COMP 5700/6700/6706
Software Process

Fall 2015

David Umphress

Process Foundations

- Lesson: Process Foundations
- Strategic Outcome: understand the rationale of using processes
- Tactical Outcomes:
  - Know how manufacturing processes have been applied to software
  - Know the contemporary and classical schools of SwE
  - Understand the purpose and benefits of processes
  - Know common processes currently in use
  - Understand the difference between a process model and a "branded" process
  - Understand the Capability Maturity Model (Integrated) for Software; know the model’s key processes

- Support material:
  - Reading, “How much process is enough”
  - Reading, “The laws of software process”
- Instant take-aways
  - interview buzzwords
- Bookshelf items:
  - MIL-STD-498*
  - IEEE Std 1074-1997 Standard for Developing Software Life Cycle Processes*
  - IEEE Std 1074.1-1995 IEEE Guide for Developing Software Life Cycle Processes*
  - IEEE Std 12207.0 - 1996 Software Life Cycle Processes*
  - DOD-STD-2167A Defense System Software Development*
  - Capability Maturity Model - Integrated*
  - Process Comparison Overview*

* downloadable from Canvas *References*

Syllabus

- Software engineering raison d’être
- Process foundations
- Common process elements
- Analysis
- Architecture
- Estimation
- Scheduling
- Construction
- Reviews
- Refactoring
- Integration
- Repatterning
- Measurements
- Process redux
- Process descriptions*
- Infrastructure*
- Retrospective

- **Process foundations**
  - **Industrial quality movement**
  - **Software quality movement**
- **Processes a la SwE**
  - **Classical school**
  - **Contemporary school**
- **Processes explored further**
  - **Processes rationale**
  - **Function of processes**
  - **Process model**
- **Samples**

Discussion …

How do we ensure that a software product works as desired?

What stands in the way of your achieving “success” the first time, every time?

Contemporary Quality Movement

- Philip Crosby
  - guiding principles:
    - “quality is conformance to requirements”
    - “quality is free, but only to those who are willing to pay heavily for it.”
  - Quality management maturity grid
    - 5 levels of maturity:
      - uncertainty, awakening, enlightenment, wisdom, certainty
    - 6 measurement categories
      - management understanding
      - quality organization status
      - problem handling
      - cost of quality as % of sales
      - quality improvement actions
      - summation of company quality posture

Quality Movement wrt Software

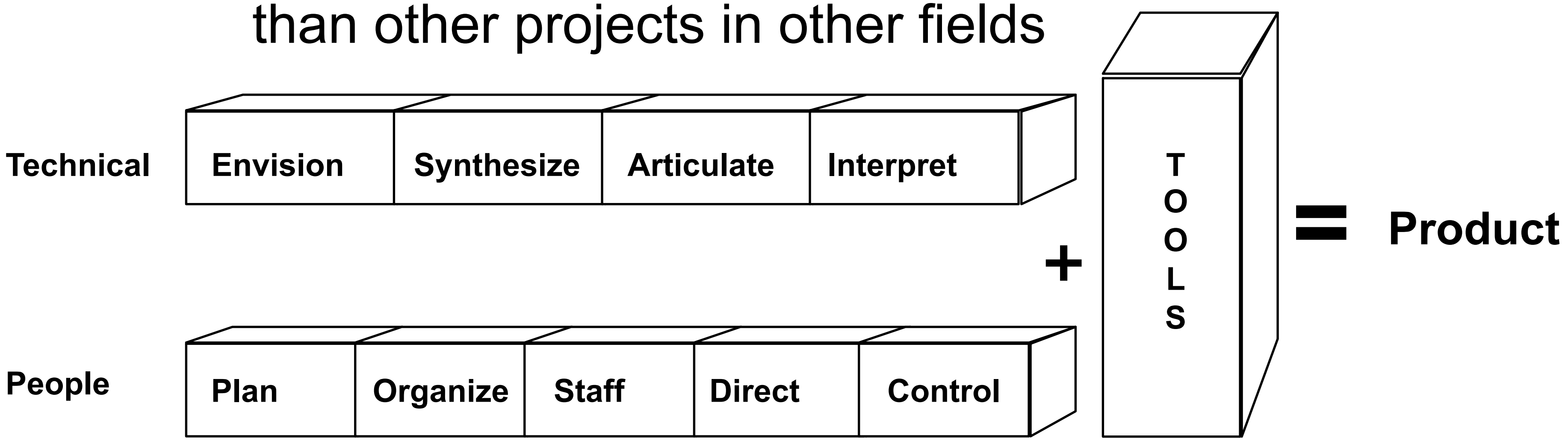
- Lennart Sandholm
  - quality policy
    - statement of corporate-wide commitment to quality
    - promulgated by senior software executive
  - quality objectives
    - statements of measurable improvements
    - e.g., number of errors per thousand LOC
  - quality system
    - used to achieve the quality objectives
    - e.g., standards, procedures, etc.
  - quality assurance organization
    - facilitating body
    - may process and/or product oriented

Premise that has evolved:

- The quality of a product is largely determined by the quality of the process that is used to develop and maintain it.
- Corollary: defined processes provide visibility into production

Ghost of SwE Past

- Classical software development philosophy
  - tenets:
    - software engineering = building software = technical activity
    - grab-bag of disjoint technical actions
    - software projects can be orchestrated no differently than other projects in other fields



Technical: Envision, Synthesize, Articulate, Interpret

+

People: Plan, Organize, Staff, Direct, Control

+ TOOLS

= Product

Devolution of Classical SwE

So …

- How well does an *ad hoc* approach scale up?
- Consider % of requirements allocated to software:

1960's F-4

8%

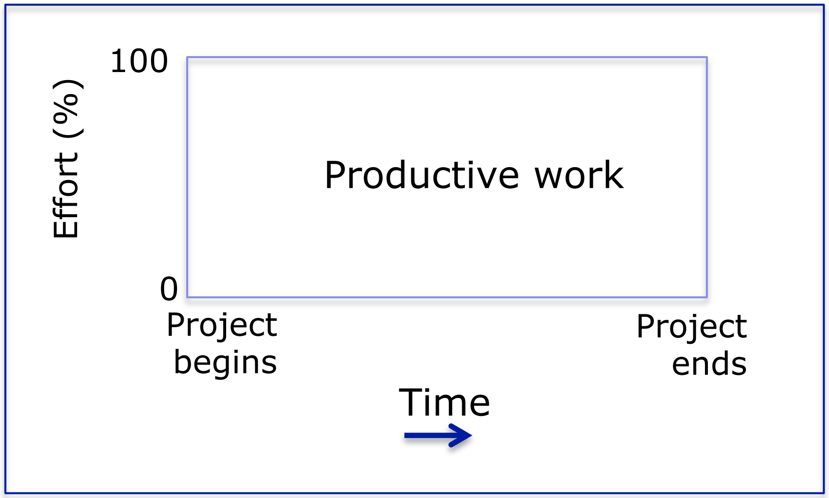
1980's F-16

45%

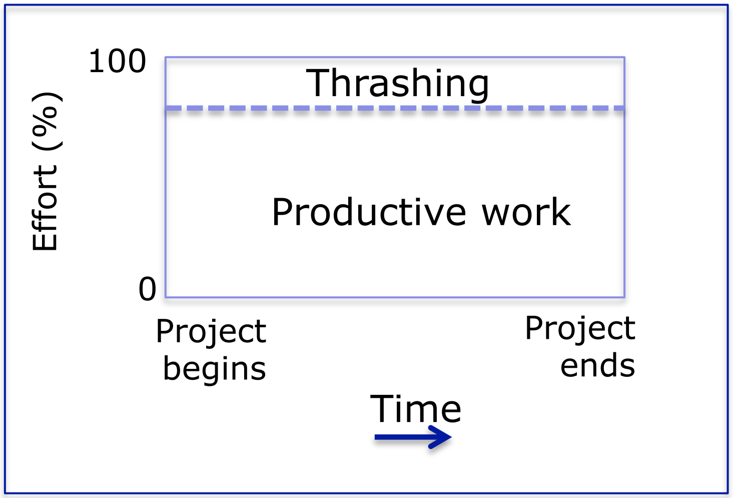
2000's F-22

80%

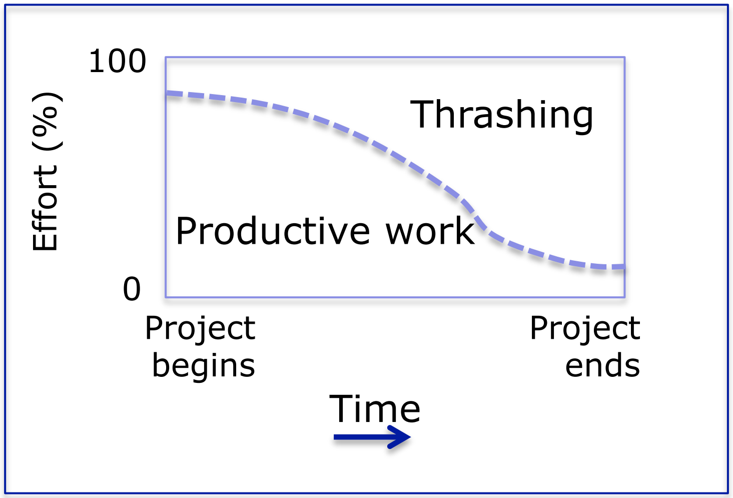
Reality sets in ….



