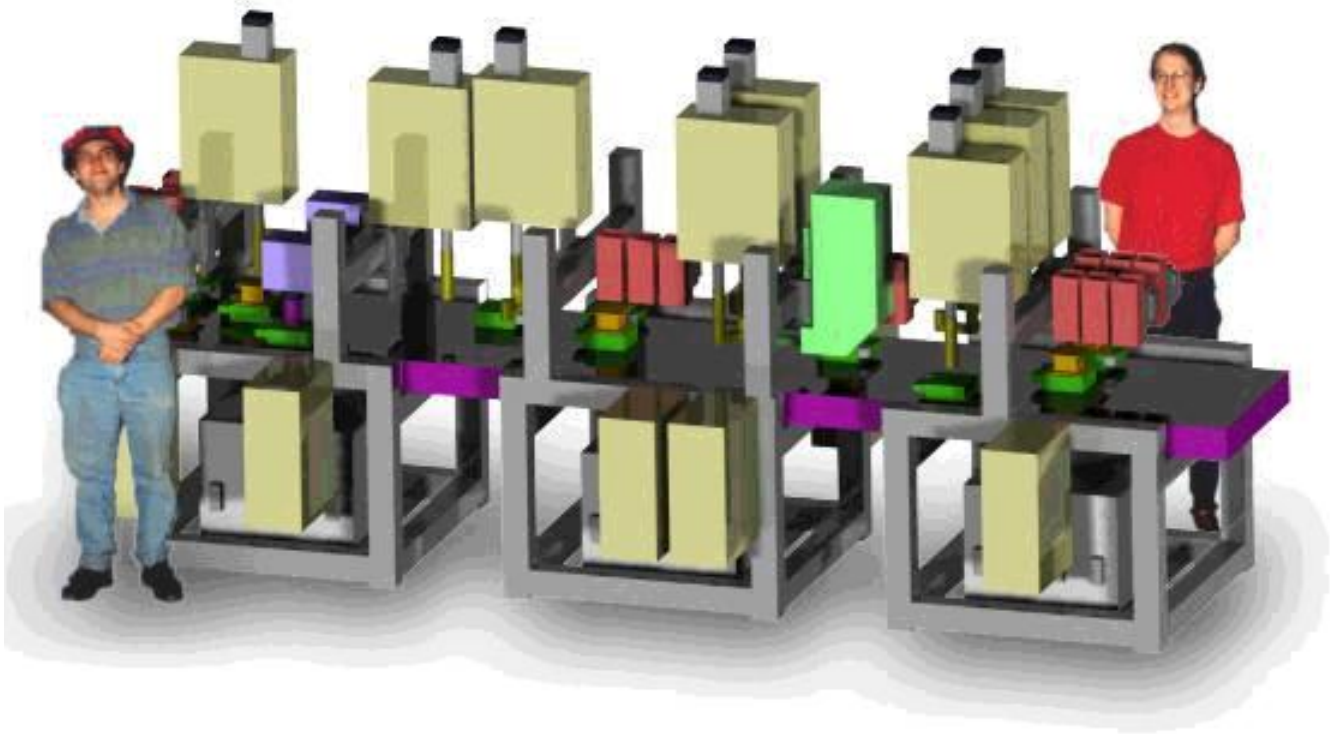
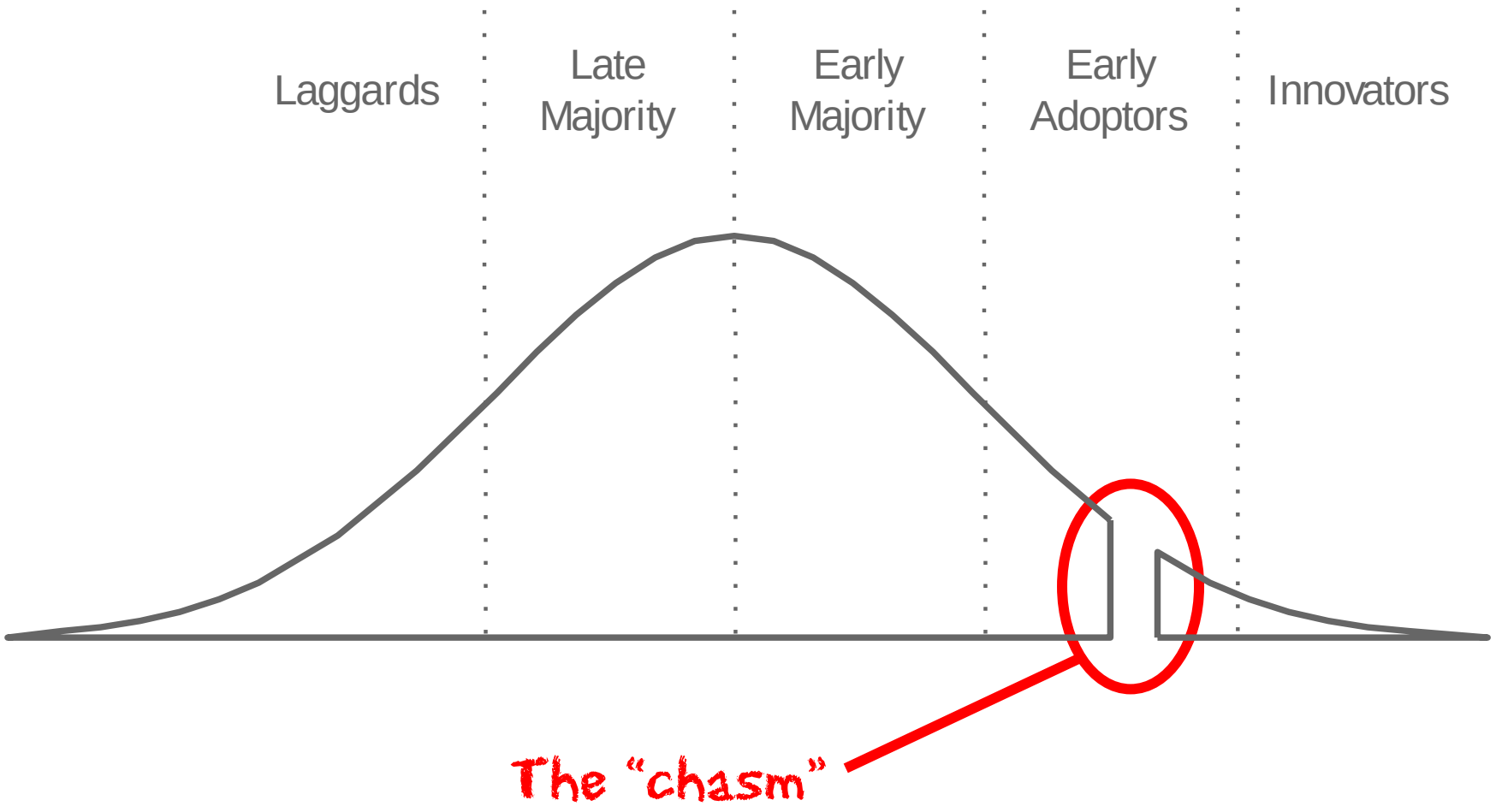
LonsdaleSystems

