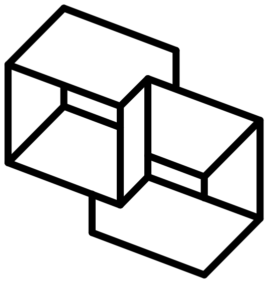
Questioning Extreme Programming



Pete McBreen, McBreen.Consulting
pete@mcbreen.ab.ca

The arguments for and against Extreme Programming are many and varied

Why is there a shift towards the so-called *Agile Methodologies*?

Extreme Programming is an answer to what question?

Under what conditions is XP successful?

Under what conditions are alternative processes more appropriate?

Is Extreme Programming for you?

Traditional software development approaches are not always successful

An extremist view of traditional software development is *Crummy Software Late*

Duff O'Melia of RoleModel Software asks some hard questions

- How do you know your code works?
- Have you ever been afraid to change code?
- How do you know your design is right?
- What happens if your star programmer is hit by a bus?

Something is rotten in the state of software development

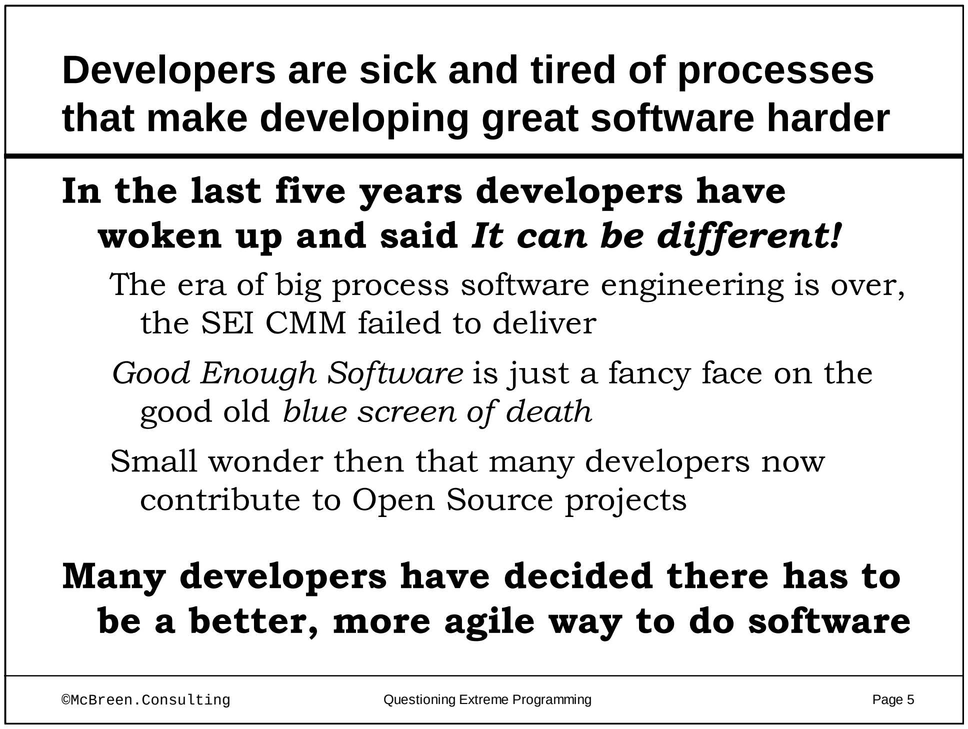
The *Dilbert Phenomenon* is a very strong hint that there are serious problems

- *Pointy Haired Boss* jokes are not funny any more

Organizations are completely dependent on software, but coding is a lousy job

- The Suits make the decisions, but the techs get blamed for failing to deliver the code

Even the promise of dot com riches turned out to be a *death march* to nowhere



Developers are sick and tired of processes that make developing great software harder