inexperienced perspective



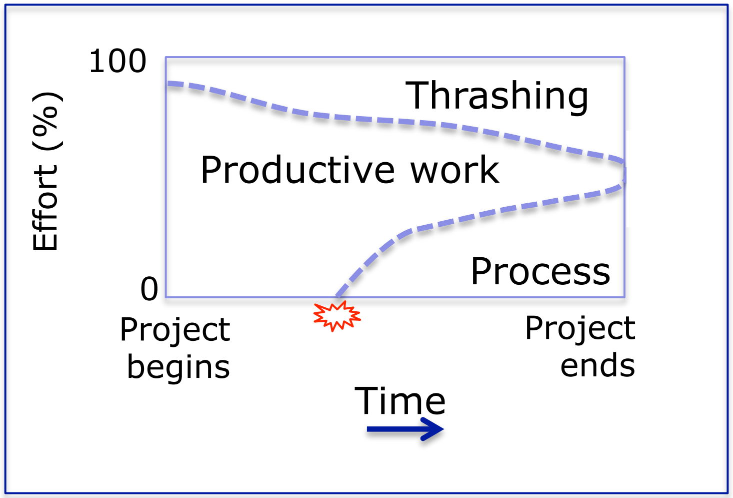
naïve perspective



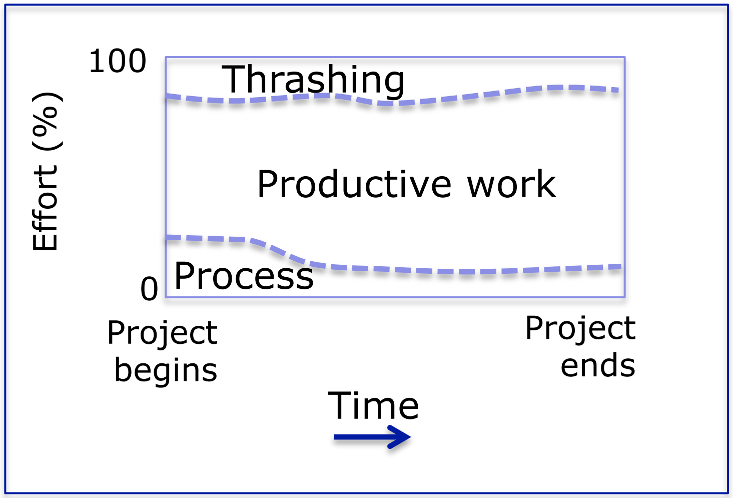
initial experience

McConnell, S., 1998, The Power of Process, *Computer*, May, pp 100-102.

Reality sets in …



realization



evolution

Process = conscious recognition of the way in which to build software

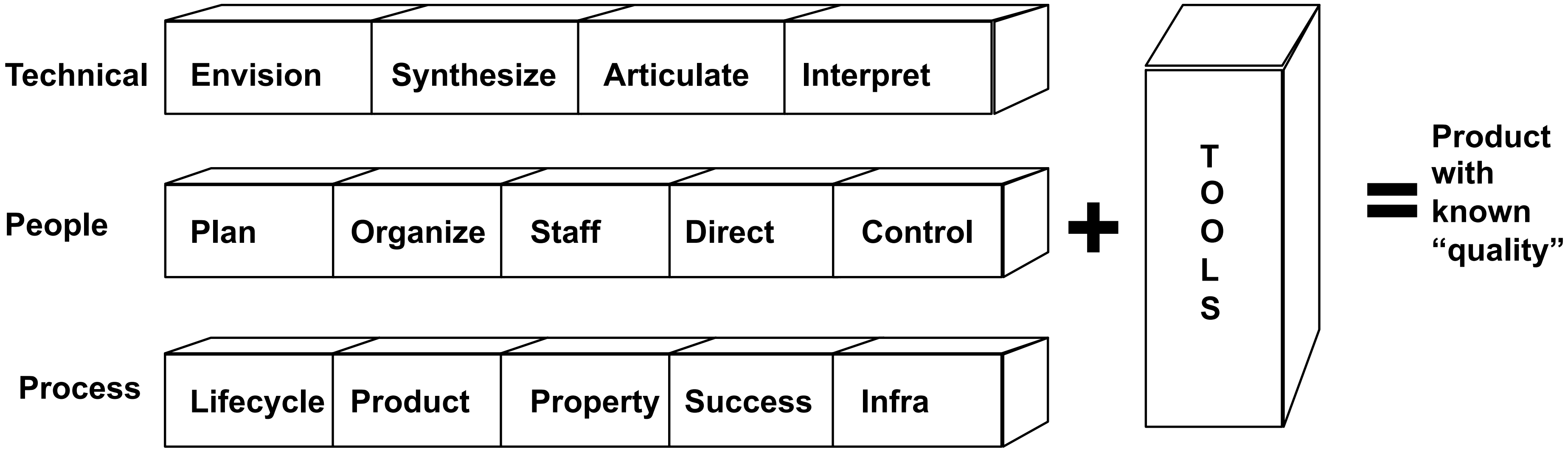
Adapted from
McConnell, S., 1998,
The Power of Process,
Computer, May, pp
100-102.

Ghost of SwE Present*

- Contemporary software development philosophy
  - tenets
    - software engineering = technical facets + managerial facets orchestrated by process
    - focus on process activities
      - teach the troops correct principles and they will govern themselves
      - Humphrey:
        - » "The quality of software is governed by the quality of its worst components."
        - » "The quality of a software component is governed by the individuals who developed it."
      - teams with comprehensive processes are more likely to contain cost and ensure quality than those without
      - processes can exist on a project-by-project basis, but are leveraged best on an organization-wide basis

SwE now

- Contemporary software development philosophy (con’t)
  - hence



Technical	Envision	Synthesize	Articulate	Interpret	
People	Plan	Organize	Staff	Direct	Control
Process	Lifecycle	Product	Property	Success	Infra

+

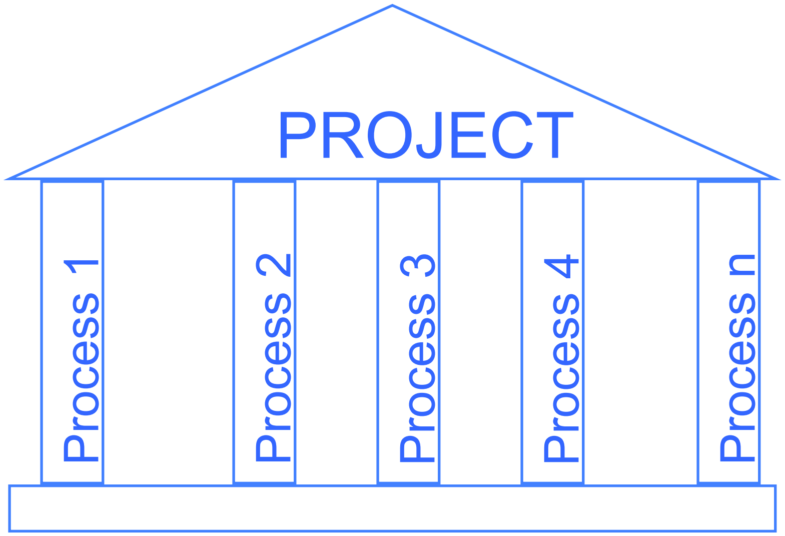
T O O L S

=

Product with known “quality”

Processes Explored Further

- Process
  - ... is the set of tasks needed to produce software
    - Note: not all tasks need be documented or defined … only the most relevant ones, where “relevance” depends on organizational culture, team culture, tools, project criticality, etc.
  - our goal: use process controls to guide production



Process Rationale

- Processes help ...
  - boost the probability of product quality
  - identify the principal activities of doing a job
  - separate routine from complex tasks
  - establish starting and stopping criteria
  - facilitate tracking and measuring performance
  - provide orderly mechanism for learning
  - establish corporate memory
  - create a defined baseline for improvement
  - **put everyone on the same page**

Humphrey

A Process Clearly States …

- Purpose
- Inputs
- Entry criteria
- Activities
- Roles
- Measures
- Verification steps
- Outputs
- Exit criteria