The Importance of Being Agile



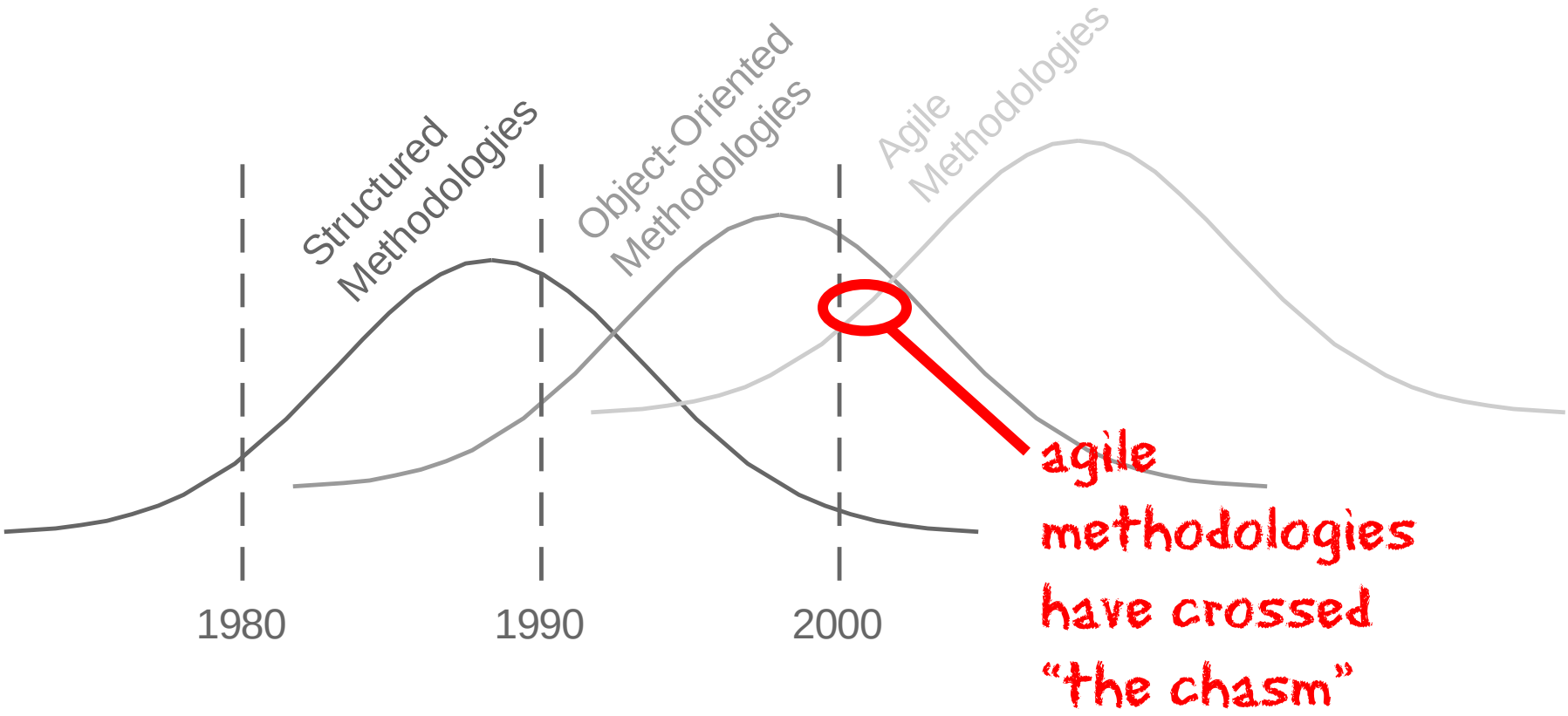
LonsdaleSystems

Technology Adoption



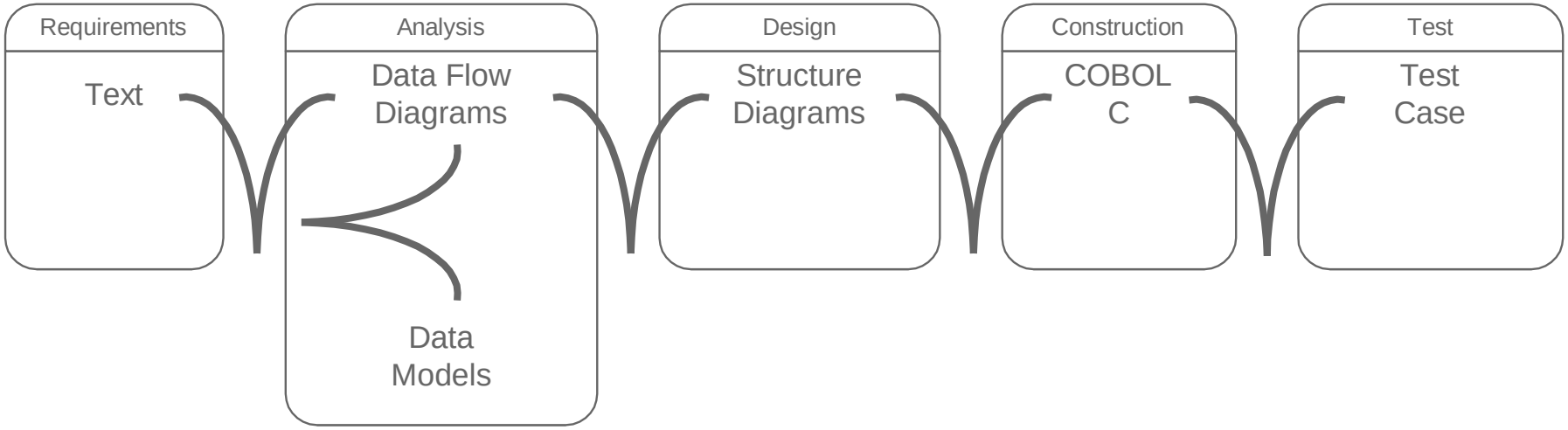
LonsdaleSystems

Methodology Waves



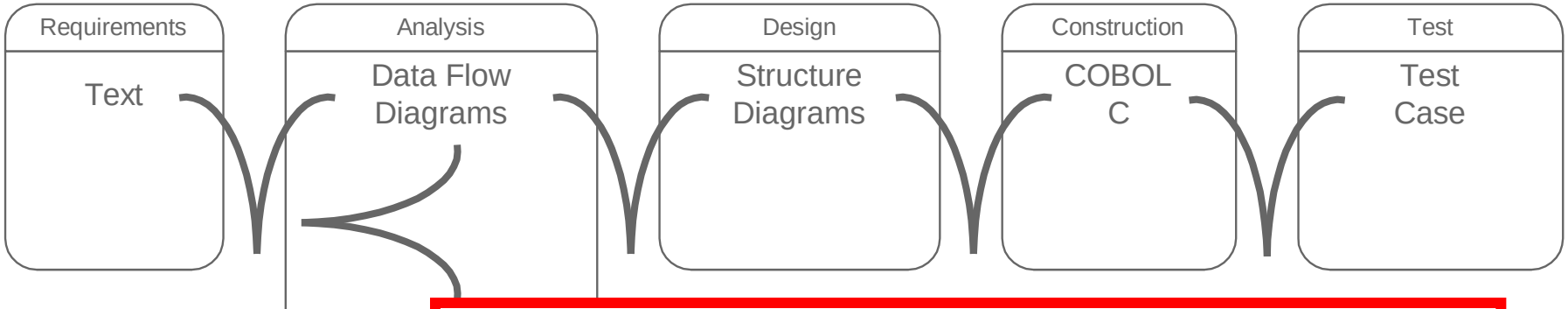
LonsdaleSystems

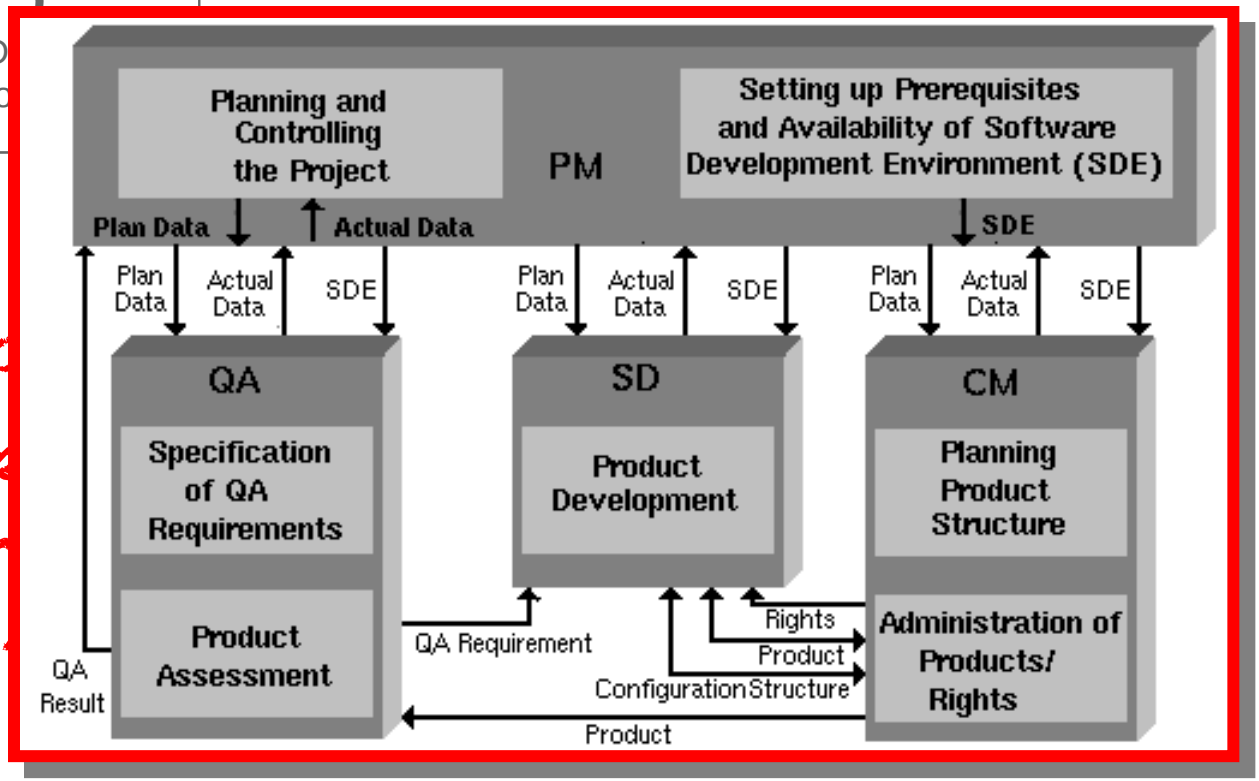
Structured Methodologies



- Transformation of work products
- Waterfall life-cycle addresses requirements and architecture risks late in the project
- Attempt to "predict" all necessary tasks

