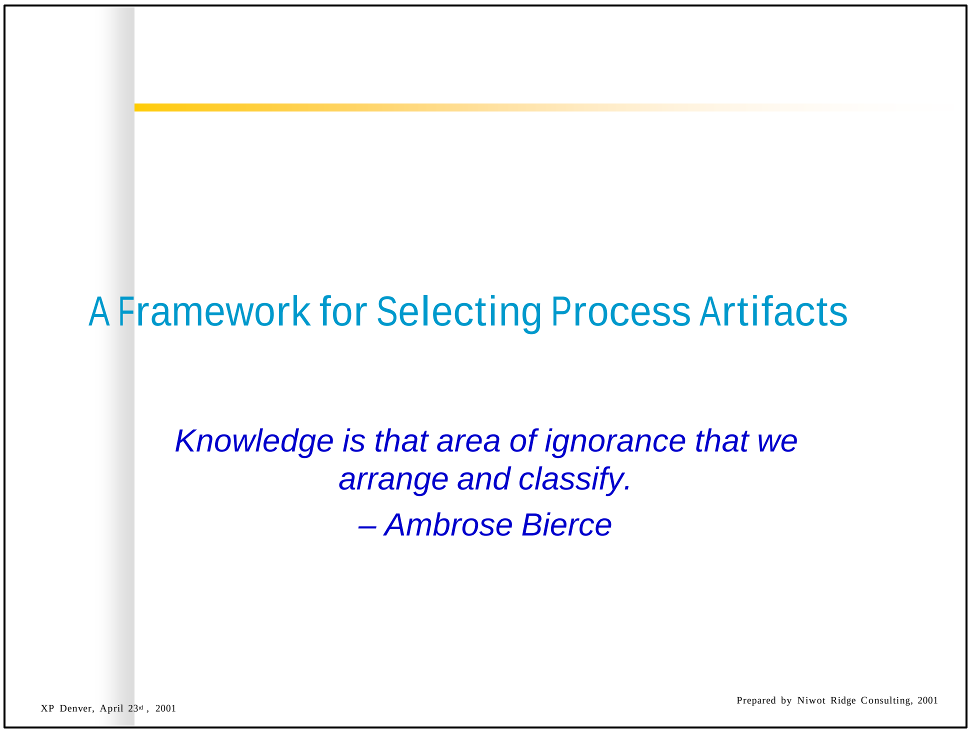
What XP misses out on is the "qualifier" that the XP benefits are constrained to a fairly small subset of development problems, with XP-attributes.

Using the Jones numbers the average number of staff ranges from 5 to 20, discounting the personal software domain.

To make the statement "suspend belief and assume costs are flat" begs the question, what happens when it is discovered that the cost curve is NOT flat?

The other side of this argument is related to the defect generation rate. If XP generates fewer defects, then the overall cost curve will be less in the end.

No analytical data to date exists to show this however, when compared to other methods. Again the Jones numbers should be examined for the background.



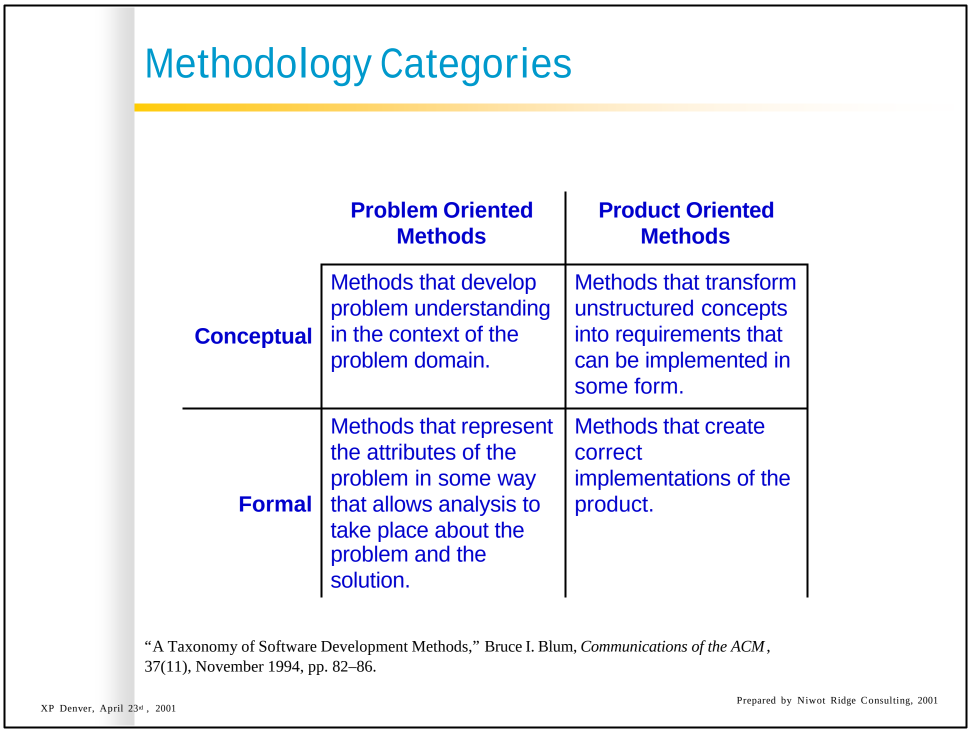
A Framework for Selecting Process Artifacts

Knowledge is that area of ignorance that we arrange and classify.
– Ambrose Bierce

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Now that we have put some light on the “appropriateness” topic, lets’ look at a framework for deploying lightweight processes.



Methodology Categories

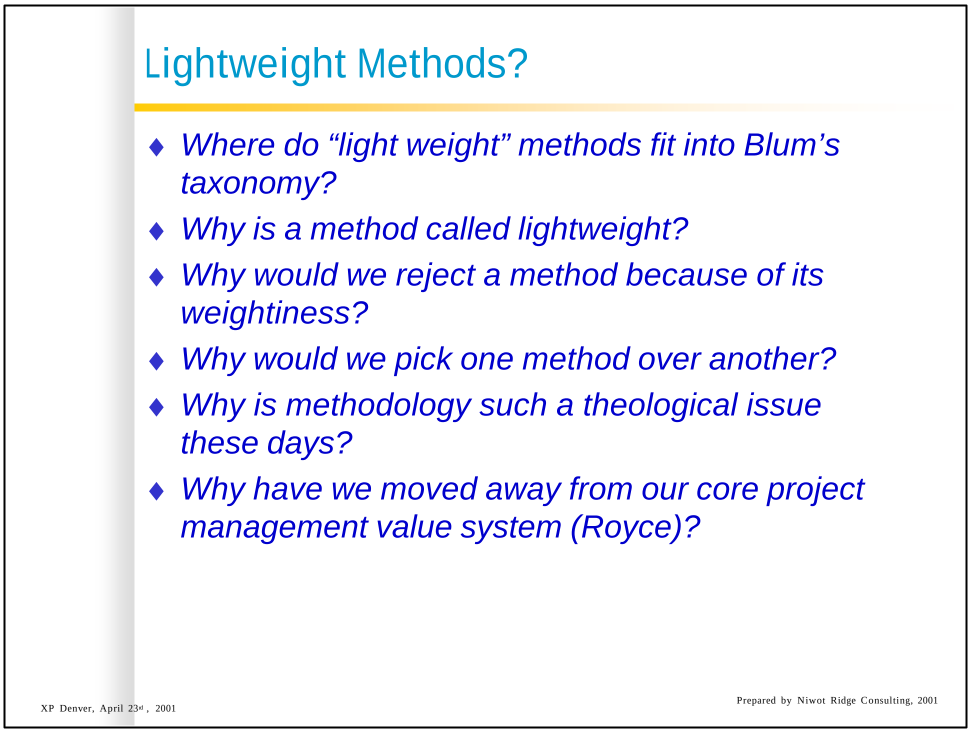
	Problem Oriented Methods	Product Oriented Methods
Conceptual	Methods that develop problem understanding in the context of the problem domain.	Methods that transform unstructured concepts into requirements that can be implemented in some form.
Formal	Methods that represent the attributes of the problem in some way that allows analysis to take place about the problem and the solution.	Methods that create correct implementations of the product.

"A Taxonomy of Software Development Methods," Bruce I. Blum, *Communications of the ACM*, 37(11), November 1994, pp. 82–86.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Blum has created a taxonomy of methodologies. This can be useful in partitioning the problem domain.