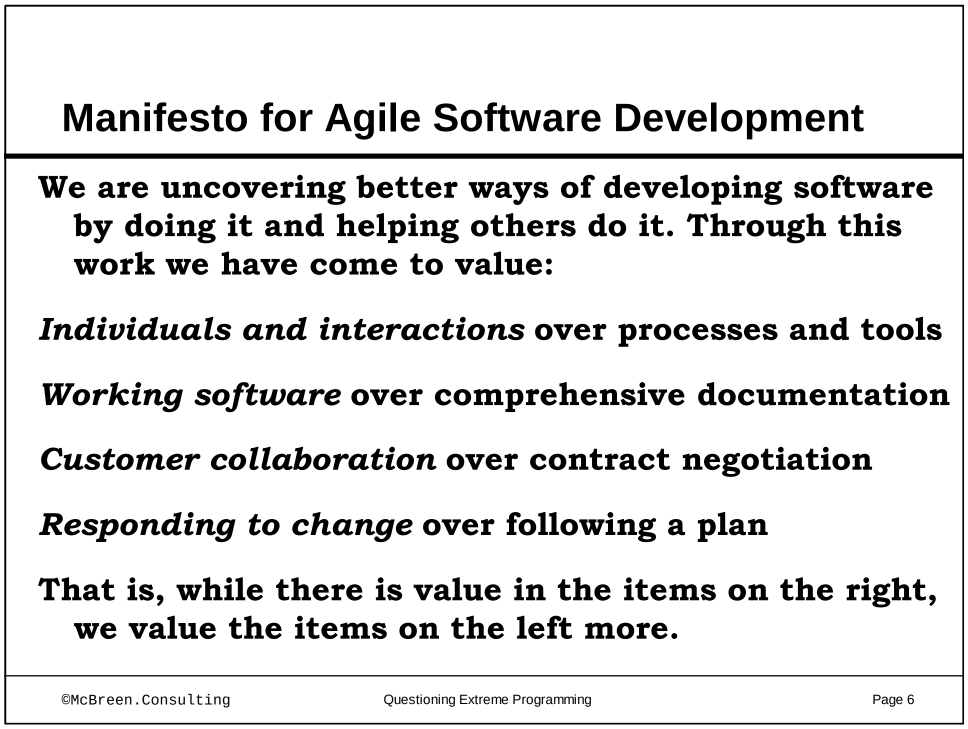
In the last five years developers have woken up and said *It can be different!*

The era of big process software engineering is over, the SEI CMM failed to deliver

Good Enough Software is just a fancy face on the good old *blue screen of death*

Small wonder then that many developers now contribute to Open Source projects

Many developers have decided there has to be a better, more agile way to do software



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

***Individuals and interactions* over processes and tools**

***Working software* over comprehensive documentation**

***Customer collaboration* over contract negotiation**

***Responding to change* over following a plan**

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

The methodologies that contributed to the manifesto are quite diverse

Adaptive Software Development

Crystal methods

Dynamic Systems Development Method

Extreme Programming

Feature Driven Development

Pragmatic Programming

Scrum

Adaptive Software Development was created by Jim Highsmith

ASD is based on the ideas of Complexity and Emergence

- Managing the *workstate* rather than the workflow is the key difference
- Collaboration and self-organization enable *emergence*
- Describes a new lifecycle Speculate, Collaborate, Learn

ASD is intended for extreme projects that push the limits of what is possible

The Crystal methods were developed by Alistair Cockburn

The Crystal methods apply the idea that all processes are situational

They are a set of human scale, tolerant processes tuned for different projects

- Trust people to be good citizens
- Use incremental development so the team gets practice at delivering, and,
- Are barely sufficient to avoid overloading people with process.

Dynamic Systems Development Method is modernized Rapid Application Development

DSDM is about controlling and managing the rapid delivery of applications

DSDM uses empowered teams to deliver quality applications through the use of timeboxed development

- Do enough and no more,
- Use active user involvement to ensure fitness for business purpose, and
- Iterative development to converge on an accurate solution.

Extreme Programming: the revenge of the programmers :-)

A set of synergistic practices that maximize the amount of work not done

A highly incremental process that allows an on-site customer to steer the project

- Based on 5 core values Communication, Simplicity, Feedback, Courage and Respect
- Uses pair programming for all production code
- Does the simplest thing that could possibly work
- Preaches the idea of Test Driven Development

©McBreen.Consulting Questioning Extreme Programming Page 11

Feature Driven Development was designed by Peter Coad

FDD aims to deliver frequent, tangible, working results

FDD is unique among the Agile methods in promoting the use of CASE tools

- Uses Feature Teams lead by a Chief Programmer
- Has a heavy focus on modeling and archetypes
- Supported by CASE tool that is also a Java IDE
- Design By Feature, Build By Feature using a two week cycle