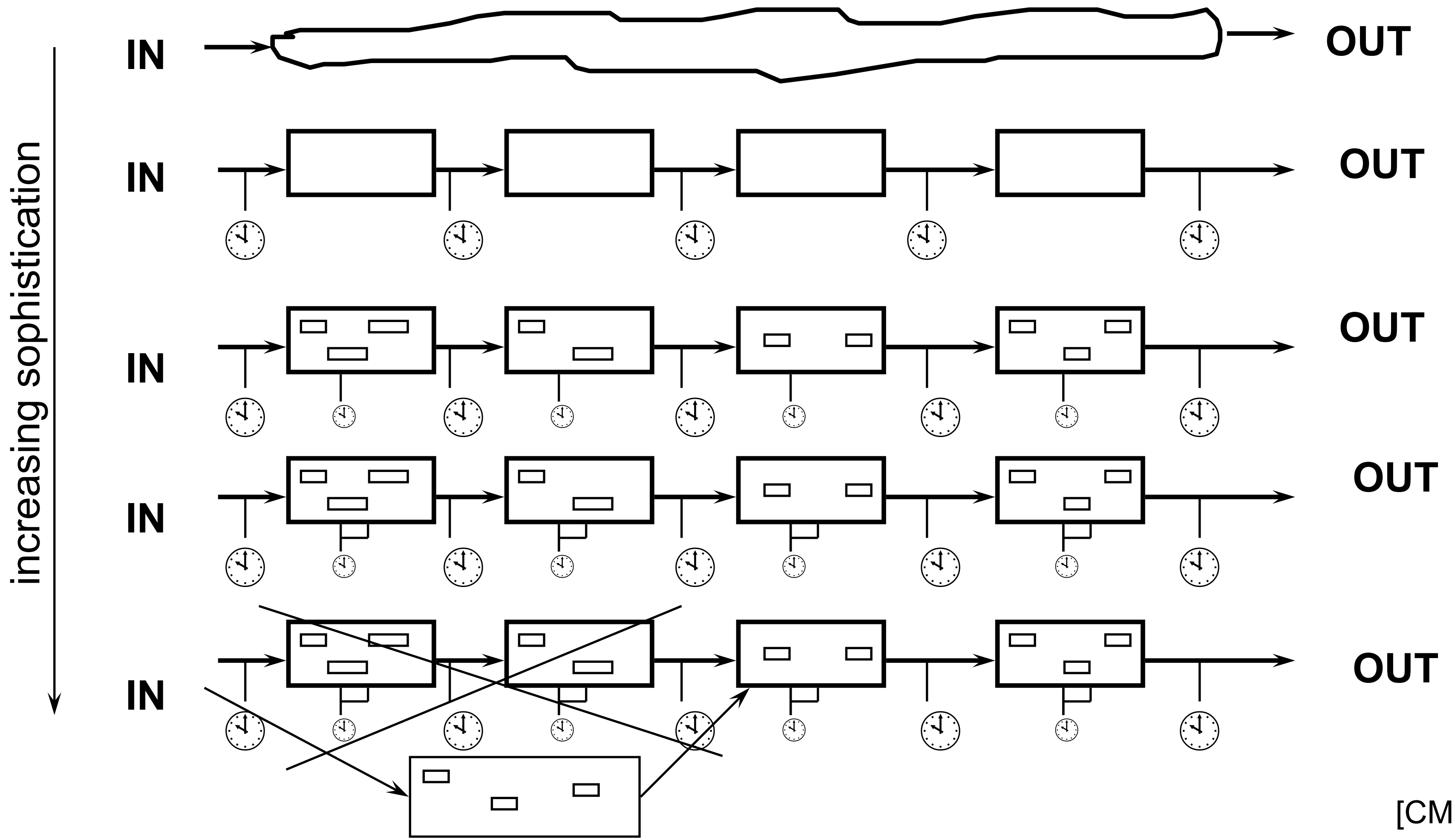
Example Process

- Personal Software Process (PSP) process script

Entry		• Problem description • Defect standard type
Tasks	1. Planning	• Produce a requirements statement • Estimate and record the required development time • Record time spent planning
	2. Development	• Design the program • Implement the design • Compile program; fix and log all defects found • Test the program and fix and log all defects found. • Record time spent in development
	3. Postmortem	• Calculate and record performance statistics
Exit		• Tested program • Completed plan, defect log, time log

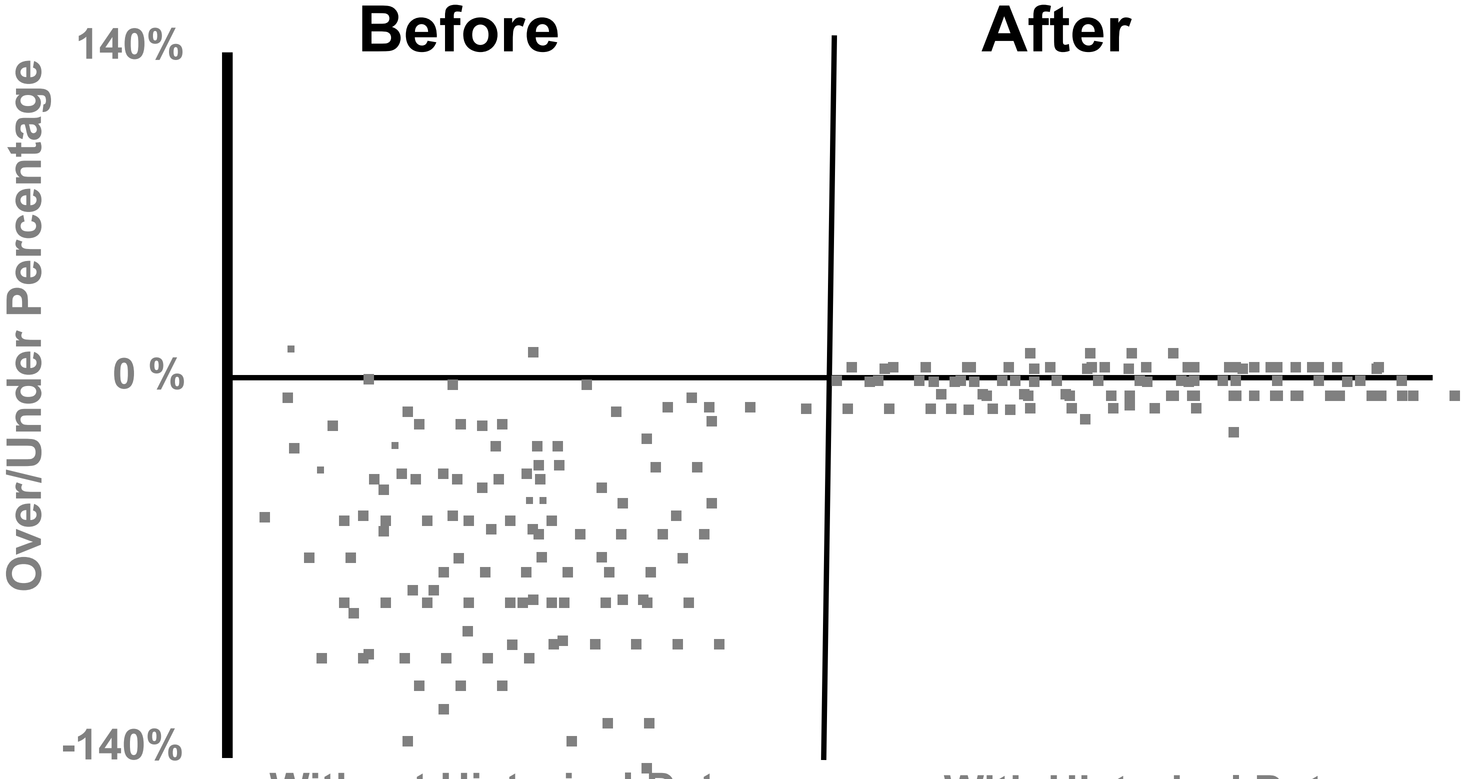
Processes Explored Further

- Function of processes: engineering insight



[CMMI]