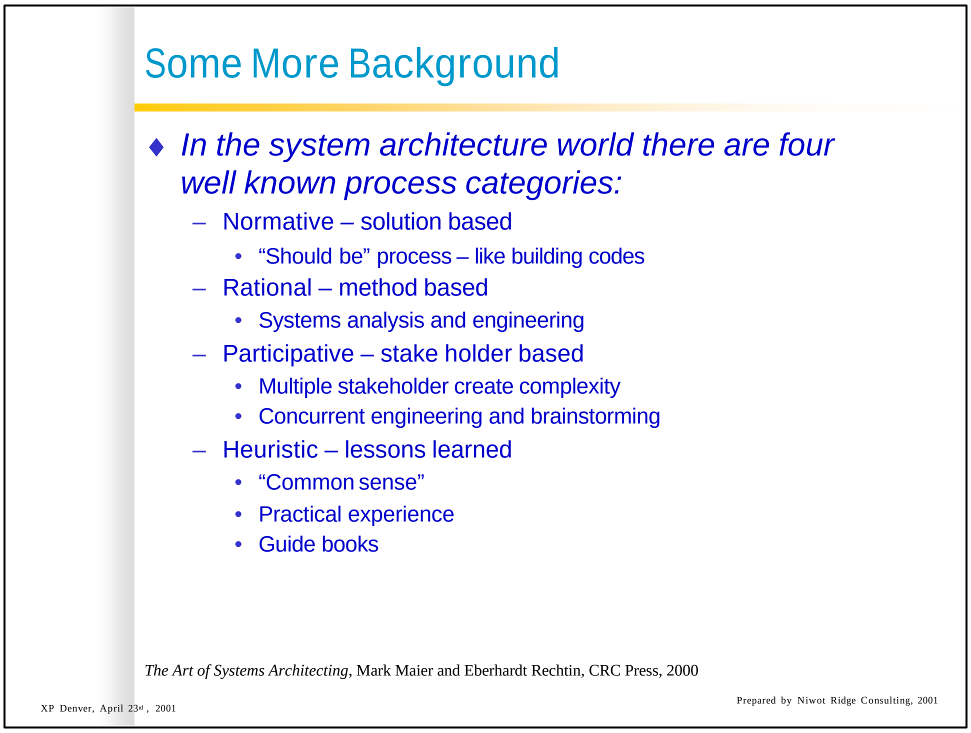
Lightweight Methods?

- *Where do “light weight” methods fit into Blum’s taxonomy?*
- *Why is a method called lightweight?*
- *Why would we reject a method because of its weightiness?*
- *Why would we pick one method over another?*
- *Why is methodology such a theological issue these days?*
- *Why have we moved away from our core project management value system (Royce)?*

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Here’s some more questions on the appropriateness of one process over another. These questions are good to ask over and over again, since once we get hooked on the theological issues of a process it is hard to break the loop of “buzz.”



Some More Background

- *In the system architecture world there are four well known process categories:*
  - Normative – solution based
    - "Should be" process – like building codes
  - Rational – method based
    - Systems analysis and engineering
  - Participative – stake holder based
    - Multiple stakeholder create complexity
    - Concurrent engineering and brainstorming
  - Heuristic – lessons learned
    - "Common sense"
    - Practical experience
    - Guide books

The Art of Systems Architecting, Mark Maier and Eberhardt Rechtin, CRC Press, 2000

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

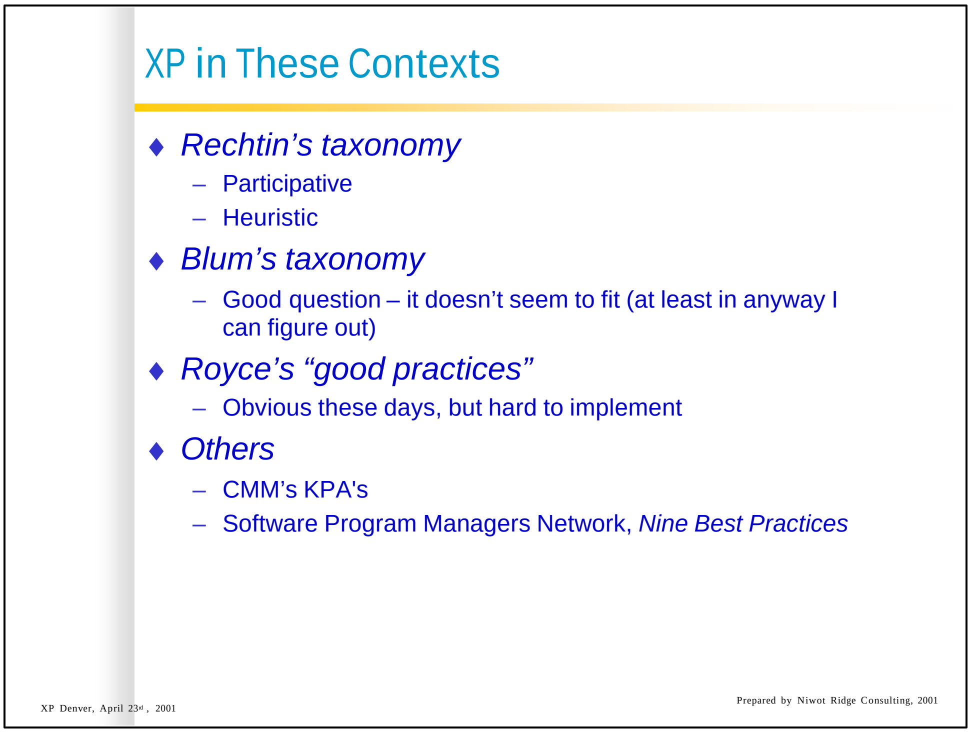
Another place to look for a mature view of process improvement is the Systems Architecture domain.

This is not the systems architecture of code and software (although they are a subset), but the architecture of complex systems. Say a ballistic missile system or a communications satellite.

Rechtin is the "father" of systems architecture. His work is now the standard description of how to build complex systems. It is not well known outside the aerospace business but is becoming more popular.

Rechtin's describes four process categories: normative, rational, participative, and heuristic. The heuristic process is now the way to build space craft and weapons systems. This doesn't mean we make it up as we go along, but that there are known ways to do things, and these "rules of thumb" are the basis of design. New and innovative designs are produced but they are based on the rules of thumb. Strength of materials, classical mechanics, electrodynamics, etc. are the basis of these heuristics.

Software development doesn't have these "first principles," so we're in big trouble when something goes wrong.



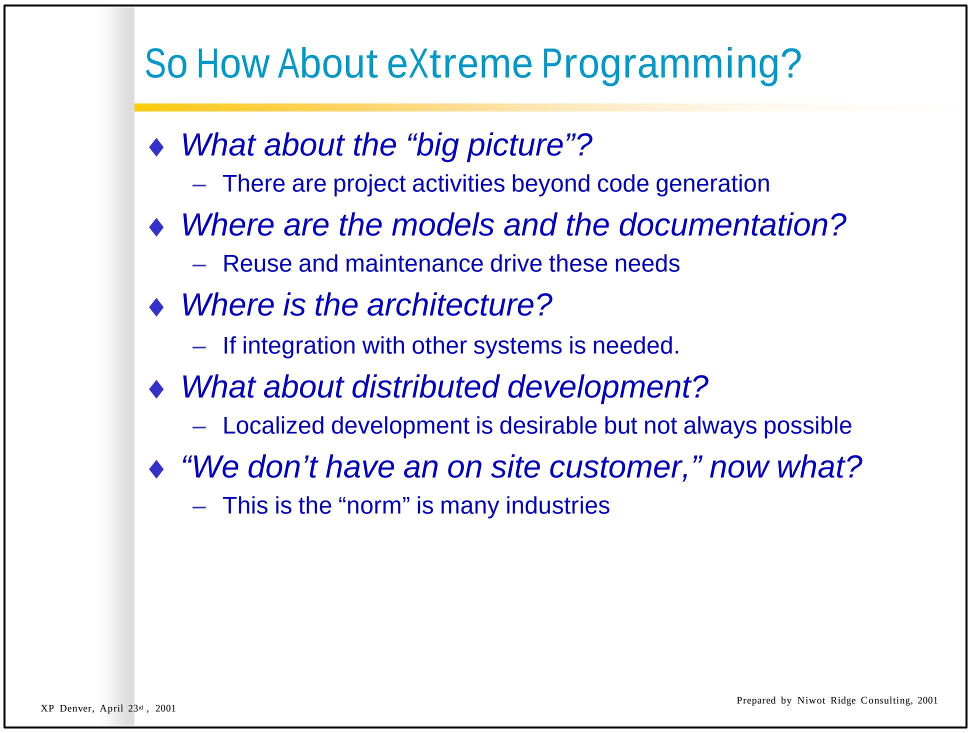
XP in These Contexts

- *Rechtin's taxonomy*
  - Participative
  - Heuristic
- *Blum's taxonomy*
  - Good question – it doesn't seem to fit (at least in anyway I can figure out)
- *Royce's "good practices"*
  - Obvious these days, but hard to implement
- *Others*
  - CMM's KPA's
  - Software Program Managers Network, *Nine Best Practices*

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Now that we have at least three different frameworks to examine XP. Let's try to fit it in.