©McBreen.Consulting Questioning Extreme Programming Page 12

Pragmatic Programming is based on a book written by Andy Hunt and Dave Thomas

Pragmatic Programming addresses the facing issues teams of 2 or 3 developers

More concerned with individual mastery of the craft of software development

- Individual craftsmanship as a foundation for overall team success
- Focuses on effective, appropriate use of tools
- Encourages a seamless approach - specification and implementation as different aspects of the same process

Scrum was created by Ken Schwaber, Jeff Sutherland and Mike Beedle

Scrum is an empirical process for managing software product development

It reintroduces flexibility, adaptability and productivity into systems development

- Work can and should be an ennobling experience
- Projects are divided into sprints to allow developers to focus on delivery
- Empirical management using frequent, first hand observations and daily scrum meetings

What is so different about these Agile approaches?

All put people and the interaction between people as the main focus of attention

All embrace the idea that dealing with partial knowledge is the key to success

Their response to schedule pressure is to prioritize and focus on early delivery

They require extensive involvement by the project sponsors and users

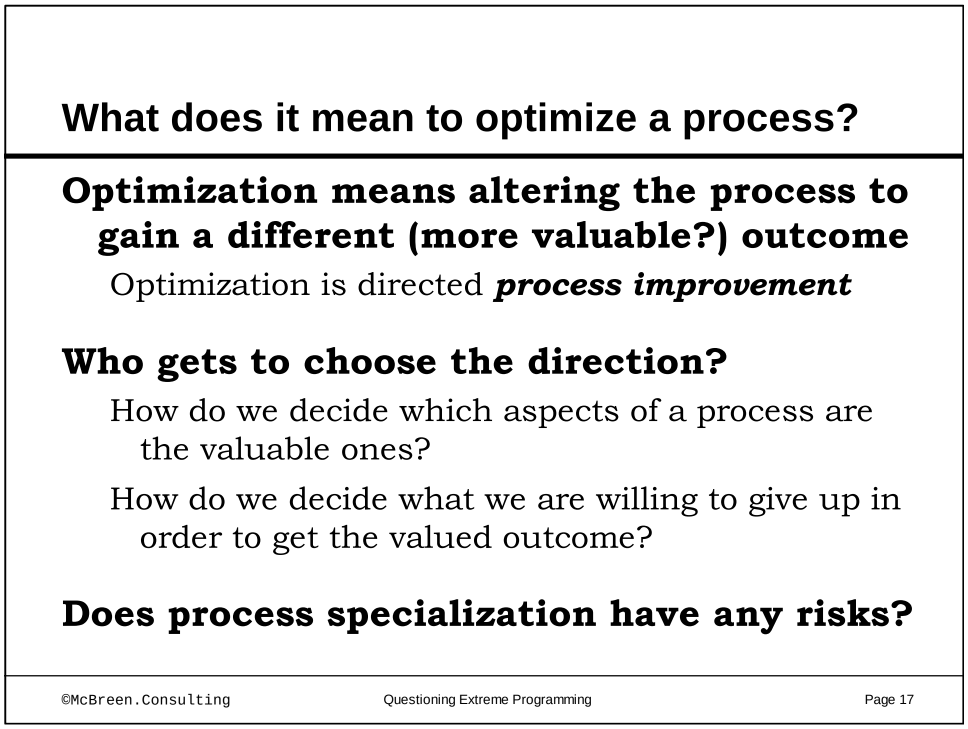
The Agile approaches are changing the way that software development is done

The Agile approaches are changing the conversation about software development

Agile shifted our attention to small teams incrementally delivering quality software

Whether this is a good thing or not is still open to question

In optimizing our process towards the Agile approach, what are we giving up?



What does it mean to optimize a process?