Empirical evidence



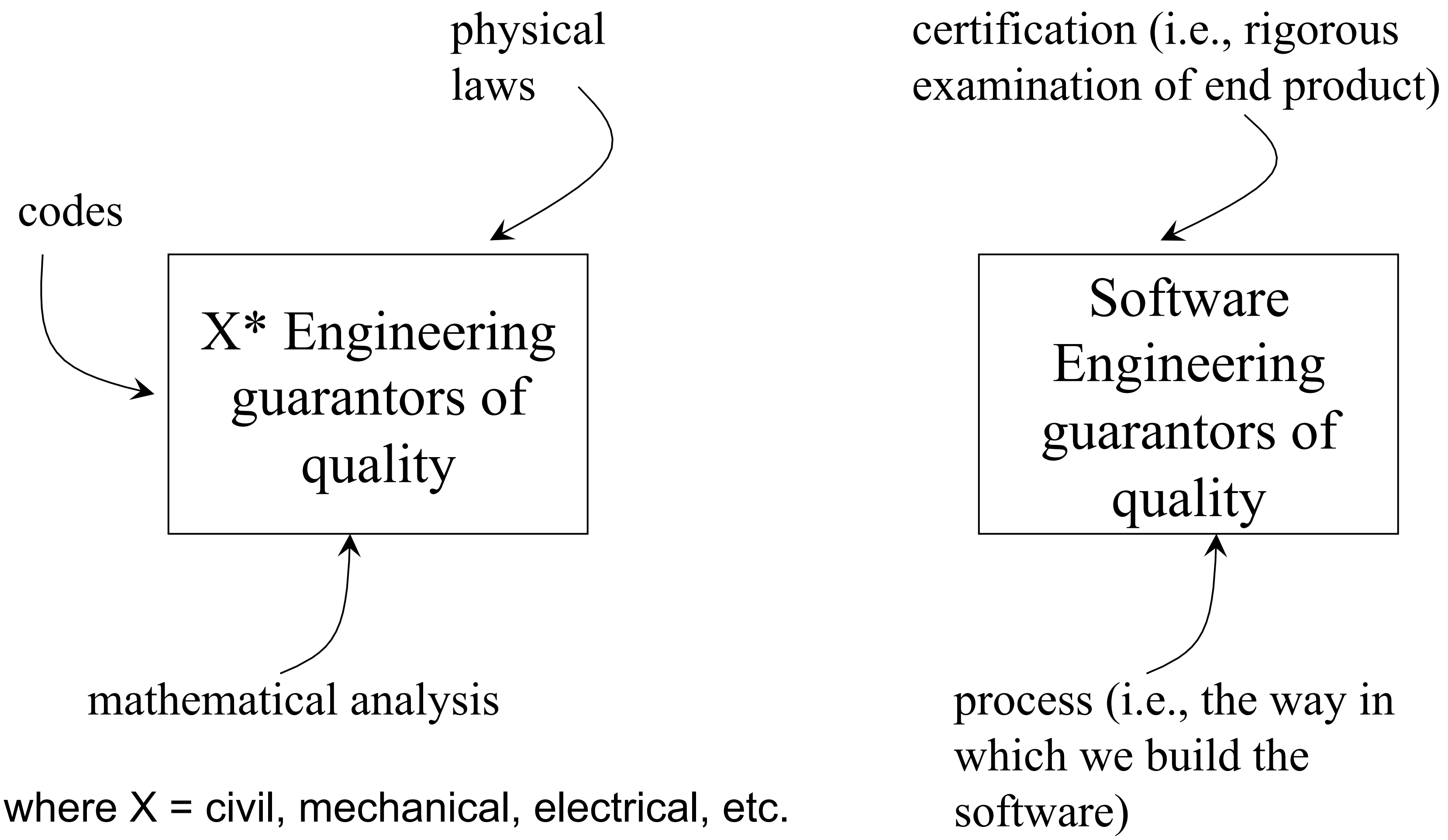
Without Historical Data
Variance between + 20% to - 145%
(Mostly Level 1 & 2)

With Historical Data
Variance between - 20% to + 20%
(Level 3)

(Based on 120 projects in Boeing Information Systems)

Reference: John D. Vu. “Software Process Improvement Journey:From Level 1 to Level 5.” 7th SEPG Conference, San Jose, March 1997.

Historical Perspective



physical laws

codes

X* Engineering guarantors of quality

mathematical analysis

certification (i.e., rigorous examination of end product)

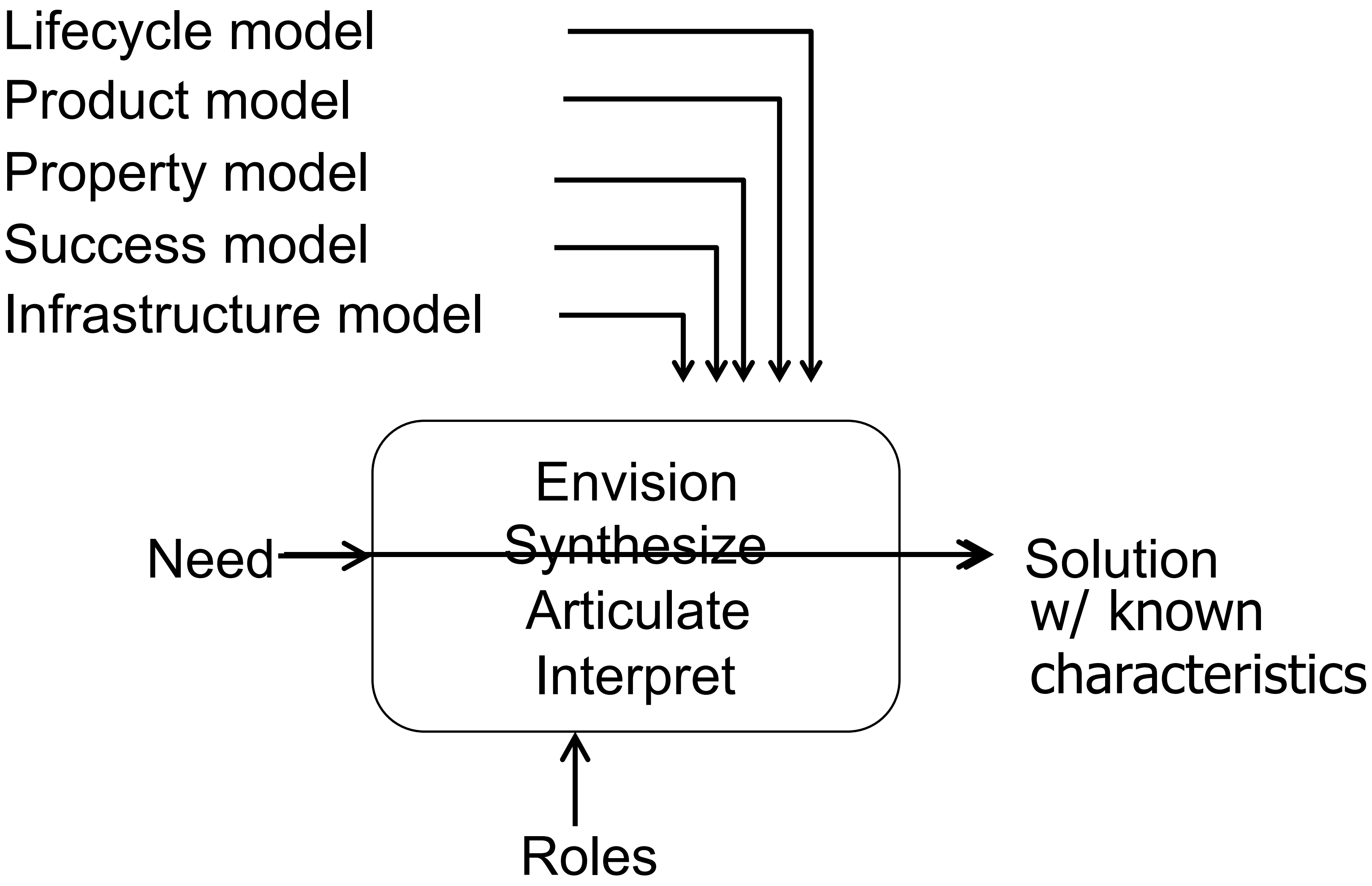
Software Engineering guarantors of quality

process (i.e., the way in which we build the software)

* where X = civil, mechanical, electrical, etc.

Traditional engineering focuses on the product; software engineering focuses on the process of building the product