Structured Methodologies

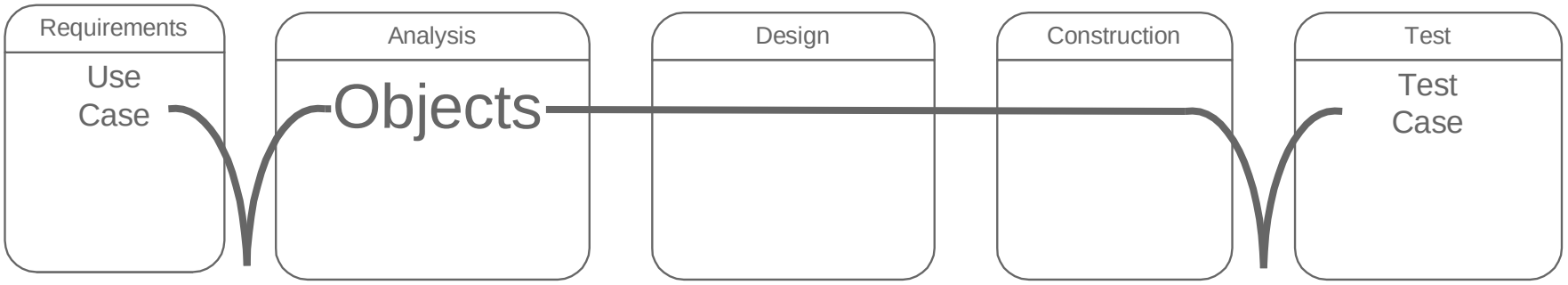




- Transfo
- Waterfa
  and arch
- Attempt

LonsdaleSystems

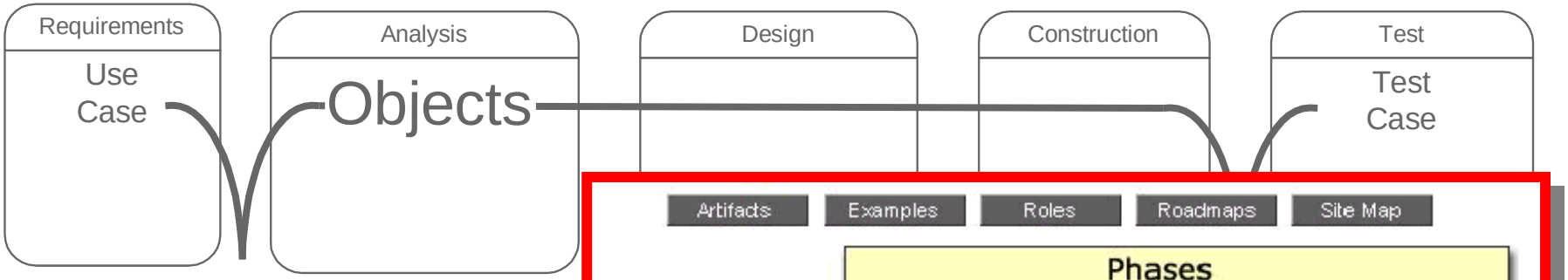
Object-Oriented Methodologies



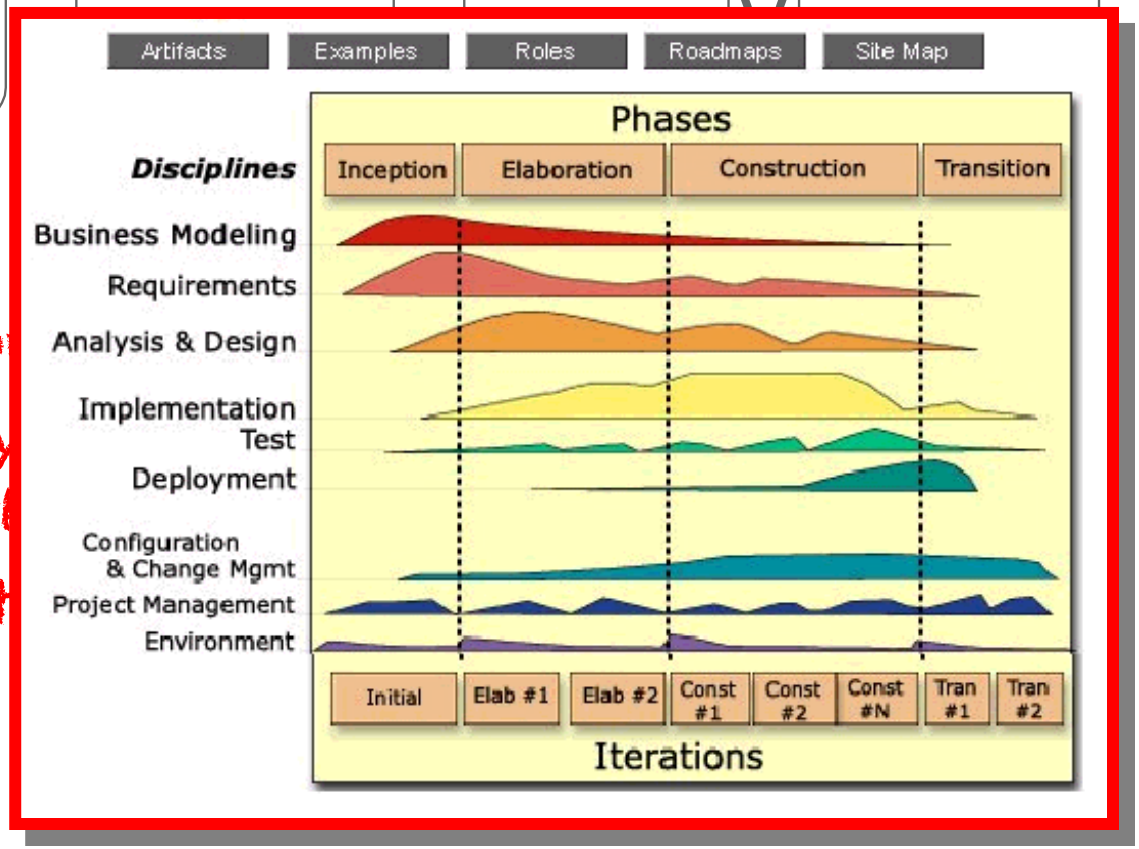
- Evolution of work products
- Iterative life-cycle addresses requirements and architecture risks early in the project
- Still attempts to "predict" all necessary tasks

LonsdaleSystems

Object-Oriented Methodologies



- Evolution of wo
- Iterative life-cy
  architecture risk
- Still attempts t



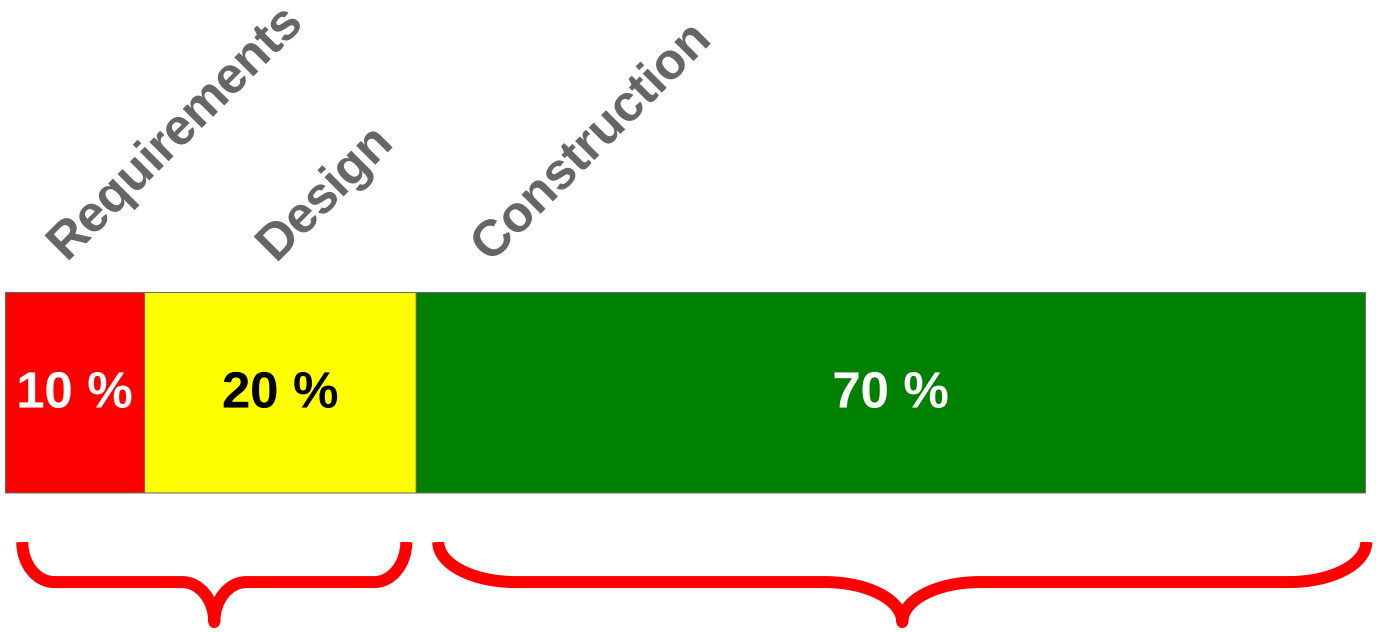
LonsdaleSystems

Agile Motivation

"Neither is it the same as manufacturing or digging big holes in the ground"

Say after me... "software development is not the same as building!"