So How About eXtreme Programming?

- *What about the "big picture"?*
  - There are project activities beyond code generation
- *Where are the models and the documentation?*
  - Reuse and maintenance drive these needs
- *Where is the architecture?*
  - If integration with other systems is needed.
- *What about distributed development?*
  - Localized development is desirable but not always possible
- *"We don't have an on site customer," now what?*
  - This is the "norm" is many industries

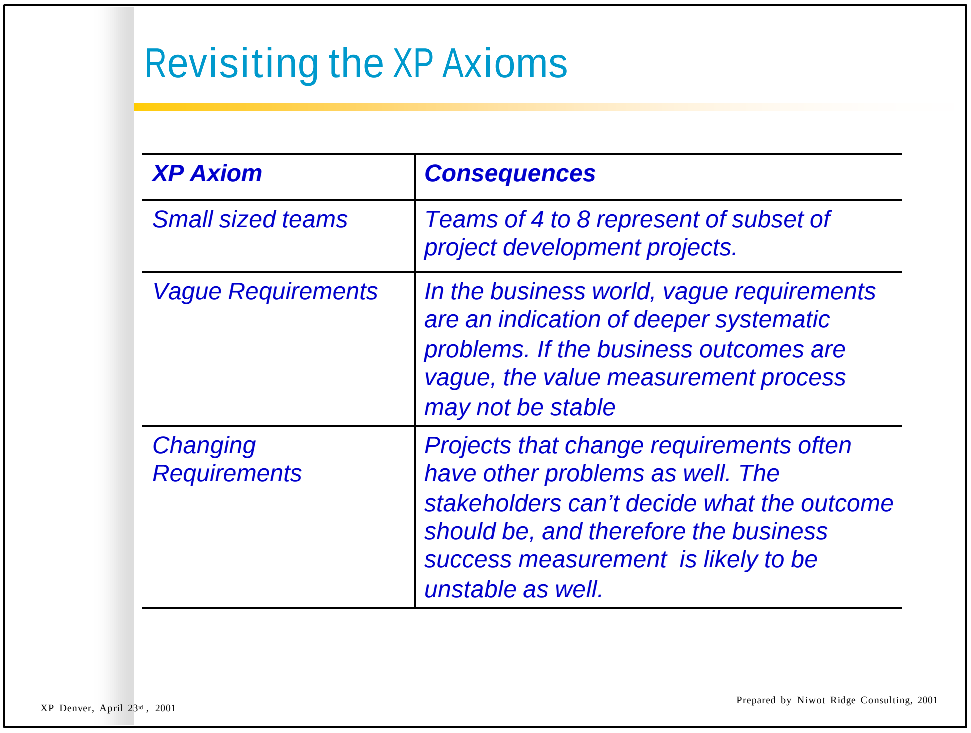
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Within this framework there are a new set of questions.

How does XP extend its practices to non-programming activities?

- How can a team of XP programmers interact with management, finance, product marketing, extended customer?
- How can the "necessary" documentation be incorporated into an XP project? Can this documentation be used for professional grade materials for the project or product? Is there such a thing a "eXterme Technical Writing?"
- Although architecture is self describing and possibly self creating in XP, the architecture issues are prominent in many business processing systems using COTS products. How can XP address these issues?
- On-Site customer and small team development is not always possible. In the EAI integration world this is rare. How can XP address the distributed development issues? (Ron Crocker has some experience here)



Revisiting the XP Axioms

XP Axiom	Consequences
Small sized teams	Teams of 4 to 8 represent of subset of project development projects.
Vague Requirements	In the business world, vague requirements are an indication of deeper systematic problems. If the business outcomes are vague, the value measurement process may not be stable
Changing Requirements	Projects that change requirements often have other problems as well. The stakeholders can't decide what the outcome should be, and therefore the business success measurement is likely to be unstable as well.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

There is an interesting approach here. XP makes the claim that IF the situation is a:

- Small team
- Vague requirements
- Changing requirements

Then XP is an appropriate process.

The flip side of this is:

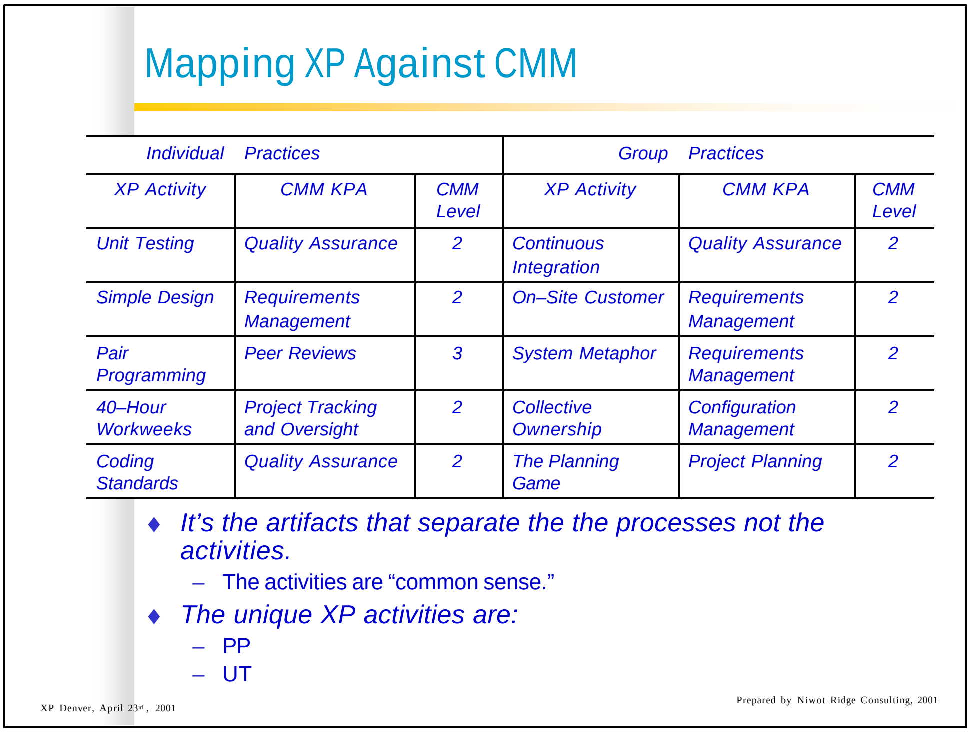
- Get the requirements sufficiently defined to control the project
- Stabilize the requirements so value can be delivered

This is the ying and yang of the process world. Do I have a process that deals with Chaos or do I get control of the Chaos.

The answer comes back to what is appropriate for the situation.

In some environment vague and changing requirements are the norm. in others they are not.

The risk is to inject the boggie man of BDUF, CMM, heavy weight process instead of asking the "appropriateness" question.



Mapping XP Against CMM

Individual Practices			Group Practices		
XP Activity	CMM KPA	CMM Level	XP Activity	CMM KPA	CMM Level
Unit Testing	Quality Assurance	2	Continuous Integration	Quality Assurance	2
Simple Design	Requirements Management	2	On–Site Customer	Requirements Management	2
Pair Programming	Peer Reviews	3	System Metaphor	Requirements Management	2
40–Hour Workweeks	Project Tracking and Oversight	2	Collective Ownership	Configuration Management	2
Coding Standards	Quality Assurance	2	The Planning Game	Project Planning	2

- *It's the artifacts that separate the the processes not the activities.*
  - The activities are "common sense."
- *The unique XP activities are:*
  - PP
  - UT

XP Denver, April 23rd, 2001 Prepared by Niwot Ridge Consulting, 2001

Although it is a bit of a stretch, and probably not well accepted by the XP community, the mapping of XP practices to the CMM Key Process Areas is straight forward.