Process … so far …



Lifecycle model

Product model

Property model

Success model

Infrastructure model

Need

Envision

Synthesize

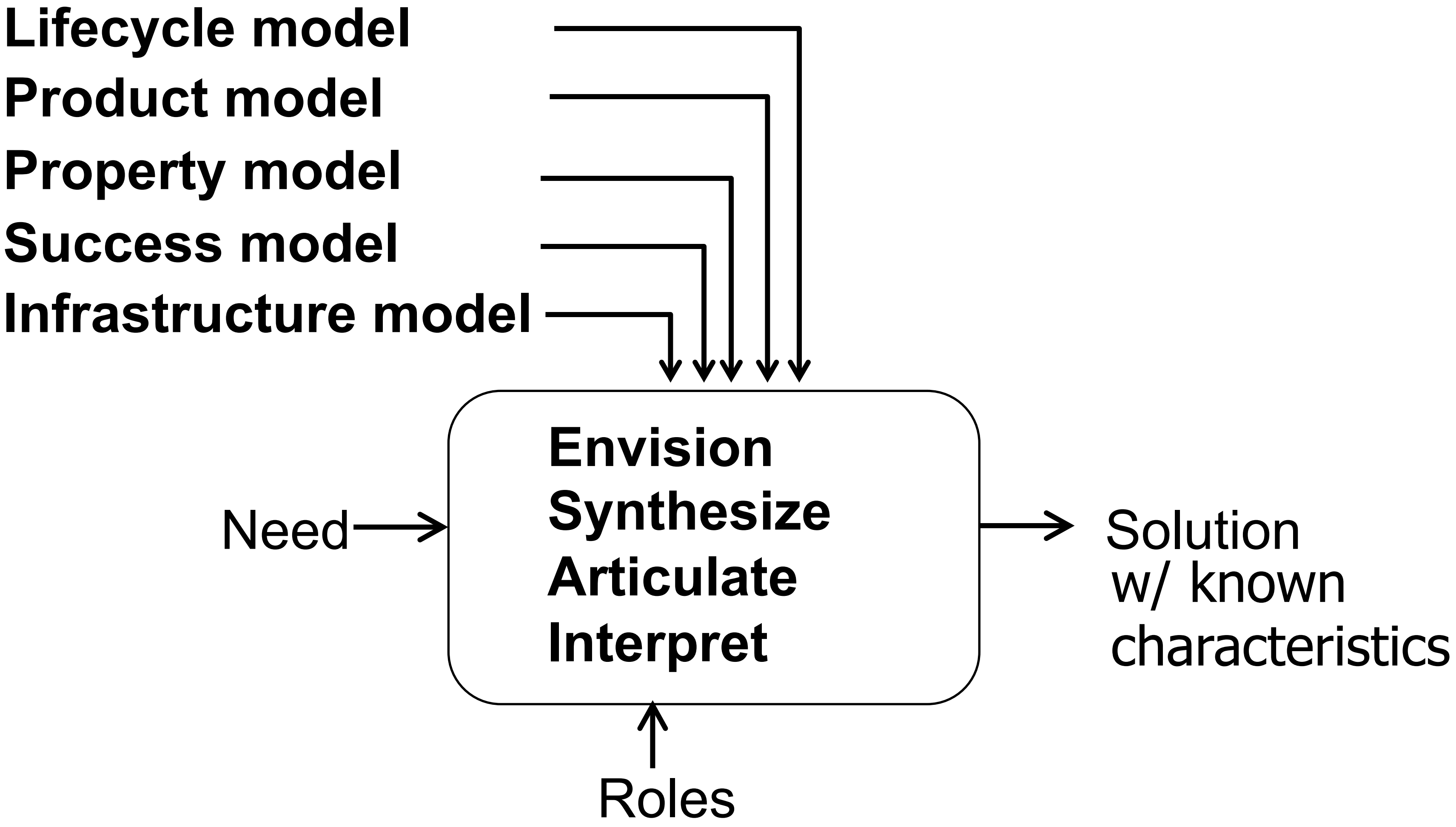
Articulate

Interpret

Solution w/ known characteristics

Roles

Beware: There is no “the” process



Lifecycle model
Product model
Property model
Success model
Infrastructure model

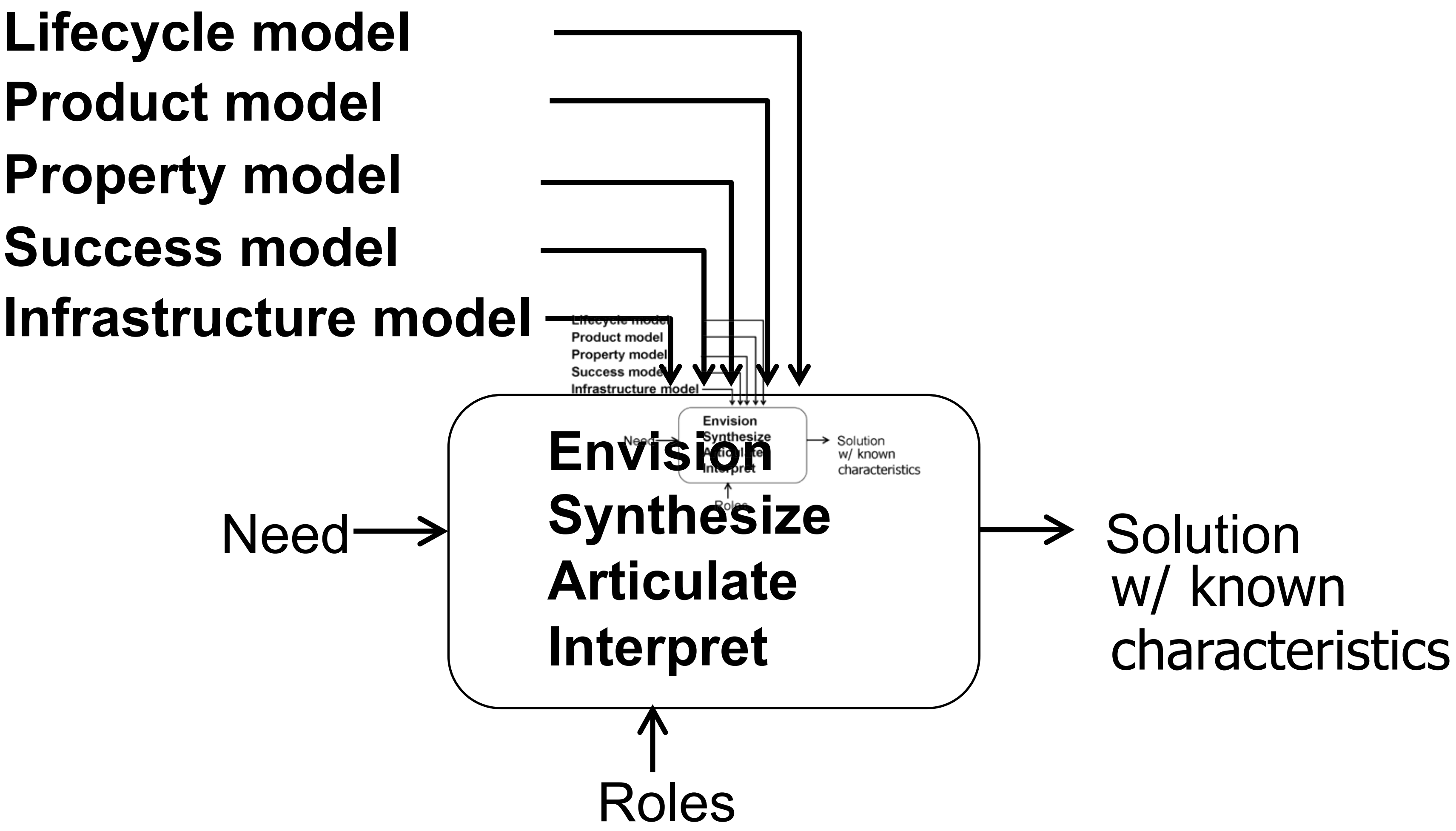
Need

Envision
Synthesize
Articulate
Interpret

Solution
w/ known
characteristics

Roles

Beware: There is no “the” process



Lifecycle model

Product model

Property model

Success model

Infrastructure model

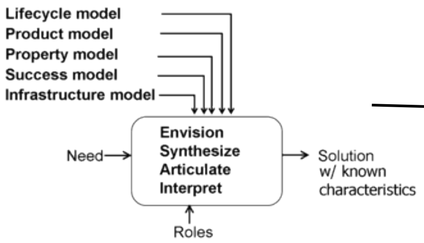
Need

Envision
Synthesize
Articulate
Interpret

Solution
w/ known
characteristics

Roles

Beware: There is no “the” process



process models

- definition
  - Descriptive characteristics of an effective development effort
- examples
  - ISO 9001
  - CMMI