Building

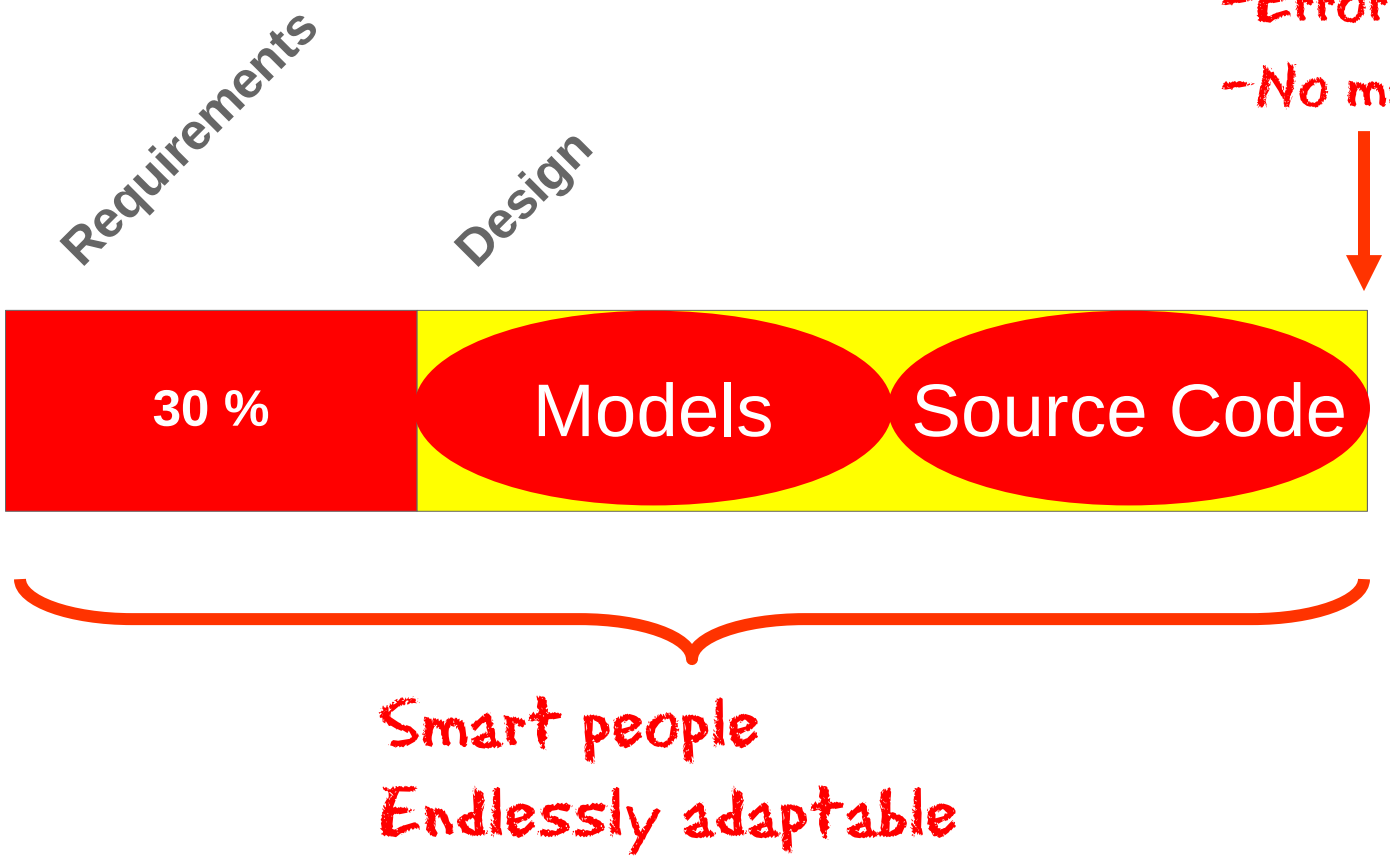

Requirements
Design
Construction
10 %
20 %
70 %
Smart people
Must obey
physical laws
Less smart people
Expensive
Time-consuming
Materials required

LonsdaleSystems

Software Development

Construction is

- Free
- Instant
- Error free
- No material required



LonsdaleSystems

Software Development

Construction is

-Free

-Instant

Agile methodologies focus on the uniqueness of software development...

Smart people

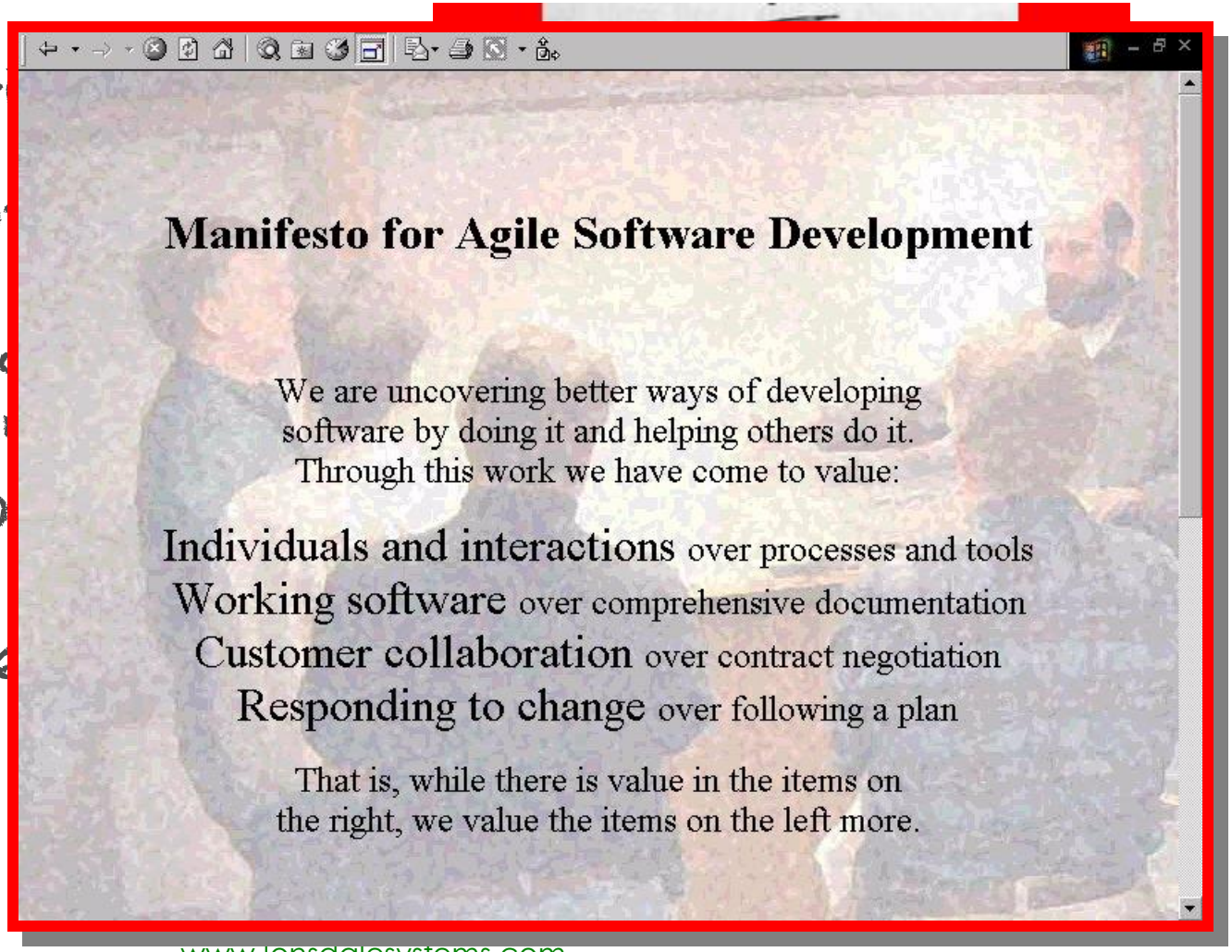
Endlessly adaptable

Agile Methodologies

- "Agile"
  - Characterised by quickness, lightness, and ease of movement; nimble
  - Mentally quick or alert: an agile mind
- Agile methodologies attempt to be "adaptive" rather than "predictive"

Agile Methodologies

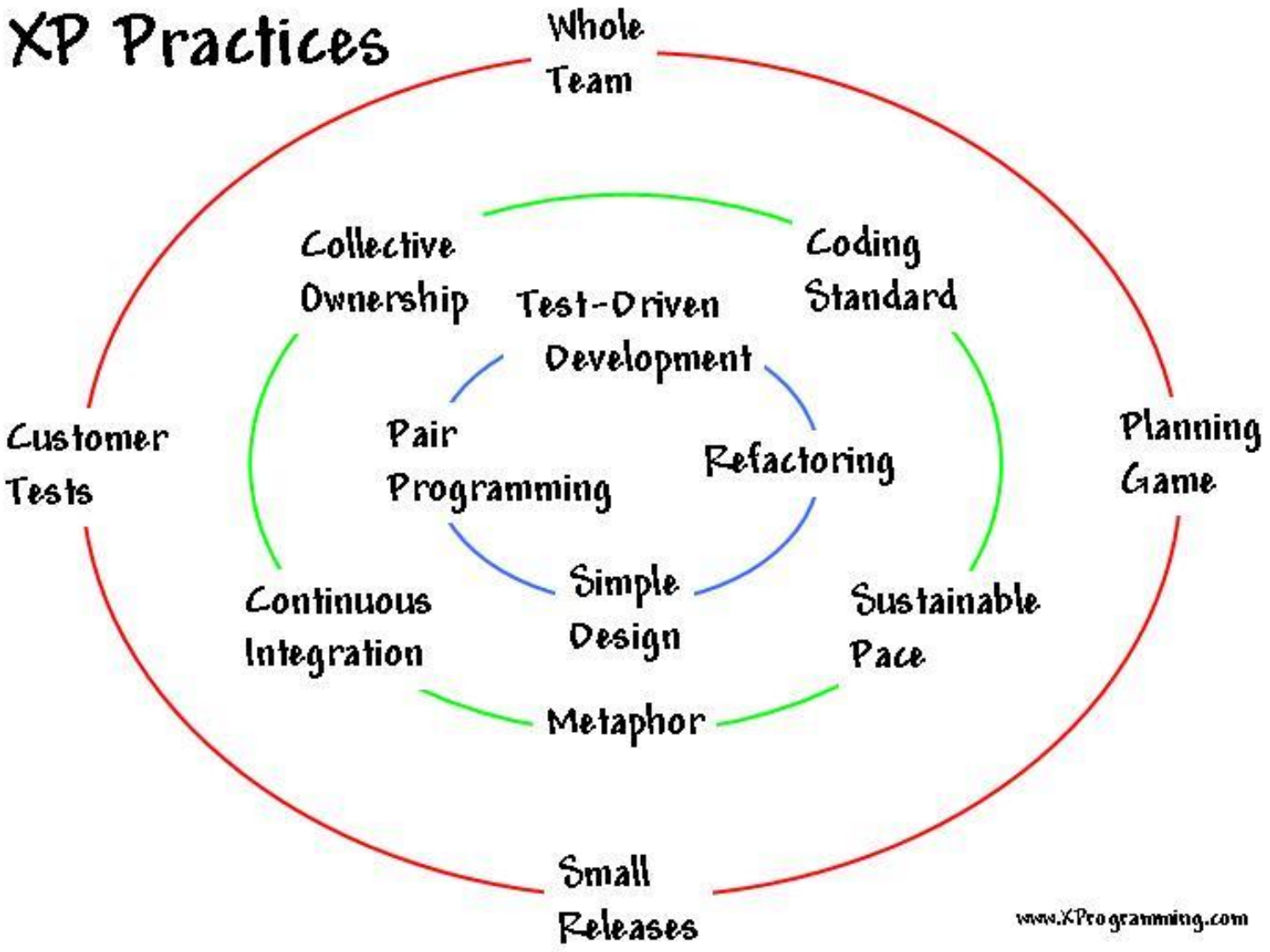
- "Agile"
  - Character
    quickness,
    and ease o
    nimble
  - Mentally
    an agile mi
- Agile metho
  attempt to
  "adaptive" r
  "predictive"