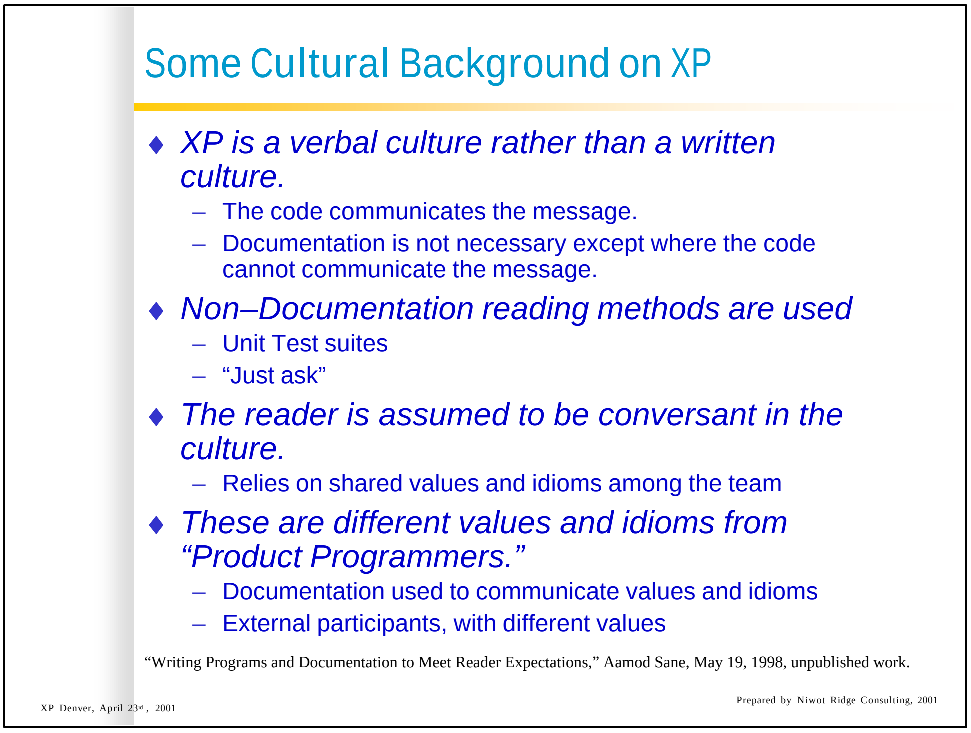
What is missing of course is the XP is "artifact light," and users of CMM expect some form of persistent artifacts to be produced. So the mapping falls apart if the "literal interpretation of XP is taken,"produce no artifacts other than the code.

This can be addressed by simply defining what documents are needed for the specific project and defining User Storied for these documents.

Extending this would have these document artifacts "per defined" by the stakeholders or the development organization.

Extending this further the configuration management and the means of expressing the User Stories and Metaphor could also be "standardized."

At this point XP looks similar to other light weight processes with a framework of supporting documents (only those necessary or deemed necessary by the customer).



Some Cultural Background on XP

- *XP is a verbal culture rather than a written culture.*
  - The code communicates the message.
  - Documentation is not necessary except where the code cannot communicate the message.
- *Non–Documentation reading methods are used*
  - Unit Test suites
  - "Just ask"
- *The reader is assumed to be conversant in the culture.*
  - Relies on shared values and idioms among the team
- *These are different values and idioms from "Product Programmers."*
  - Documentation used to communicate values and idioms
  - External participants, with different values

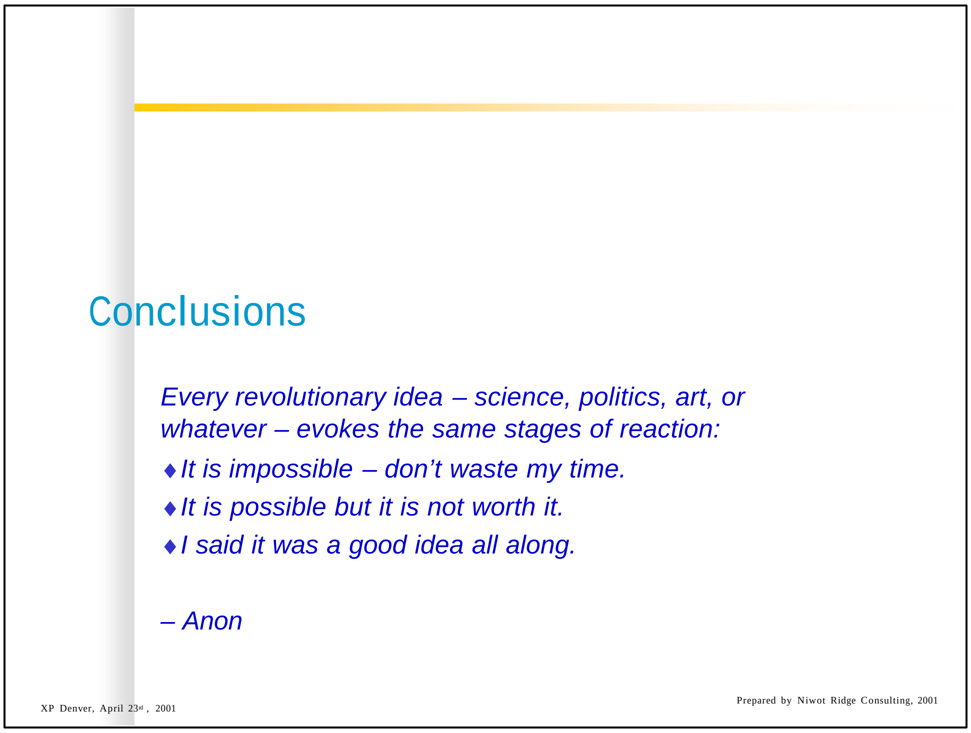
"Writing Programs and Documentation to Meet Reader Expectations," Aamod Sane, May 19, 1998, unpublished work.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The cultural aspects of XP and other Light Weight Processes is just now being understood (at least by those outside the process domain). XP has a unique set of cultural attributes. Understanding these attributes and how they effect the deployment and use of XP is important.

If you don't take the time to understand the cultural impacts on your organization, then these will come back to bite you.



Conclusions

Every revolutionary idea – science, politics, art, or whatever – evokes the same stages of reaction:

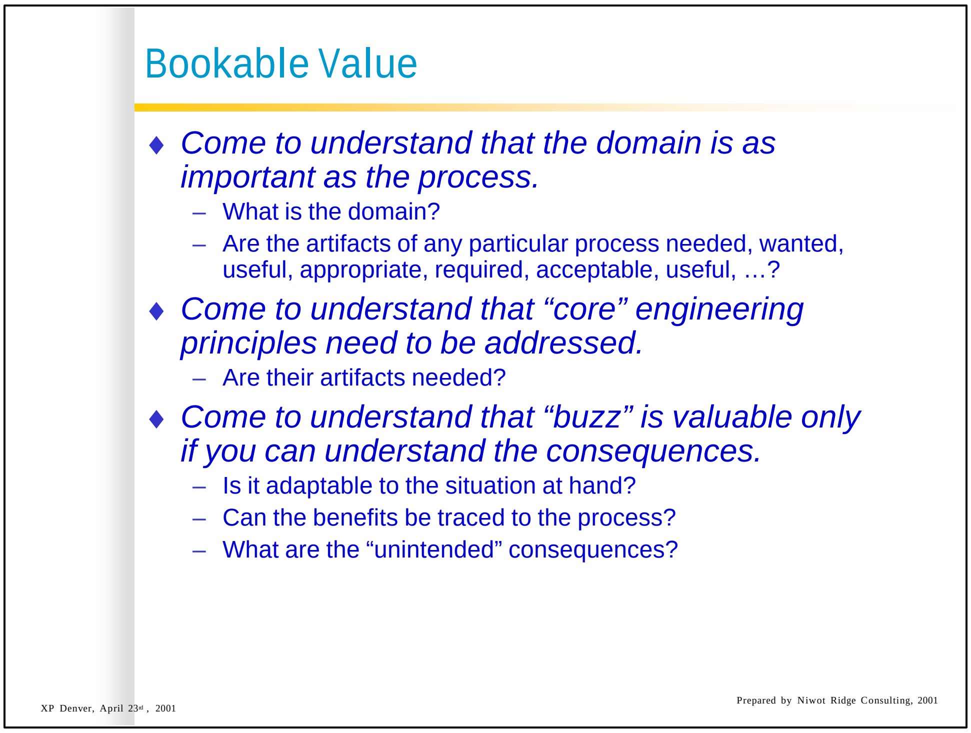
- *It is impossible – don't waste my time.*
- *It is possible but it is not worth it.*
- *I said it was a good idea all along.*

– Anon

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

As we near the end here, let's ask about value.