"branded" processes

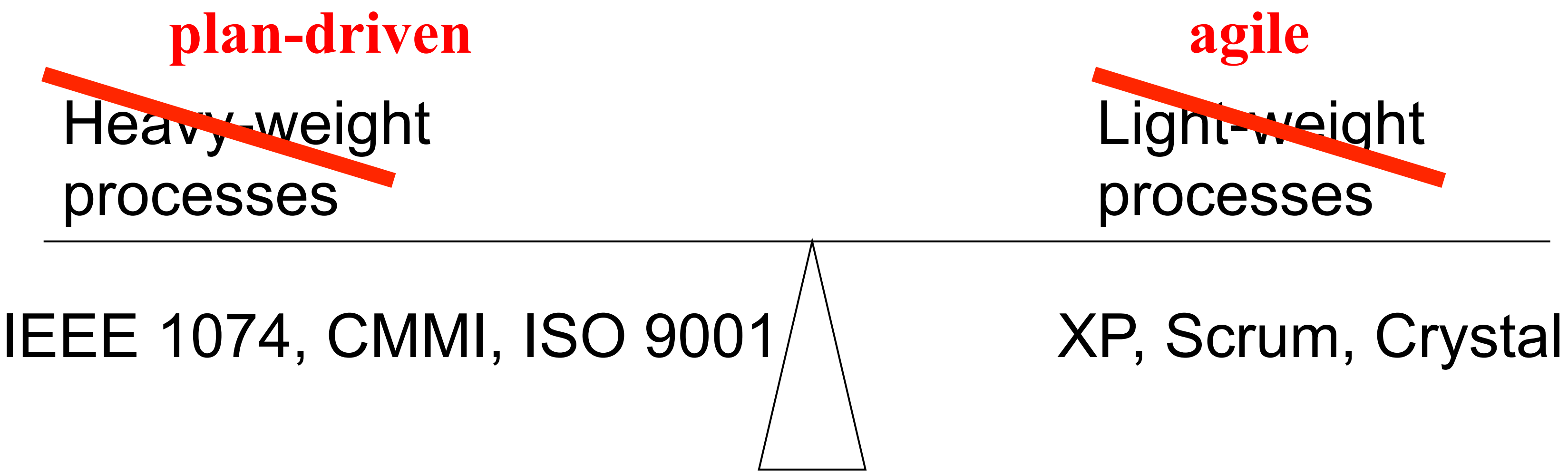
- definition
  - Prescriptive descriptions of tasks required to develop a software solution
- examples
  - IEEE 1074
  - MIL-STD-498
  - Extreme Programming

instantiate as

SOFTWARE PROCESS

Historical Perspective

Process Darwinism



plan-driven

~~Heavy-weight~~ processes

agile

~~Light-weight~~ processes

IEEE 1074, CMMI, ISO 9001

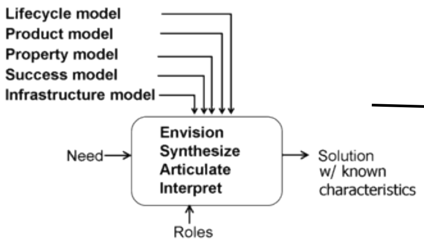
XP, Scrum, Crystal

requirements containment	vs	requirements adaptation
design-oriented	vs	construction-oriented
predictive	vs	adaptive

Industry Practices

- "Branded" processes (prescriptive)
  - MIL-STD 2167A
  - MIL-STD-498 -> IEEE 12207.0
  - IEEE 1074
  - XP
  - NASA xxx, Boeing xxx, etc., etc.
- Process models (descriptive)
  - ISO 9001
  - CMMI (Capability Maturity Model – Integrated)
  - SPICE (Software Process Improvement and Capability dEtermination … ISO/IEC TR 15504)

"Branded" Processes



process models

- definition
  - Descriptive characteristics of an effective development effort
- examples
  - ISO 9001
  - CMMI

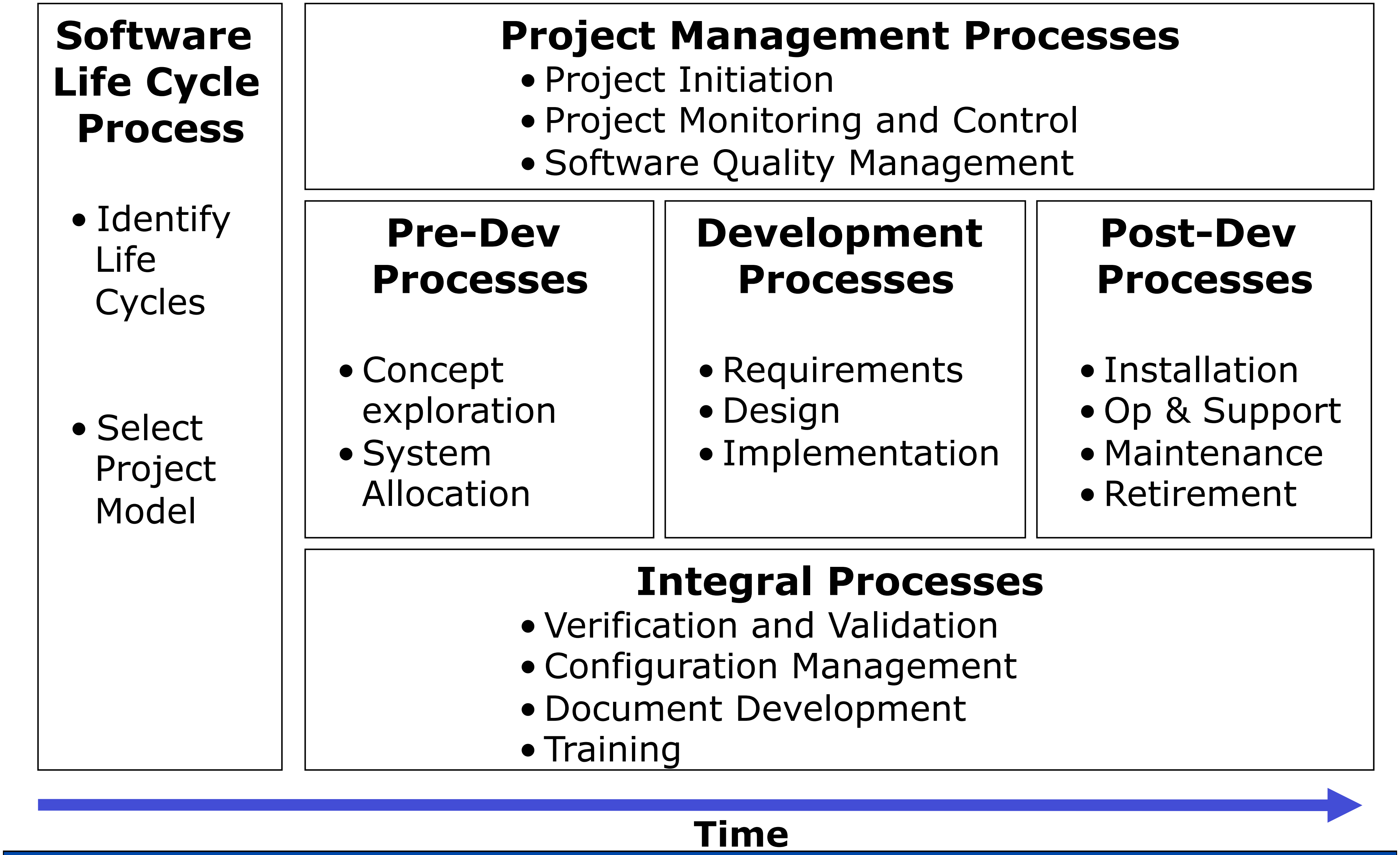
"branded" processes

- definition
  - Prescriptive descriptions of tasks required to develop a software solution
- examples
  - IEEE 1074
  - MIL-STD-498
  - Extreme Programming

instantiate as

SOFTWARE PROCESS

IEEE Std 1074



Software Life Cycle Process

- Identify Life Cycles
- Select Project Model

Project Management Processes

- Project Initiation
- Project Monitoring and Control
- Software Quality Management