Optimization means altering the process to gain a different (more valuable?) outcome

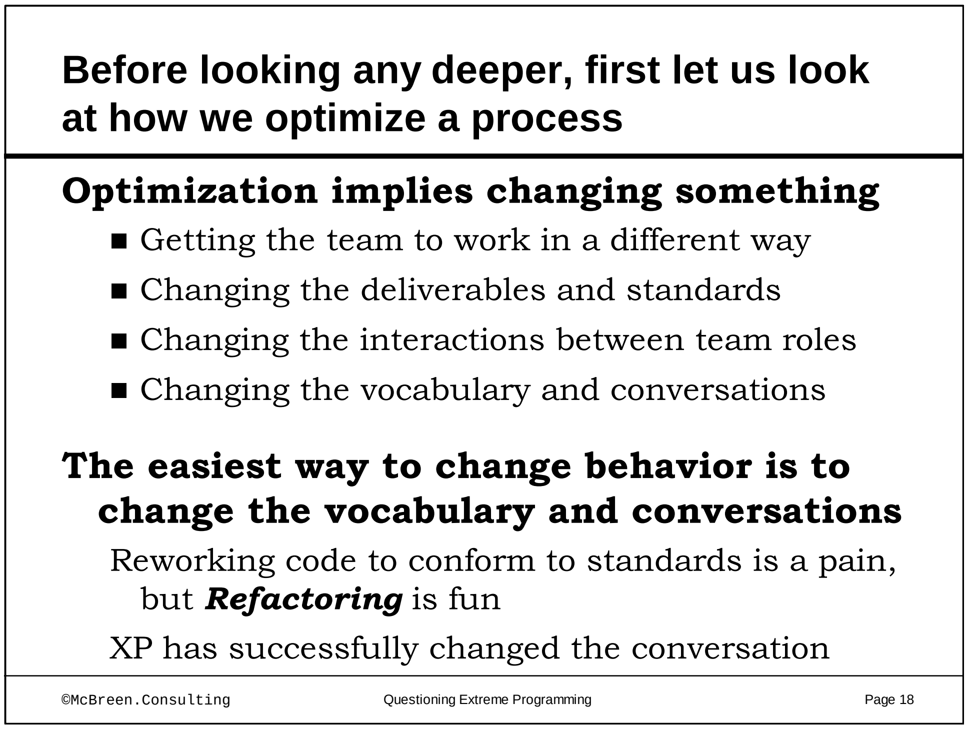
Optimization is directed ***process improvement***

Who gets to choose the direction?

How do we decide which aspects of a process are the valuable ones?

How do we decide what we are willing to give up in order to get the valued outcome?

Does process specialization have any risks?



Before looking any deeper, first let us look at how we optimize a process

Optimization implies changing something

- Getting the team to work in a different way
- Changing the deliverables and standards
- Changing the interactions between team roles
- Changing the vocabulary and conversations

The easiest way to change behavior is to change the vocabulary and conversations

Reworking code to conform to standards is a pain, but ***Refactoring*** is fun

XP has successfully changed the conversation

What does Extreme Programming optimize? Hint: Look at the project outcomes

Developer enjoyment ☺

Entire team agrees this is the best project they've ever been on - Ron Jeffries speaking about C3

Predictable, sustained and sustainable pace

The project can go at any pace it wants to. The point is steering, not going full speed

Maximizing the team's tacit knowledge

The best way to keep the knowledge of a system alive and well is to maintain the continuity of the development team. - Robert Martin

What else might you want to optimize for?

The business as usual answer is *everything* but it doesn't work out that way

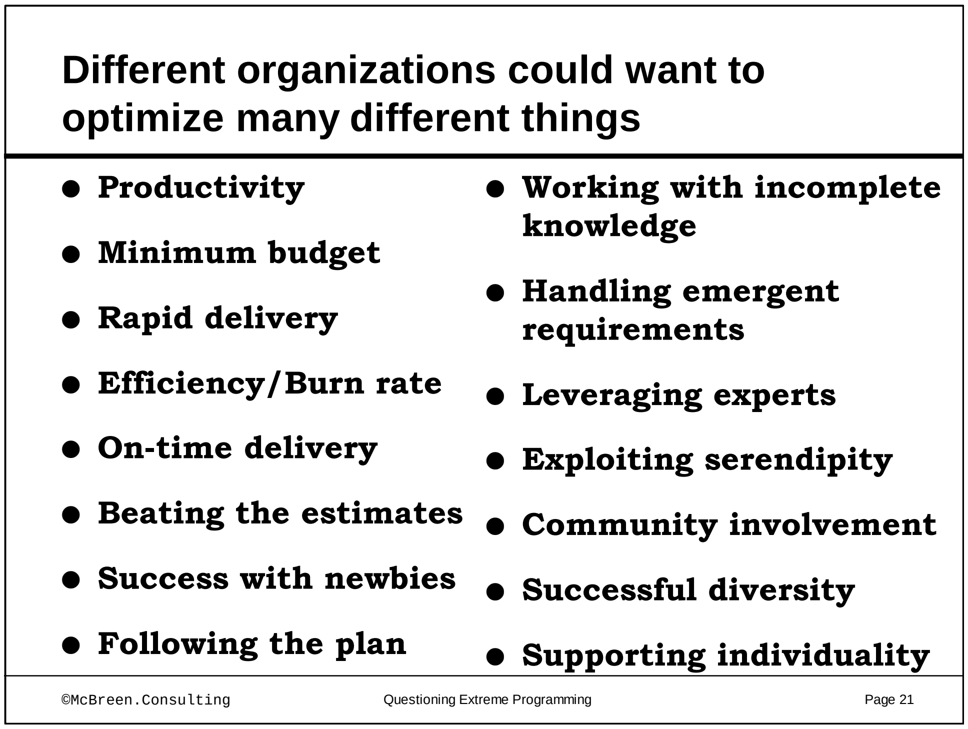
When everything is important, the item that gets paid attention to seems to be pseudo-random