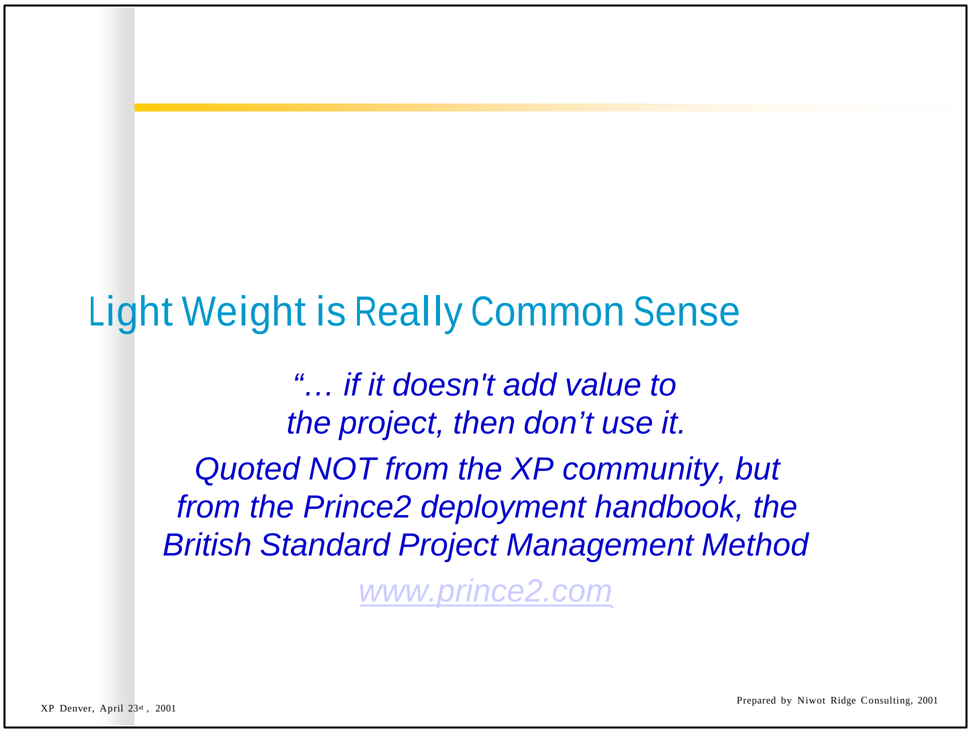
Bookable Value

- *Come to understand that the domain is as important as the process.*
  - What is the domain?
  - Are the artifacts of any particular process needed, wanted, useful, appropriate, required, acceptable, useful, …?
- *Come to understand that "core" engineering principles need to be addressed.*
  - Are their artifacts needed?
- *Come to understand that "buzz" is valuable only if you can understand the consequences.*
  - Is it adaptable to the situation at hand?
  - Can the benefits be traced to the process?
  - What are the "unintended" consequences?

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

What topics have we visited?



Light Weight is Really Common Sense

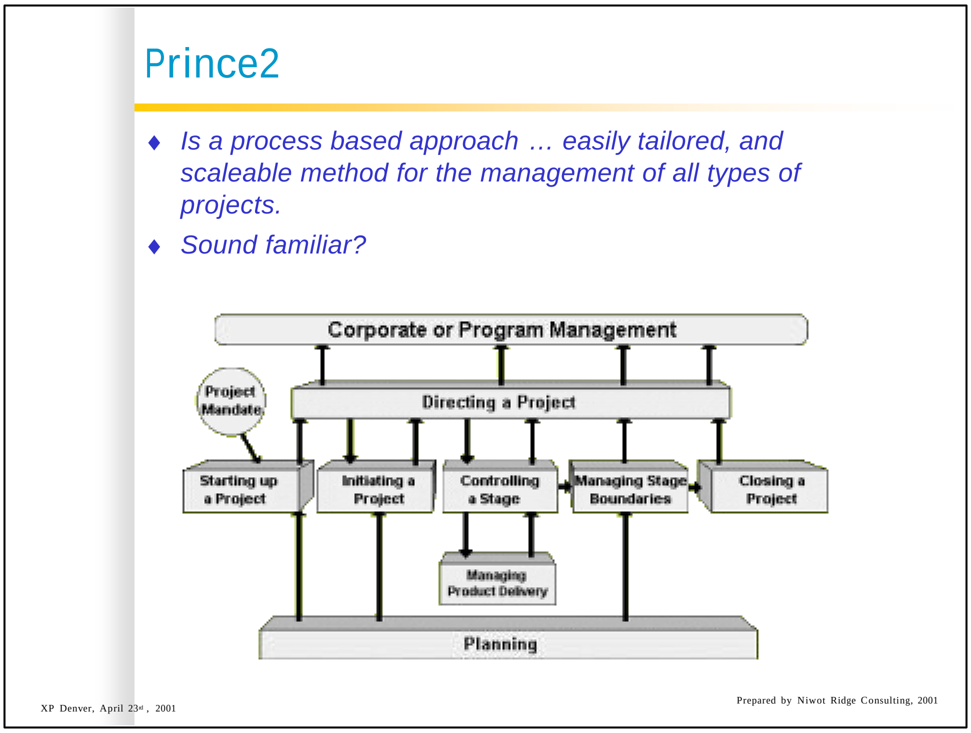
"… if it doesn't add value to the project, then don't use it.

Quoted NOT from the XP community, but from the Prince2 deployment handbook, the British Standard Project Management Method

www.prince2.com

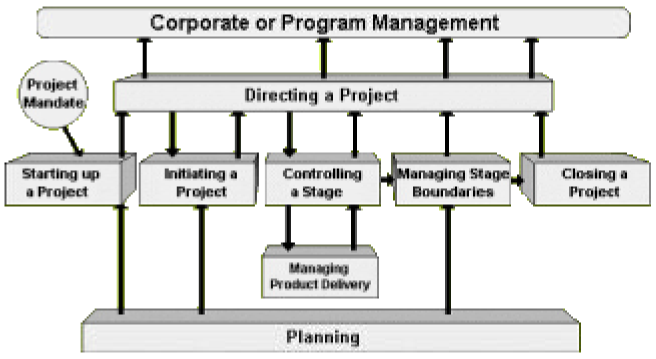
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001



Prince2

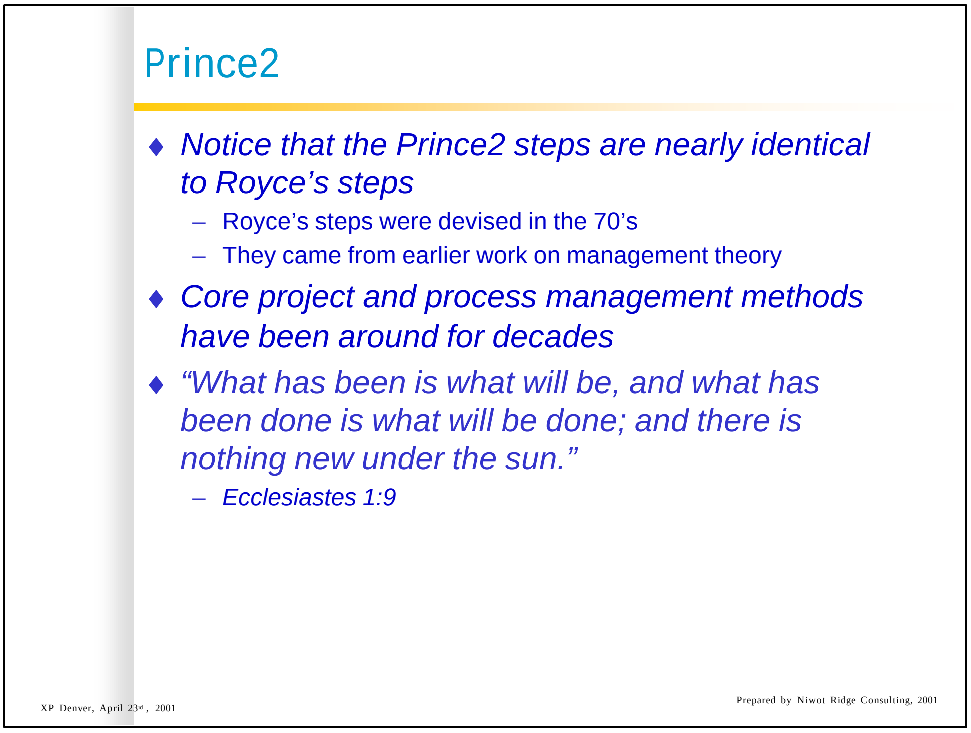
- *Is a process based approach … easily tailored, and scaleable method for the management of all types of projects.*
- *Sound familiar?*



XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Just a s a quick aside there is a standard project management method in the UK, Prince2, that uses similar words to the lightweight processes.