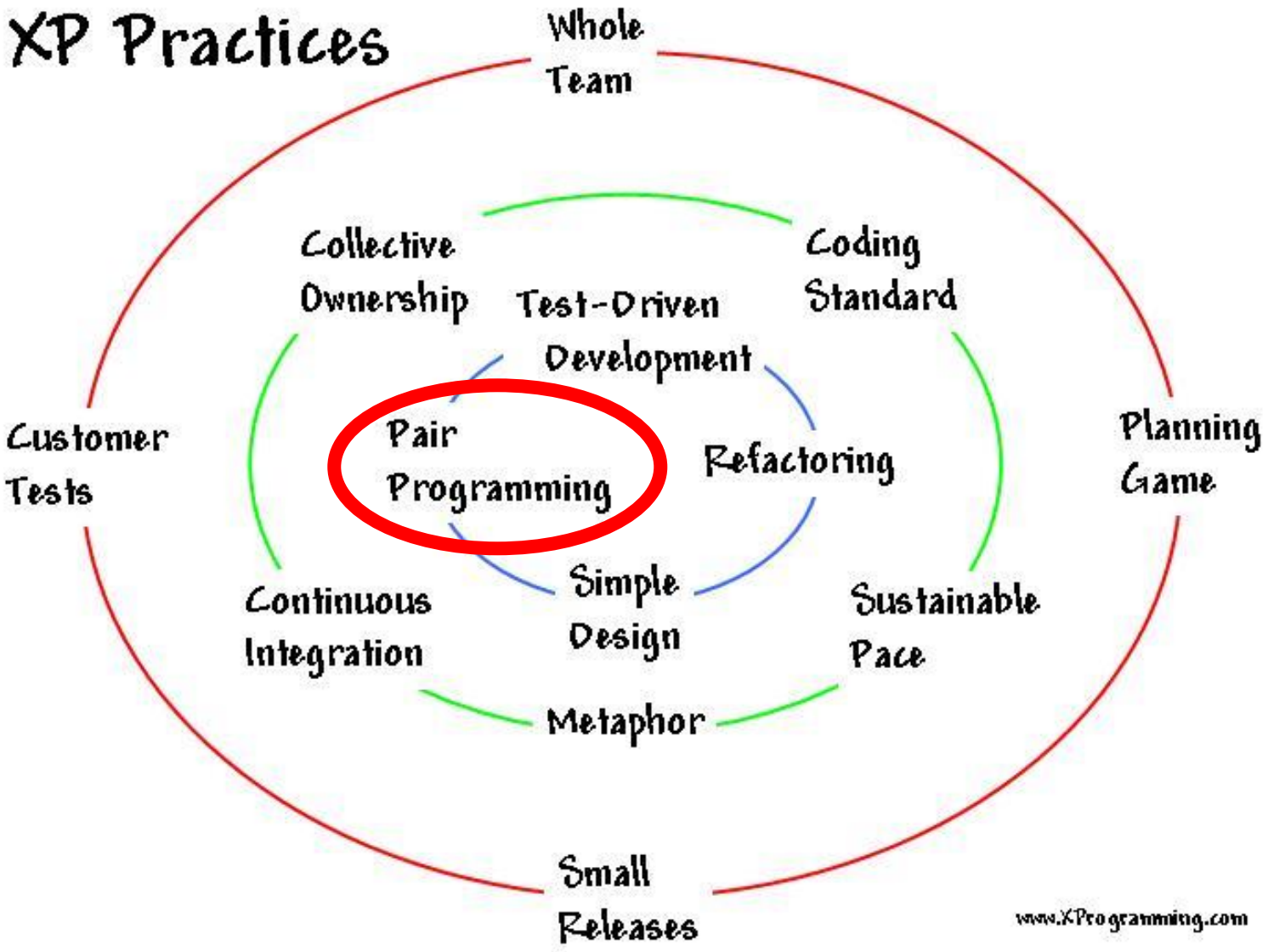
Agile Methodologies

- eXtreme Programming (XP)
- Crystal
- Scrum
- Feature Driven Development (FDD)
- Dynamic System Development Method (DSDM)

eXtreme Programming



eXtreme Programming



eXtreme Programming

XP Practices

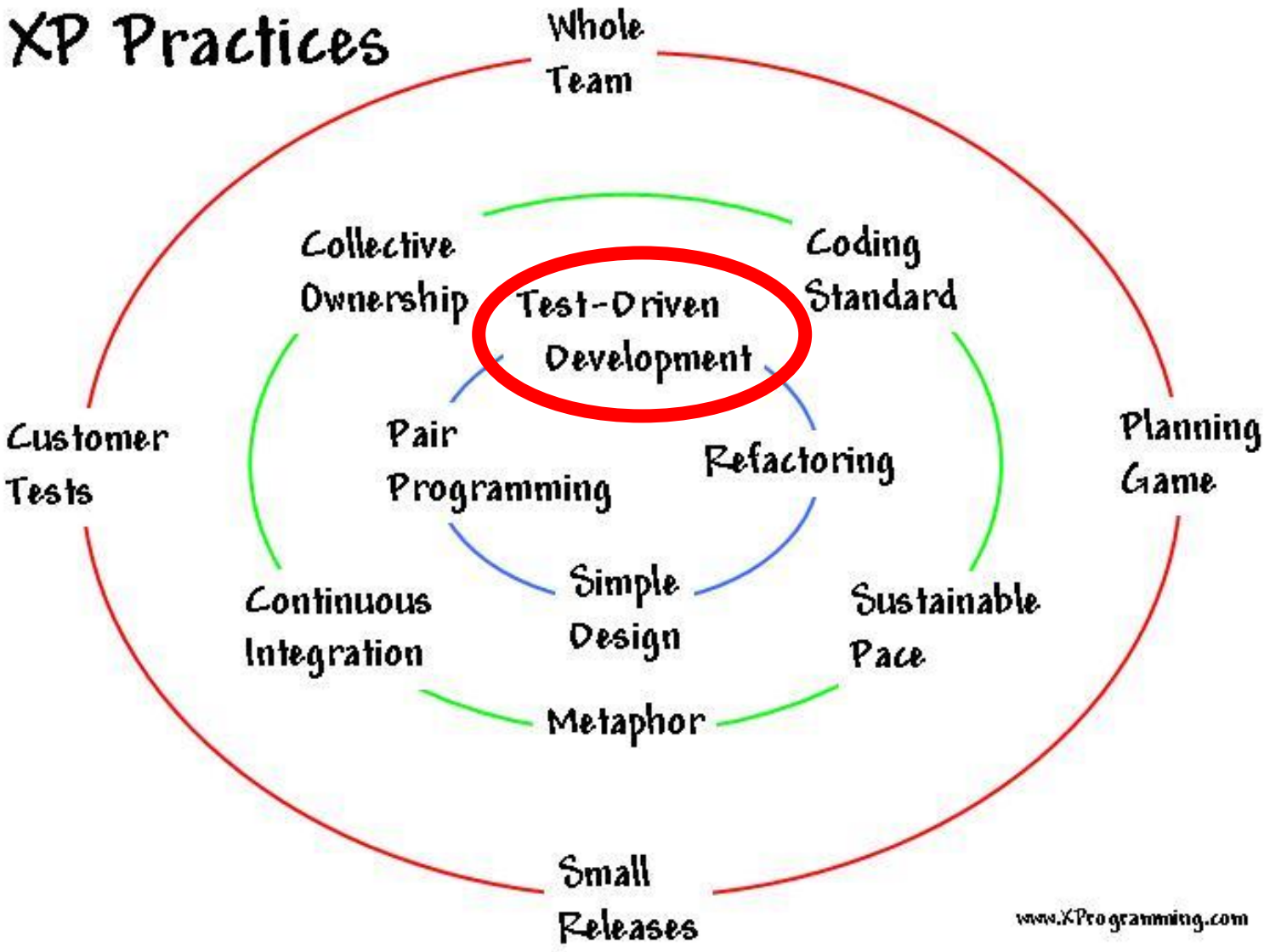
Whole

Peer reviews are a proven technique for improving quality. Pair programming is simply a logical extension of peer reviews...