Faster, Better, Cheaper is not realistic

On Time, On Budget, On Mars - pick two

As Jim Highsmith suggests, choose an appropriate *Mission Profile*

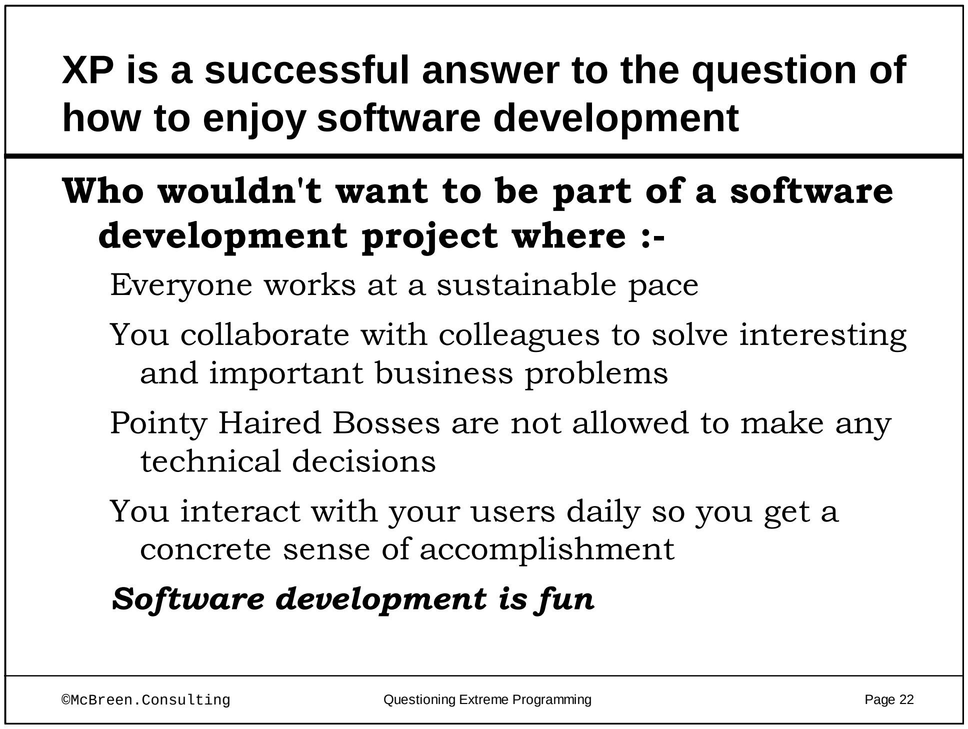
Then optimize to excel at the chosen outcome



Different organizations could want to optimize many different things

- Productivity
- Minimum budget
- Rapid delivery
- Efficiency/Burn rate
- On-time delivery
- Beating the estimates
- Success with newbies
- Following the plan
- Working with incomplete knowledge
- Handling emergent requirements
- Leveraging experts
- Exploiting serendipity
- Community involvement
- Successful diversity
- Supporting individuality

©McBreen.Consulting Questioning Extreme Programming Page 21



XP is a successful answer to the question of how to enjoy software development

Who wouldn't want to be part of a software development project where :-

Everyone works at a sustainable pace

You collaborate with colleagues to solve interesting and important business problems

Pointy Haired Bosses are not allowed to make any technical decisions

You interact with your users daily so you get a concrete sense of accomplishment

Software development is fun

©McBreen.Consulting Questioning Extreme Programming Page 22

The greatest pitfall of Extreme Programming is that it sounds too good to be true

The evidence to date is that XP really and truly works

XP was not a corporate initiative, it was just a couple of people with a vision

- In five short years it has drastically reshaped software development
- Even Rational now supports XP with a plug-in for the Rational Unified Process (RUP)

©McBreen.Consulting Questioning Extreme Programming Page 23

So what are the pre-conditions for XP?