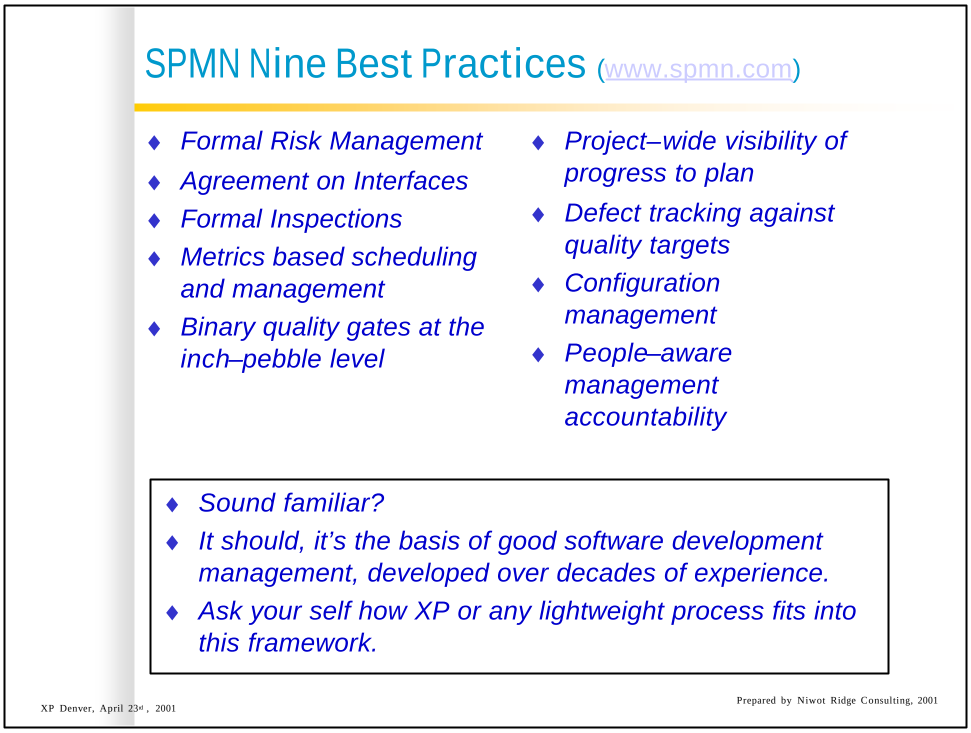


Prince2

- *Notice that the Prince2 steps are nearly identical to Royce's steps*
  - Royce's steps were devised in the 70's
  - They came from earlier work on management theory
- *Core project and process management methods have been around for decades*
- *"What has been is what will be, and what has been done is what will be done; and there is nothing new under the sun."*
  - *Ecclesiastes 1:9*

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001



SPMN Nine Best Practices (www.spmn.com)

- Formal Risk Management
- Agreement on Interfaces
- Formal Inspections
- Metrics based scheduling and management
- Binary quality gates at the inch–pebble level
- Project–wide visibility of progress to plan
- Defect tracking against quality targets
- Configuration management
- People–aware management accountability

- Sound familiar?
- It should, it's the basis of good software development management, developed over decades of experience.
- Ask your self how XP or any lightweight process fits into this framework.

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

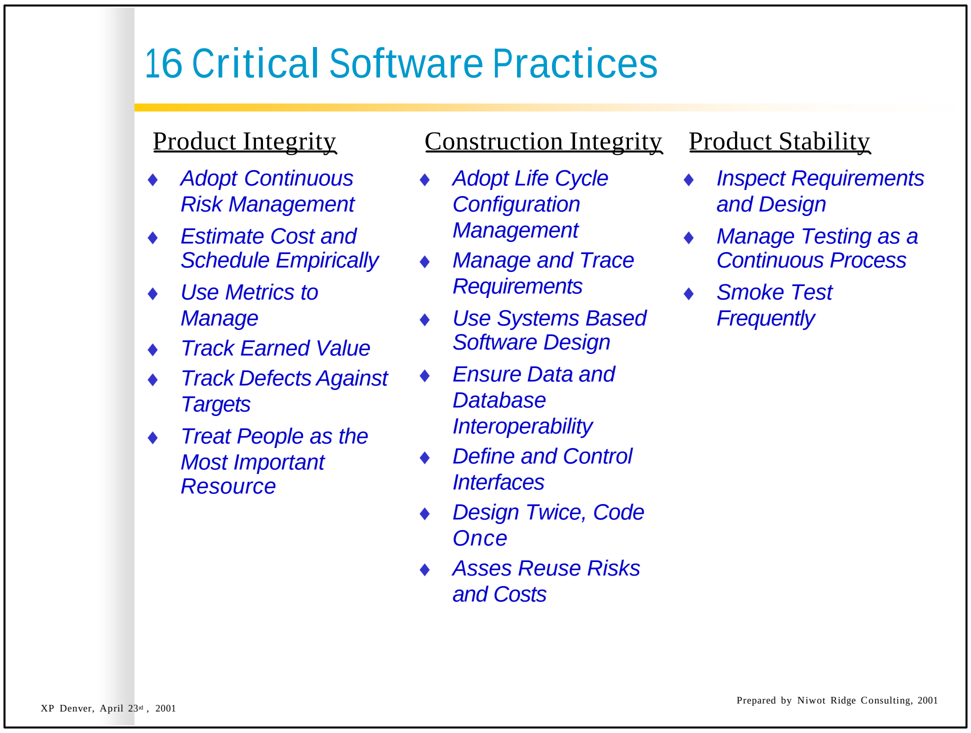
Another framework that we didn't talk about is the Software Program Managers Network Nine Best Practices.

This framework will be presented in the case study.

The Nine Best Practices is one of those "mind altering" experiences. Over the past three years of managing and improving a moderate sized software organization (27 folks), every time and I mean every time we got in trouble it was because we violated one of the Nine Best Practices.

One of the significant learnings in all this was to have some framework to fall back on when the project gets in trouble.

My recommendation is to have a higher level framework rather than a lower level framework. Remembering that programming is only one part of the development process, the framework you choose needs to address programming as well as the "product" and "business" aspects. The Nine Best Practices is one way. There are others.



16 Critical Software Practices

Product Integrity

- Adopt Continuous Risk Management
- Estimate Cost and Schedule Empirically
- Use Metrics to Manage
- Track Earned Value
- Track Defects Against Targets
- Treat People as the Most Important Resource

Construction Integrity

- Adopt Life Cycle Configuration Management
- Manage and Trace Requirements
- Use Systems Based Software Design
- Ensure Data and Database Interoperability
- Define and Control Interfaces
- Design Twice, Code Once
- Asses Reuse Risks and Costs

Product Stability

- Inspect Requirements and Design
- Manage Testing as a Continuous Process
- Smoke Test Frequently

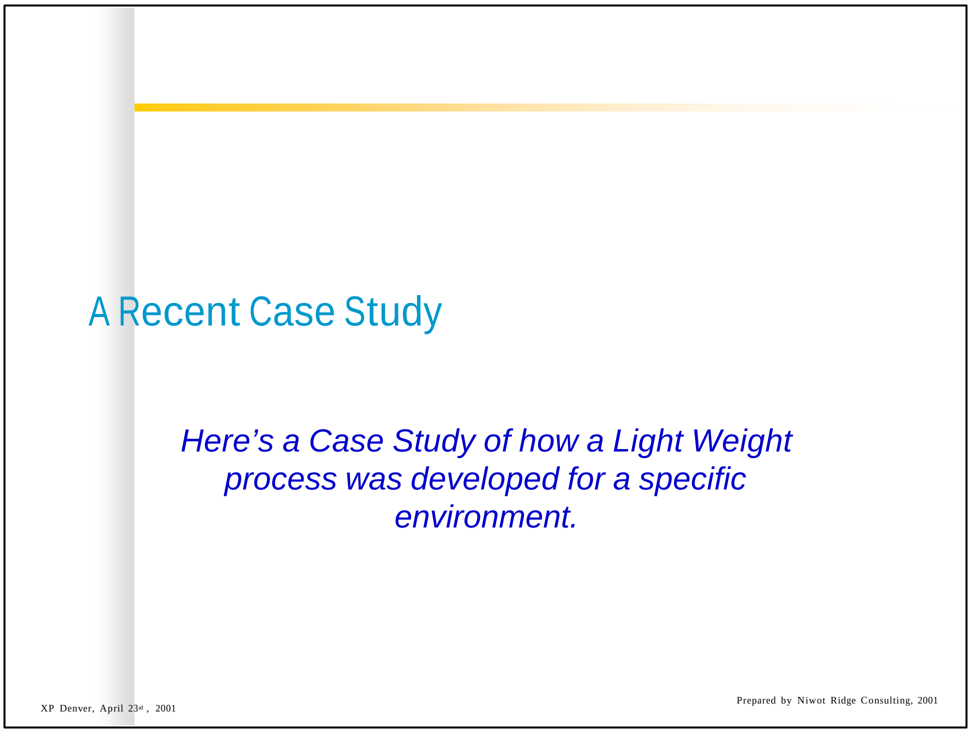
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Here's another framework for software development that is independent of any specific methodology. There are some aspects of this framework that are probably not very interesting to the Light Weight process domain, but there are others that are natural processes. Some of which are actually used in XP.