Pre-Dev Processes

- Concept exploration
- System Allocation

Development Processes

- Requirements
- Design
- Implementation

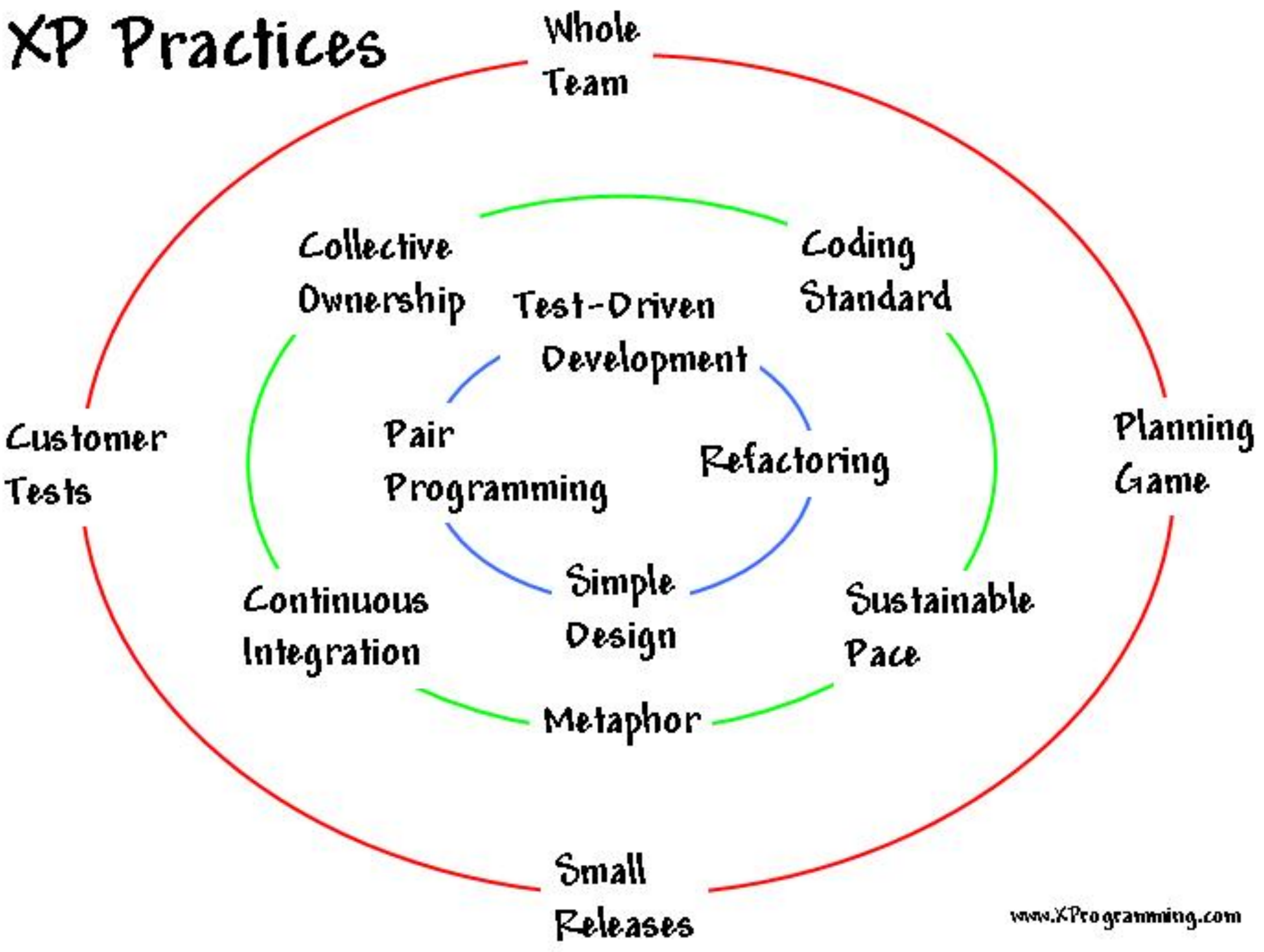
Post-Dev Processes

- Installation
- Op & Support
- Maintenance
- Retirement

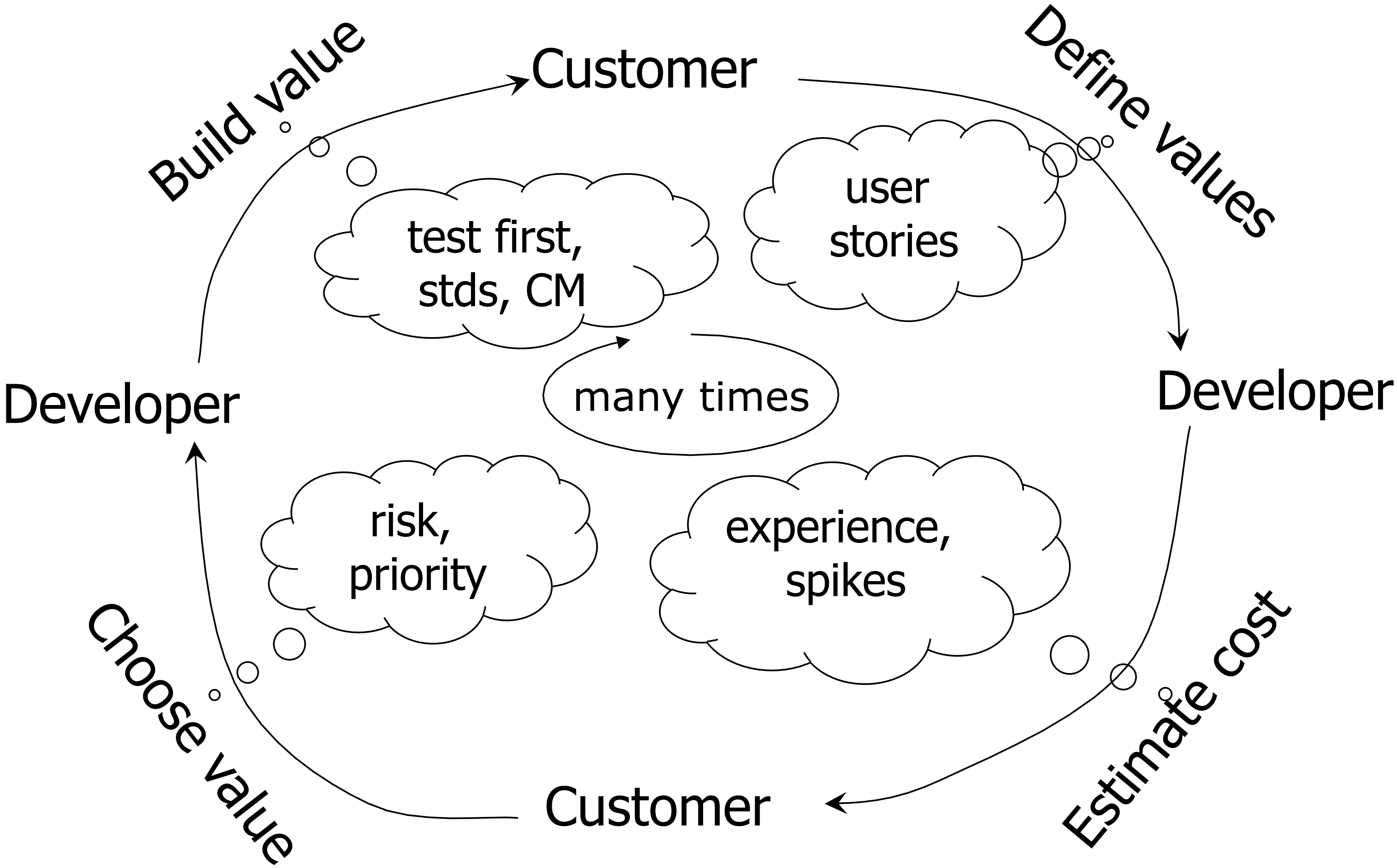
Integral Processes

- Verification and Validation
- Configuration Management
- Document Development
- Training

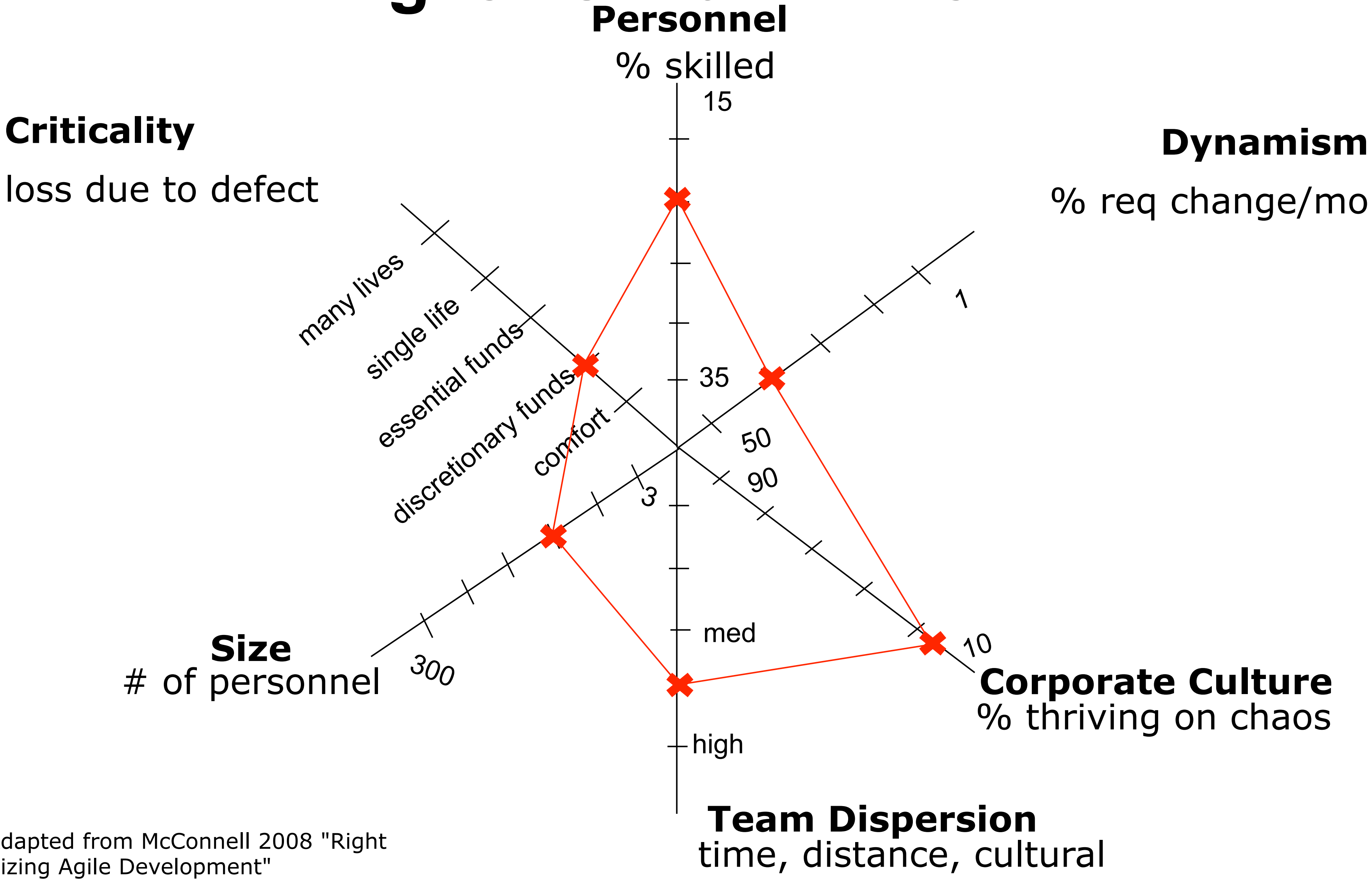
Extreme Programming (XP)