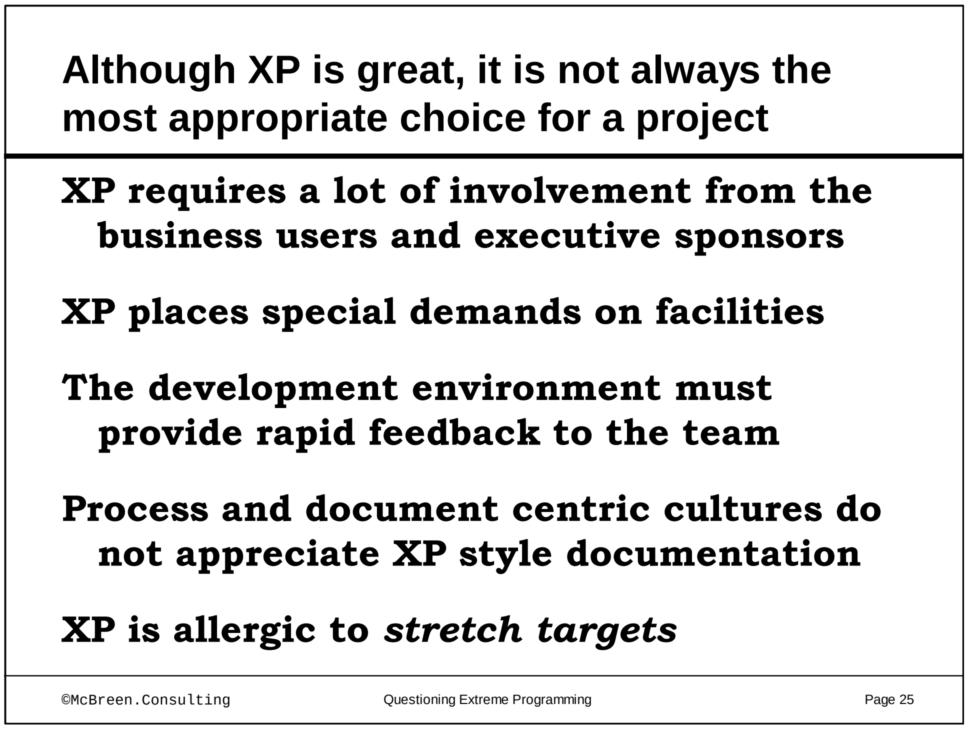
A small co-located team

Knowledgeable domain experts who make project direction decisions quickly

A *Point Haired Boss* free zone

An organization where delivered results count more than following the plan

©McBreen.Consulting Questioning Extreme Programming Page 24



Although XP is great, it is not always the most appropriate choice for a project

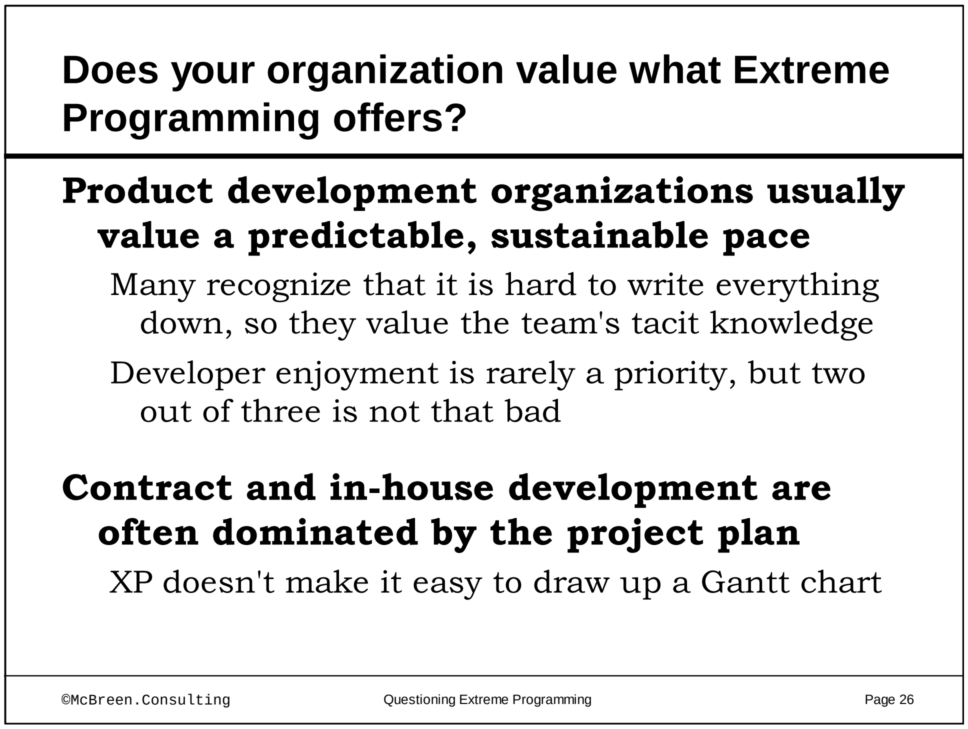
XP requires a lot of involvement from the business users and executive sponsors

XP places special demands on facilities

The development environment must provide rapid feedback to the team