The point here is there is a wealth of knowledge and experience in "adapting" methods to the Light Weight process world that does not depend on the structured and sometimes dogmatic approaches of XP. Most of it is common sense. Most of it comes from the heuristics of process improvement process.

All of it can be adapted to specific project situations if the time and effort is taken to first understand what the specific needs are.



A Recent Case Study

Here's a Case Study of how a Light Weight process was developed for a specific environment.

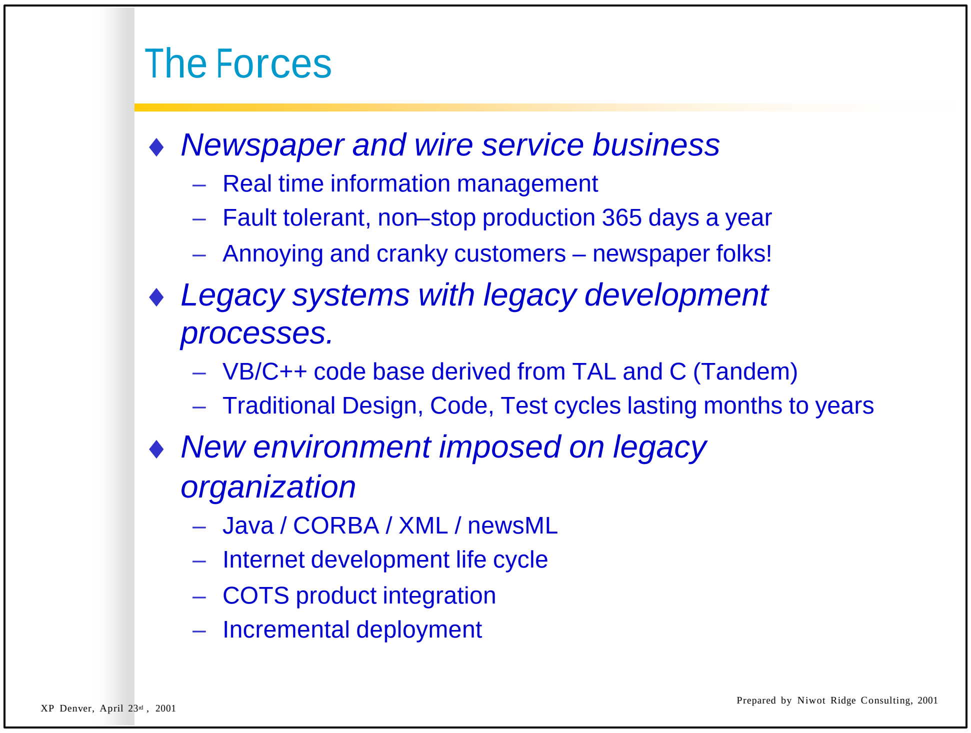
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

I've spent the last three years working at a newspaper software company in Sacramento.

Our firm provided high level system architecture, COTS integration strategies, and process improvement.

I will present this engagement as a case study for light weight process selection and deployment.



The Forces

- *Newspaper and wire service business*
  - Real time information management
  - Fault tolerant, non–stop production 365 days a year
  - Annoying and cranky customers – newspaper folks!
- *Legacy systems with legacy development processes.*
  - VB/C++ code base derived from TAL and C (Tandem)
  - Traditional Design, Code, Test cycles lasting months to years
- *New environment imposed on legacy organization*
  - Java / CORBA / XML / newsML
  - Internet development life cycle
  - COTS product integration
  - Incremental deployment

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

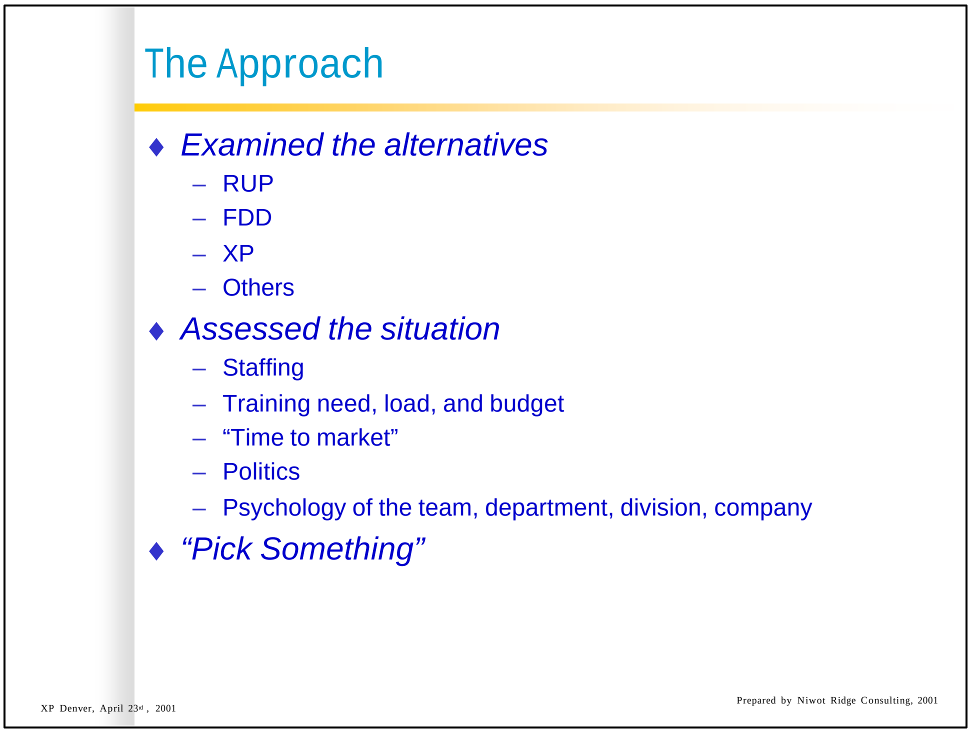
Here are the forces driving the team.

These forces are not unique, nor is the software or the development process unique.

What is unique is the unprecedented demand for products in the face of competition.

The newspaper software space provides very high reliability products to very conservative customers. These customers buy software to support the production of the newspaper. The major capital item in the paper is the press. A nice color press can cost up to $300M. Everyone in the newspaper is focused on running this press at maximum efficiency.

The software systems are all focused on producing metal plates to be mounted on the press.



The Approach

- *Examined the alternatives*
  - RUP
  - FDD
  - XP
  - Others
- *Assessed the situation*
  - Staffing
  - Training need, load, and budget
  - "Time to market"
  - Politics
  - Psychology of the team, department, division, company
- *"Pick Something"*

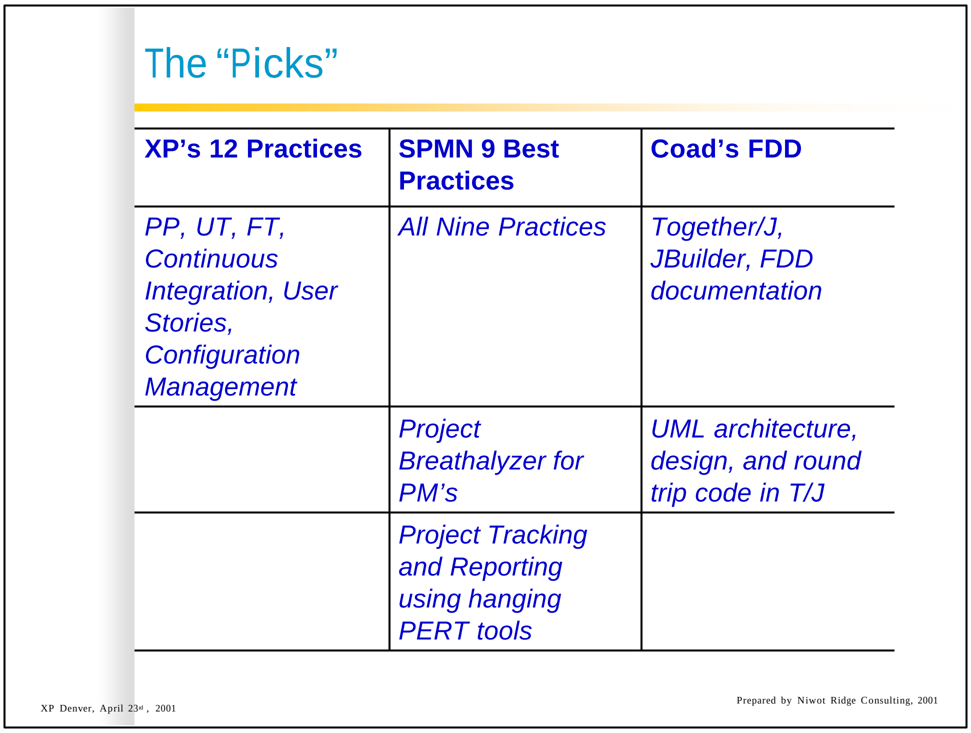
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

Our first approach was to adapt the problem to the methodology. We soon discovered that many aspects of the problem were not actually adaptable.