Small Releases

www.XProgramming.com

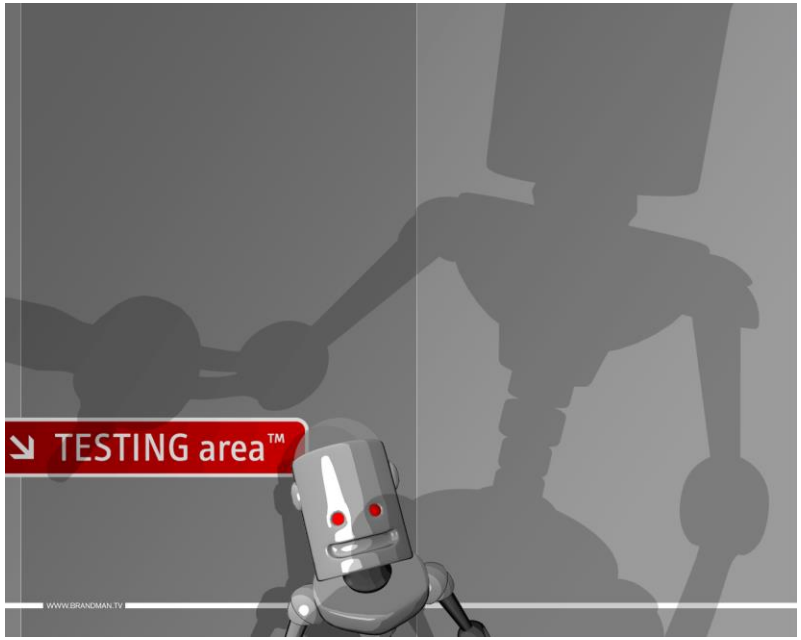
eXtreme Programming



XP Practices

Whole

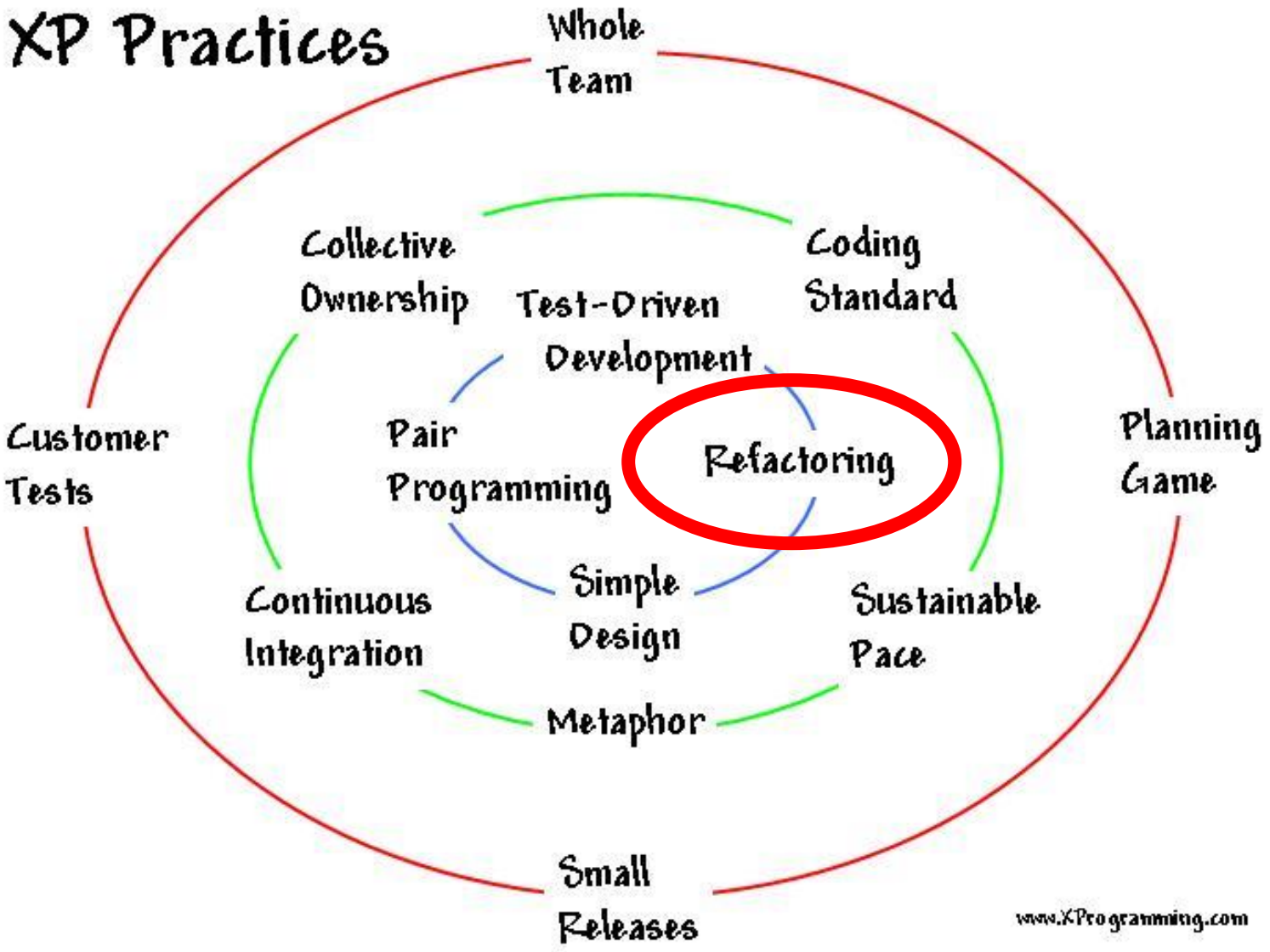
Testing is fundamental to software quality. Designing test cases before coding and employing automated test tools is simply a logical extension...



Small Releases

www.XProgramming.com

eXtreme Programming



XP Practices

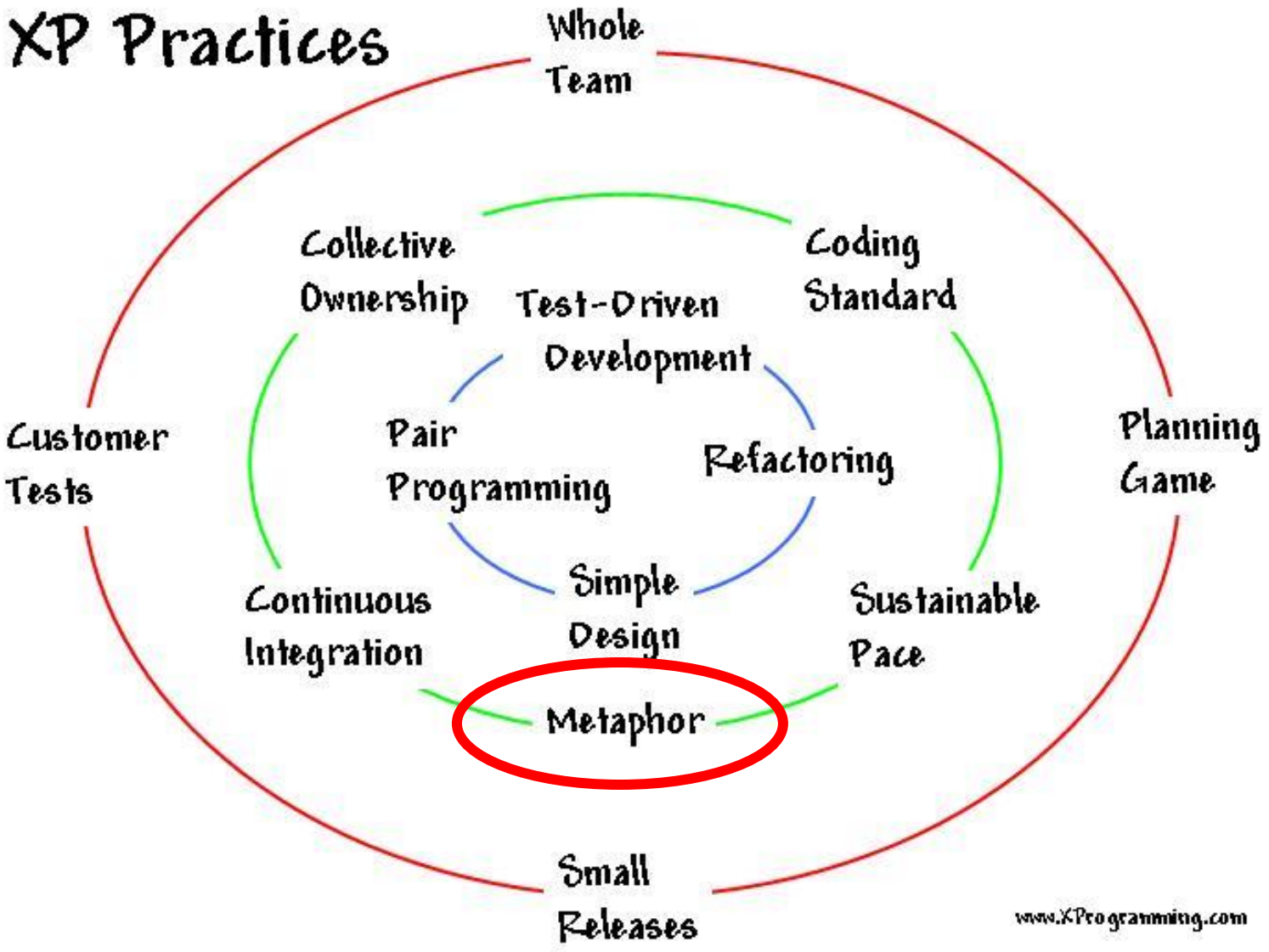
Whole

Even simple designs suffer the effects of entropy over time. Continuous design improvement (refactoring) is a Way of preventing this from happening...