Process and document centric cultures do not appreciate XP style documentation

XP is allergic to *stretch targets*



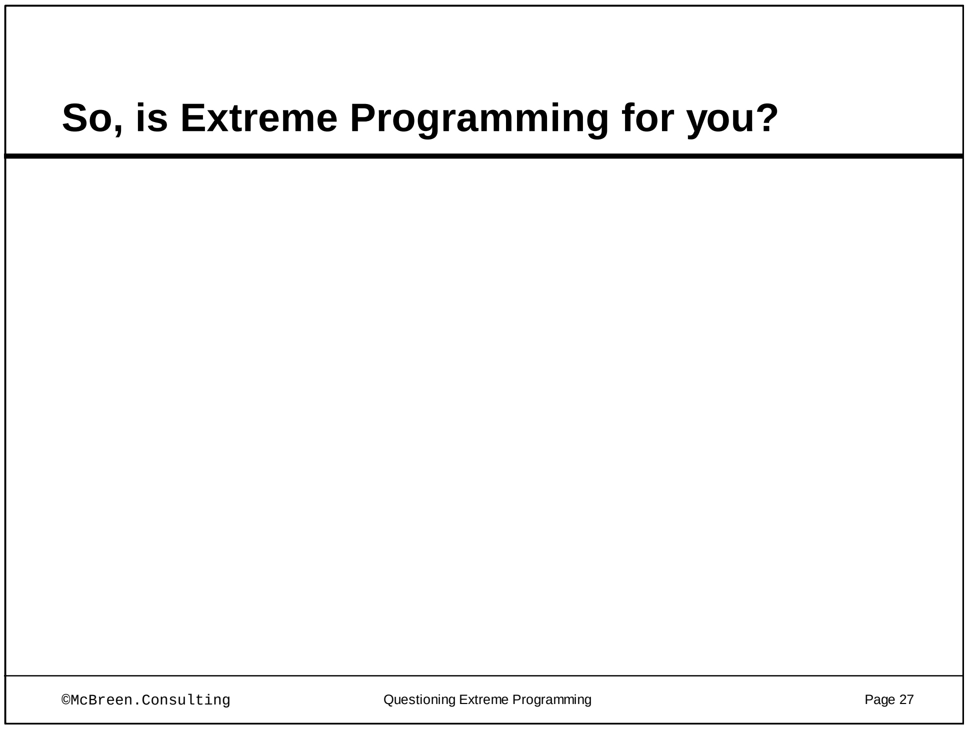
Does your organization value what Extreme Programming offers?

Product development organizations usually value a predictable, sustainable pace

- Many recognize that it is hard to write everything down, so they value the team's tacit knowledge
- Developer enjoyment is rarely a priority, but two out of three is not that bad

Contract and in-house development are often dominated by the project plan

- XP doesn't make it easy to draw up a Gantt chart



So, is Extreme Programming for you?