This was annoying at first, then we discovered that there are underlying "truths" to the development process.

- The development team has extensive experience in newspapers, functional decomposition development and mainframe processes. We couldn't simply "change the team."
- The customers had specific requirements that demanded some level of documentation. Fault tolerant real time systems are documentation centric, since many of the components come from outside the team (COTS products).
- The embedded tools of newspaper production are specified by the users not the vendors. Adobe, Quark, and others are standards in the industry. How these products work is outside our control.
- Newspapers have specific time lines for the acquisition and installation of products. These are built around trade shows. The international trade shows are the major milestones of any newspaper software company. Missing a trade show deadline means disaster for a vendor.



The "Picks"

XP's 12 Practices	SPMN 9 Best Practices	Coad's FDD
PP, UT, FT, Continuous Integration, User Stories, Configuration Management	*All Nine Practices*	*Together/J, JBuilder, FDD documentation*
	Project Breathalyzer for PM's	*UML architecture, design, and round trip code in T/J*
	Project Tracking and Reporting using hanging PERT tools	

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

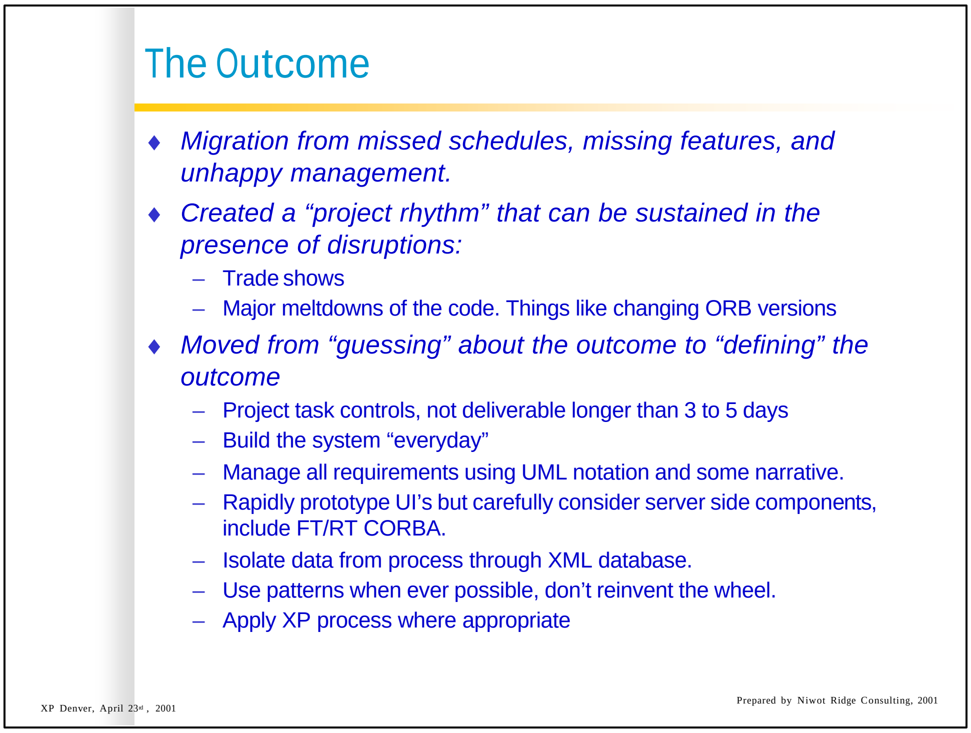
So we went on a shopping trip, starting with the Nine Best Practices.

The SPMN has a clever Project Breathalyzer test that the managers used to "test" the project's health.

We tried several UML tools and ended up with Together/J and Coad's FDD design and development process.

XP was examined and we culled out several practices and applied then to the project:

- PP – we used pairs of developers for specific tasks. On other tasks PP did not work for a variety of reasons.
- Continuous integration was first applied using Steve McConnell's "smoke test" strategy. It took some effort to get the system to build an run every day, but the payoffs are obvious.
- User Stores – Use Cases and User Stories were traditional ways of gathering requirements. We simply formalized them in UML.
- CM – was always a problem. By forcing daily builds and some test, CM also had to work automatically



The Outcome

- *Migration from missed schedules, missing features, and unhappy management.*
- *Created a "project rhythm" that can be sustained in the presence of disruptions:*
  - Trade shows
  - Major meltdowns of the code. Things like changing ORB versions
- *Moved from "guessing" about the outcome to "defining" the outcome*
  - Project task controls, not deliverable longer than 3 to 5 days
  - Build the system "everyday"
  - Manage all requirements using UML notation and some narrative.
  - Rapidly prototype UI's but carefully consider server side components, include FT/RT CORBA.
  - Isolate data from process through XML database.
  - Use patterns when ever possible, don't reinvent the wheel.
  - Apply XP process where appropriate

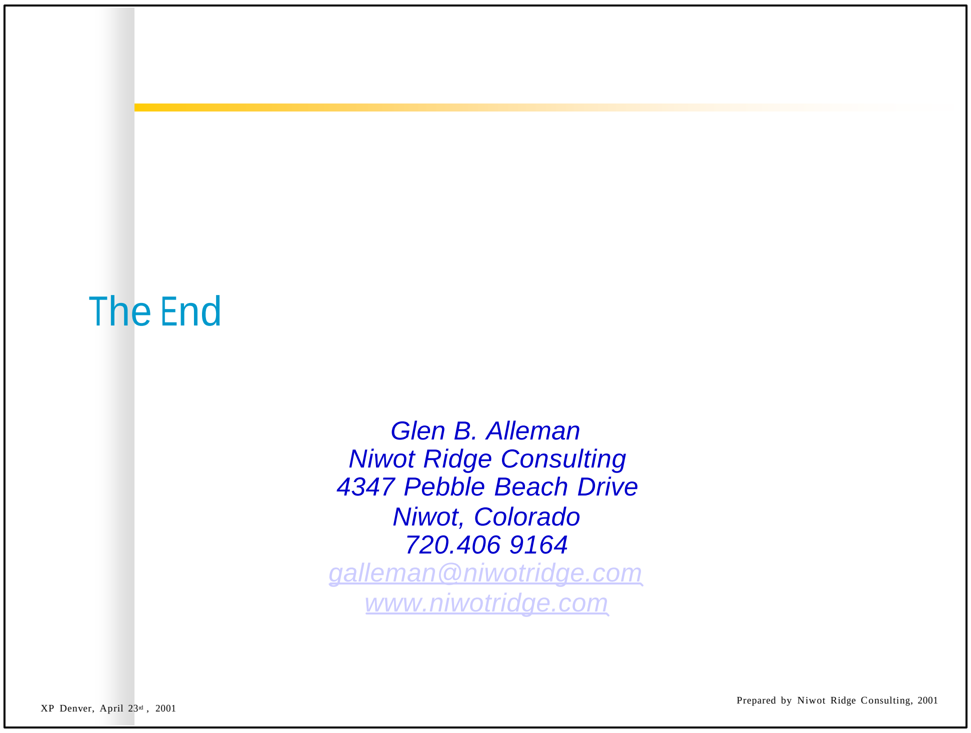
XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001

The identification of the needs of the project first drove the selection process. this was followed by the "use" of the various processes and the discovery of what worked and what didn't.

The details of this 3 years cycle are left to another time, but the learnings of the group are significant.

- Question each assumption of a proposed method. What may work in one environment may not work in another.
- Don't assume that since it hasn't been tried it WILL work. Assume that since it hasn't been tried, discussed in the literature and documented somewhere, it WON'T work. Experimenting with our employers money is sporty business in these post-internet times.
- This doesn't mean we're not open to new ideas, just that what ever process is adapted needs to be examined in light of our past experiences and the appropriateness of the current situation.
- Process improvement is a continuous improvement process in itself. Keep good records, track progress against goals, define the metric upfront so you can know when things are working.



The End

Glen B. Alleman
Niwot Ridge Consulting
4347 Pebble Beach Drive
Niwot, Colorado
720.406 9164
galleman@niwotridge.com
www.niwotridge.com

XP Denver, April 23rd, 2001

Prepared by Niwot Ridge Consulting, 2001