Small Releases

www.XProgramming.com

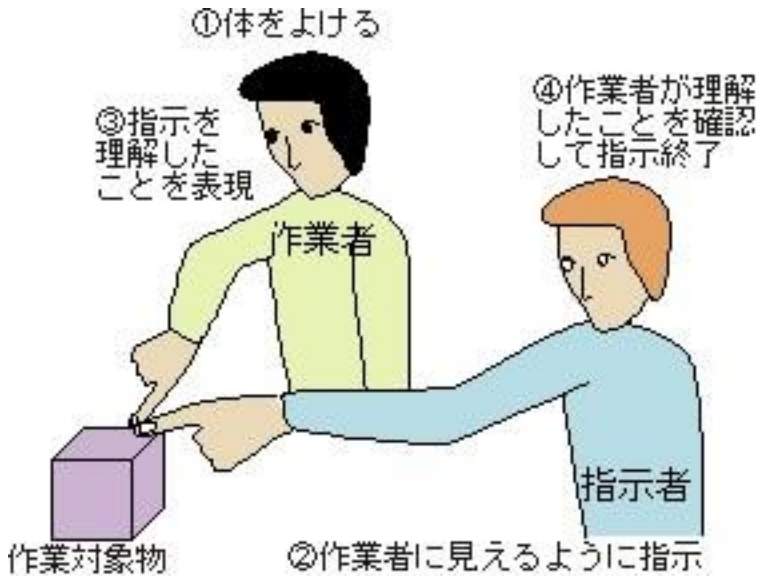
eXtreme Programming



XP Practices

Whole

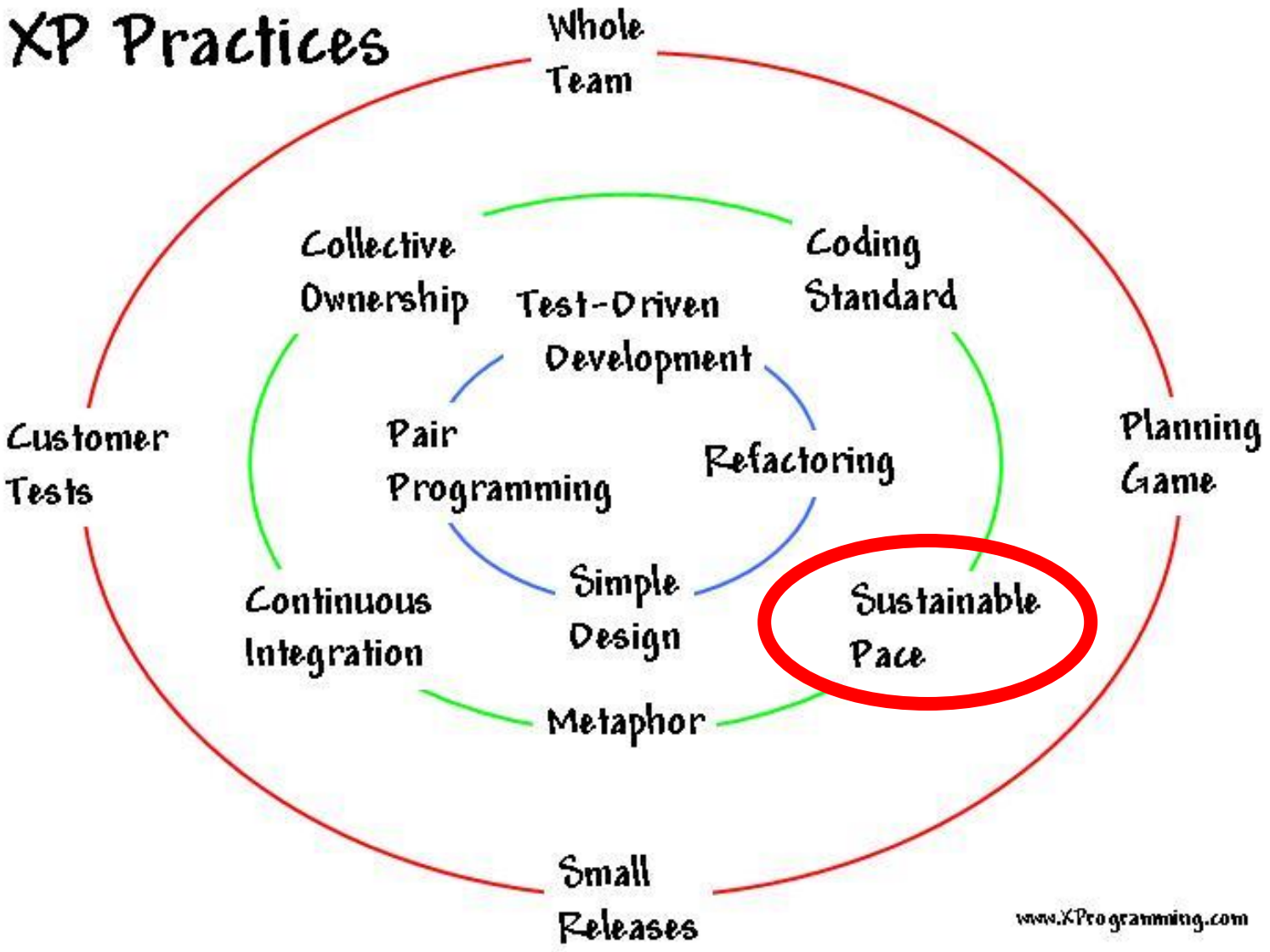
All project teams
Have their own
vocabulary to
describe systems.
A system metaphor
is simply a way of
formalising this...



Small
Releases

www.XProgramming.com

eXtreme Programming



Whole

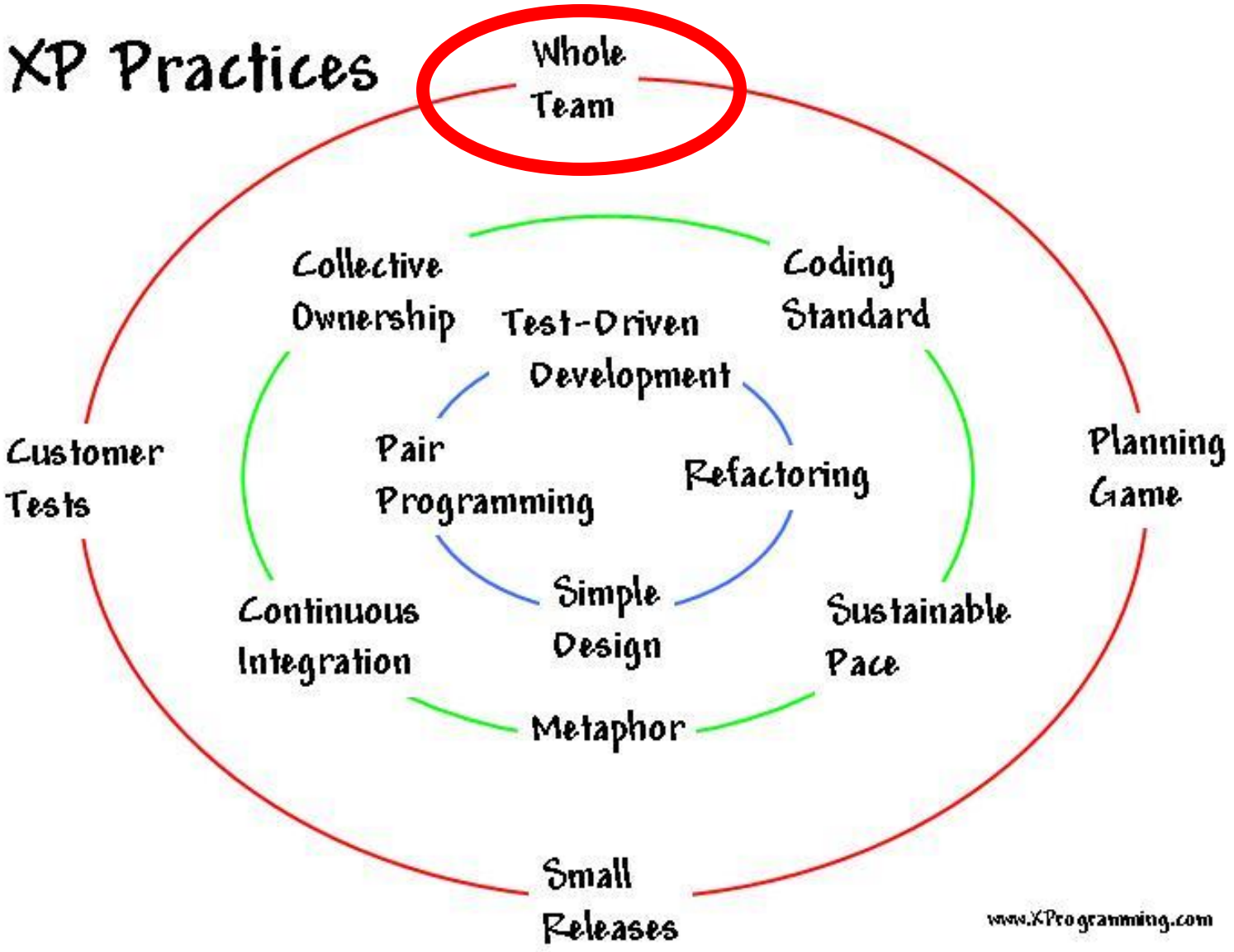
It is obvious that tired and exhausted developers do not give their best. Maintaining a sustainable pace is a sensible response...

Metaphor

Small Releases

www.XProgramming.com

eXtreme Programming



XP Practices

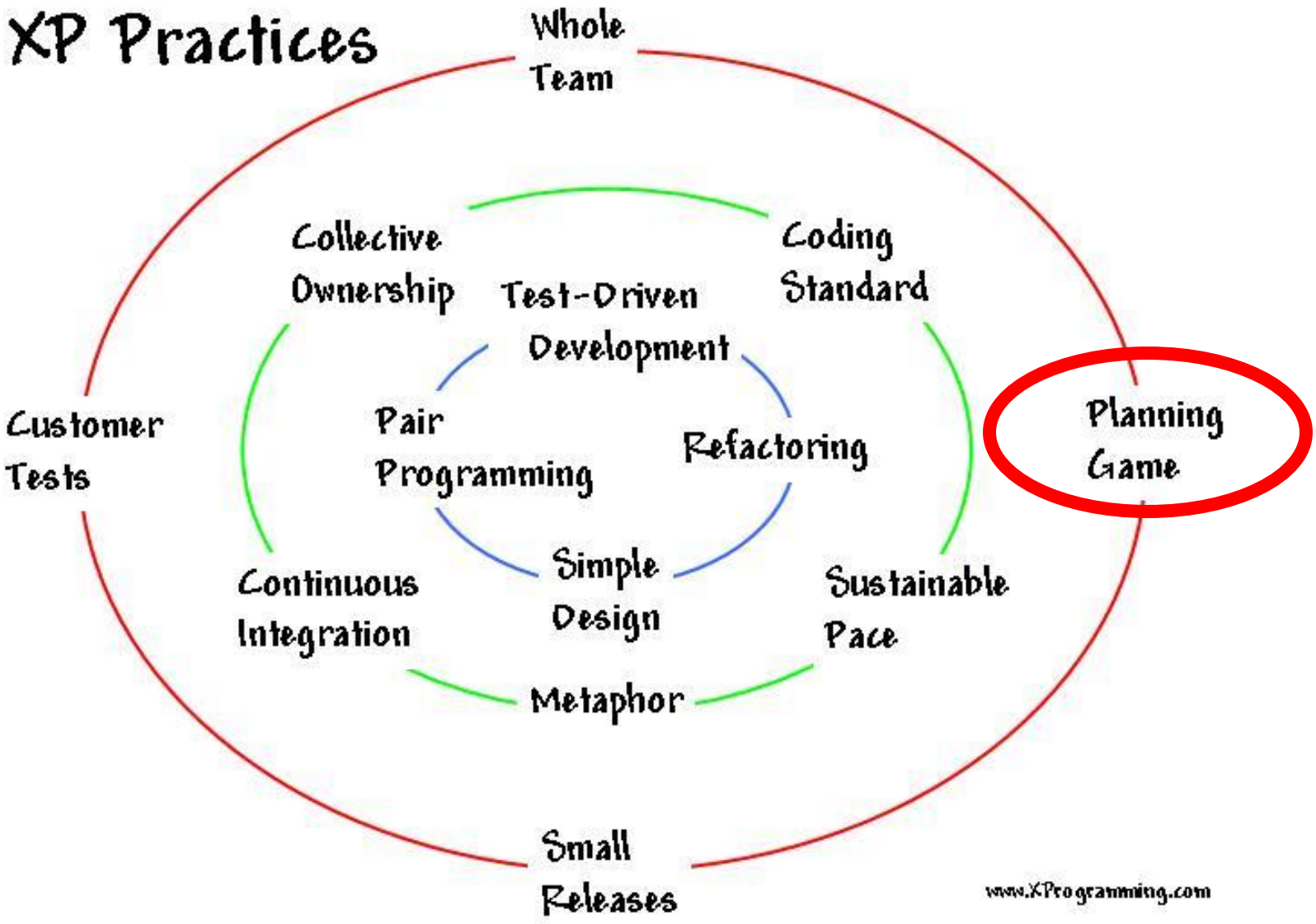
Whole

Communication is an important aspect of all projects. Locating all team members together with a full-time customer representative improves communication...

Small Releases

www.XProgramming.com

eXtreme Programming



Scope + Quality = Time + Cost

Small Releases

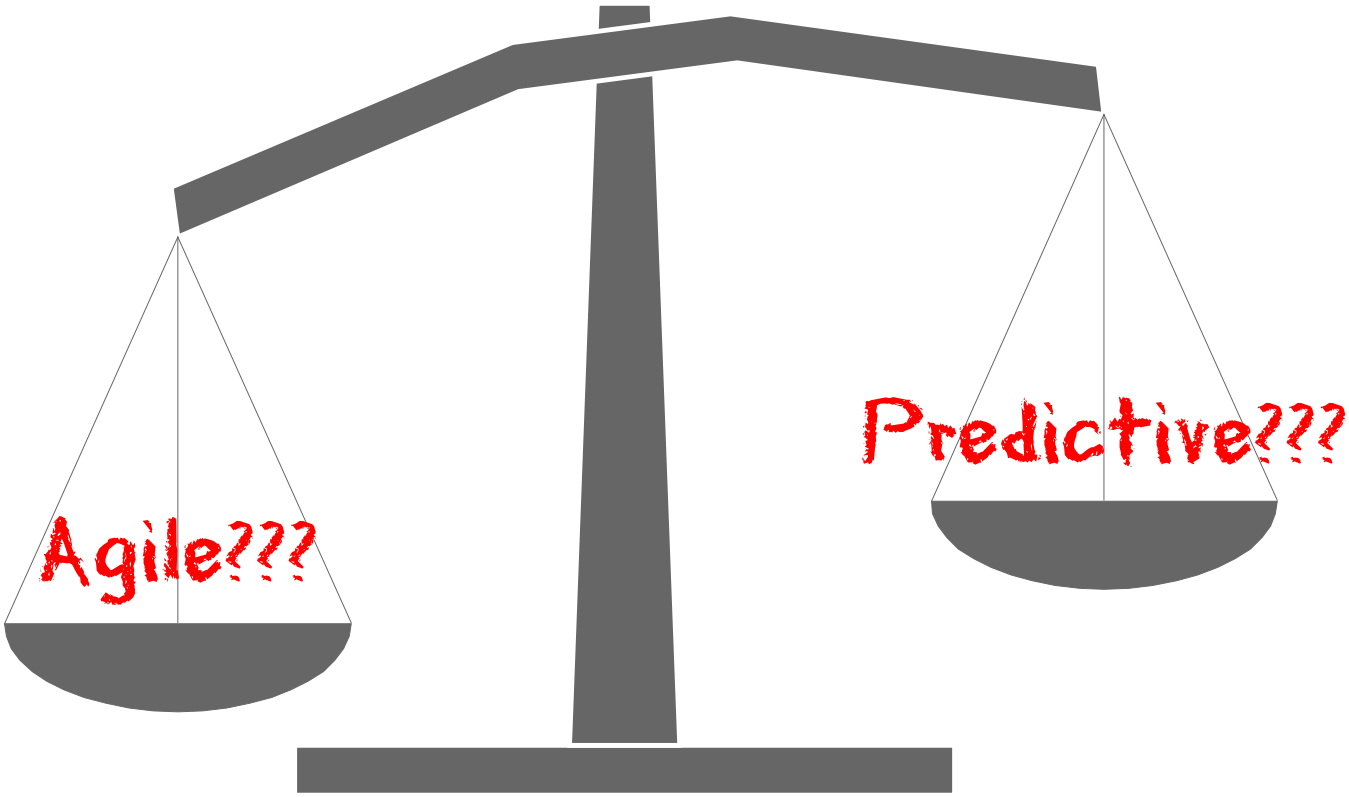
www.XProgramming.com

Scope + Quality = Time + Cost

- ✓Developer estimate time and cost
- ✓Users prioritise scope and quality
- ✓Time and cost are fixed for an iteration
- ✓Scope is planned for an iteration

LonsdaleSystems

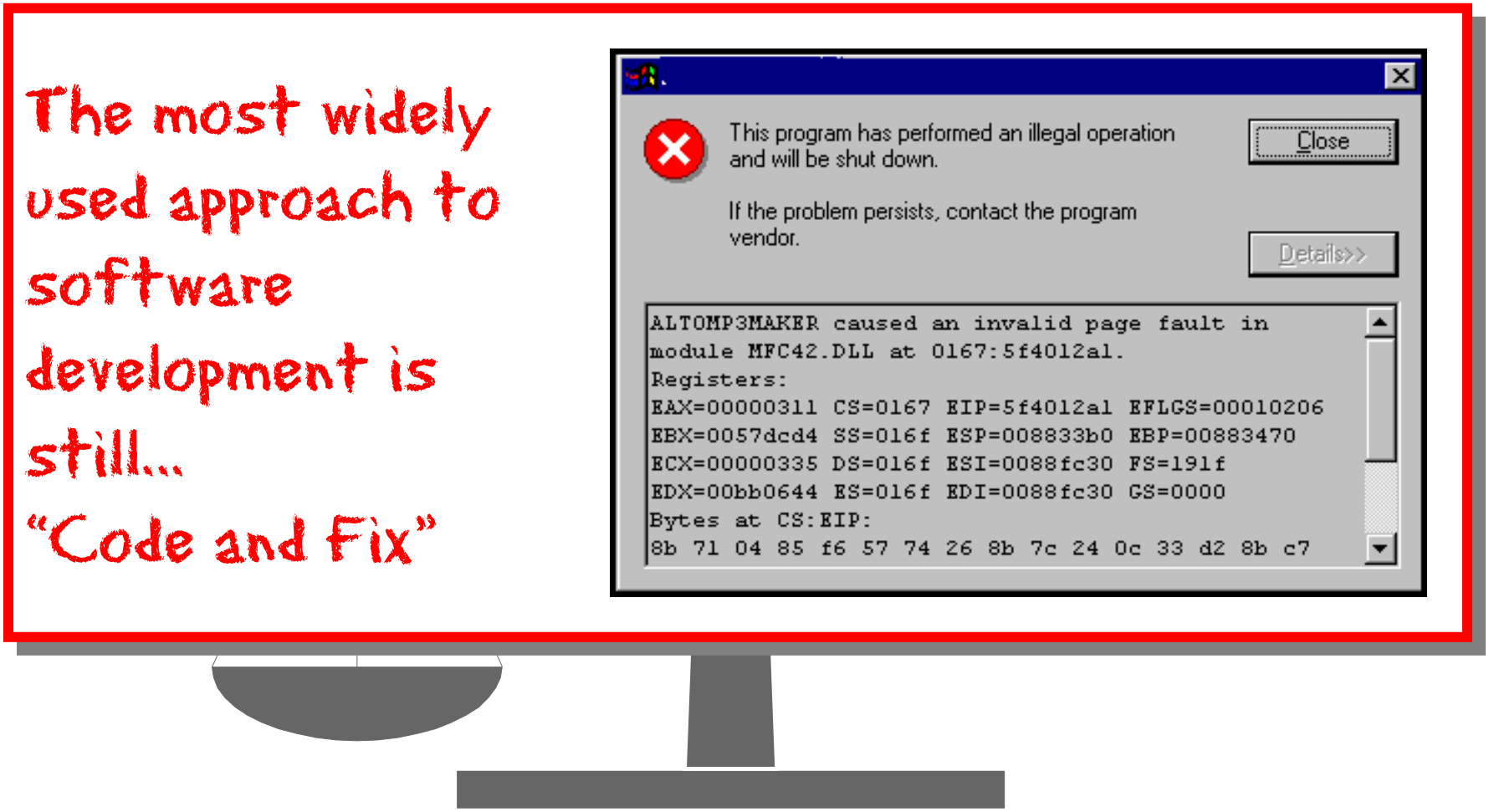
Which Approach?



Which Approach?

The most widely used approach to software development is still...

"Code and Fix"



Which Approach?

"Responding to change over following a plan"
"Great! Now I have a reason to avoid planning and just code whatever comes up next..."

LonsdaleSystems

Which Approach?

	Agile	Predictive
Developers	Superior skills	Adequate skills
Customers	Committed	Available
Requirements	Emerging, changing	Known early, stable
Architecture	Current requirements only	Current and future requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams	Larger teams
Objective	Rapid value	High certainty