Extreme Programming (XP)

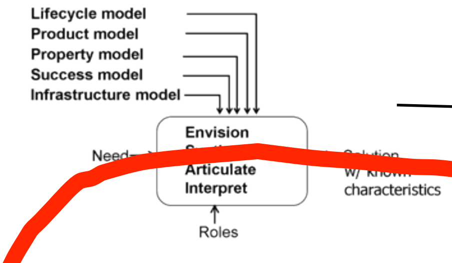

Customer
Define values
Developer
Estimate cost
Customer
Choose value
Developer
Build value
user stories
experience, spikes
risk, priority
test first, stds, CM
many times

Agile vs Plan-Driven



Adapted from McConnell 2008 "Right Sizing Agile Development"

Model Processes



process models

- definition
  - Descriptive characteristics of an effective development effort
- examples
  - ISO 9001
  - CMMI

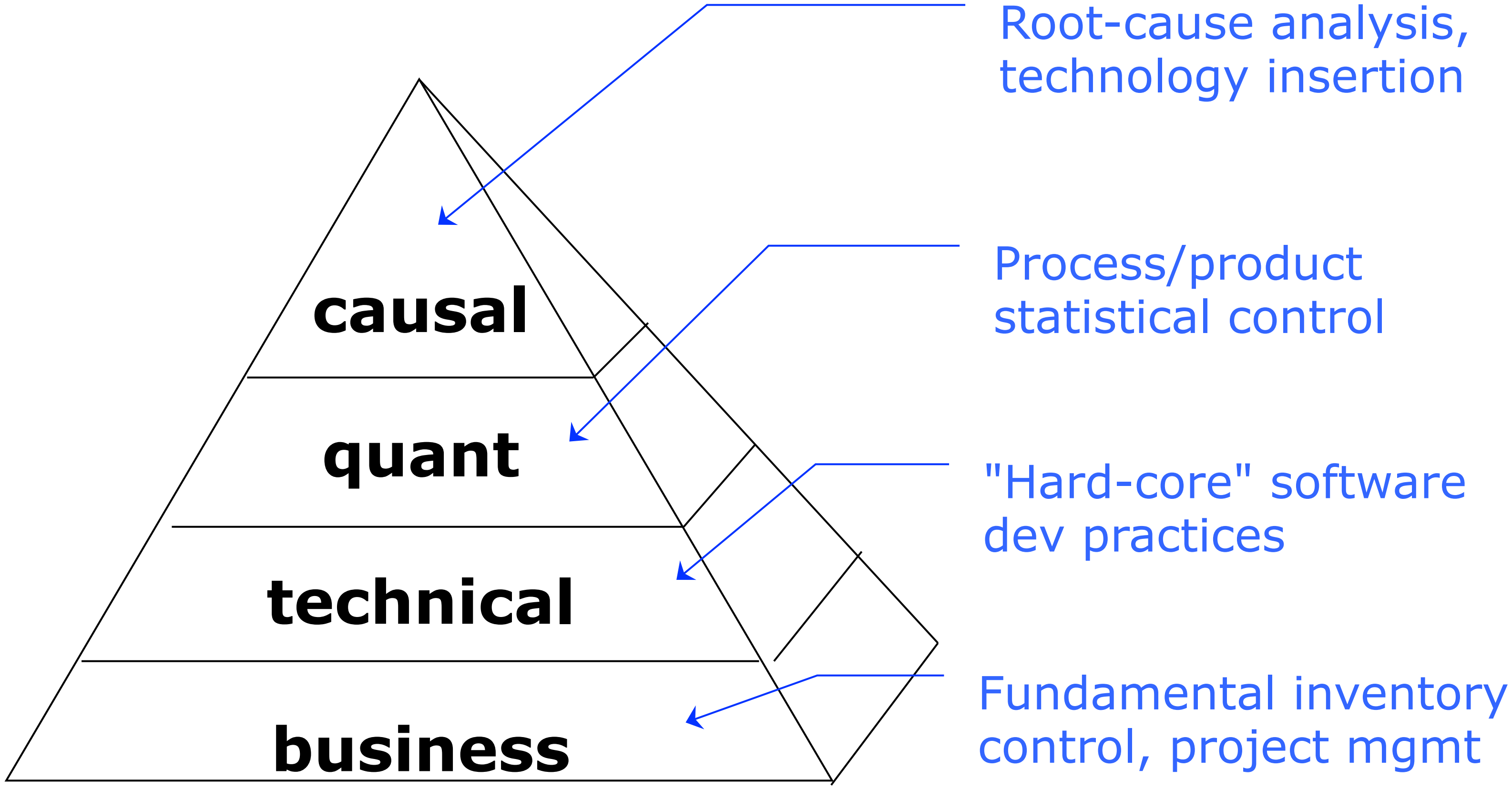
"branded" processes

- definition
  - Prescriptive descriptions of tasks required to develop a software solution
- examples
  - IEEE 1074
  - MIL-STD-498
  - Extreme Programming

instantiate as

SOFTWARE PROCESS

Practice Pyramid


Root-cause analysis, technology insertion
causal
Process/product statistical control
quant
"Hard-core" software dev practices
technical
Fundamental inventory control, project mgmt
business

Each level of the pyramid builds on the practices below it.

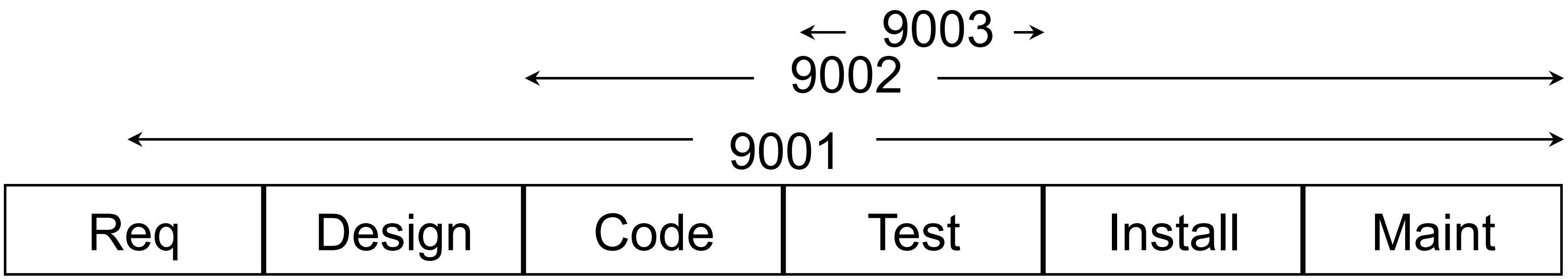
It is possible to carry out upper practices without lower practices, but doing so requires energy.

Process Model Sample

- ISO 9000
  - … defines minimum process requirements that must be met to ensure quality
  - … is a framework: states what, not how
  - written originally for manufacturing, but also applied to software

ISO 9000 (con't)

- collection of individual, but related standards
  - 9000-1 Guidelines for selection and use
  - 9000-3 Guidelines for application of 9001 to software
  - 9001 Model for quality assurance in design, development, production, installation, servicing
  - 9002 Model for QA in production, installation, servicing
  - 9003 Model for QA in final inspection and test
  - 9004 Guidelines for interpretation of 9001, 9002, 9003



ISO 9000 (con’t)

- 9001 Processes (interpreted through 9000-3)
  - **Framework**

    management responsibility
    quality system
    internal quality system audits
    corrective action
  - **Life cycle activities**

    contract review
    purchaser’s requirements spec
    development planning
    quality planning
    design and implementation
    testing and validation
    acceptance
    replication, delivery, and installation
    maintenance
  - **Supporting activities**

    config management
    document control
    quality records
    measurement
    rules, practices, and conventions
    tools and techniques
    purchasing
    included software product training

Process Model Sample

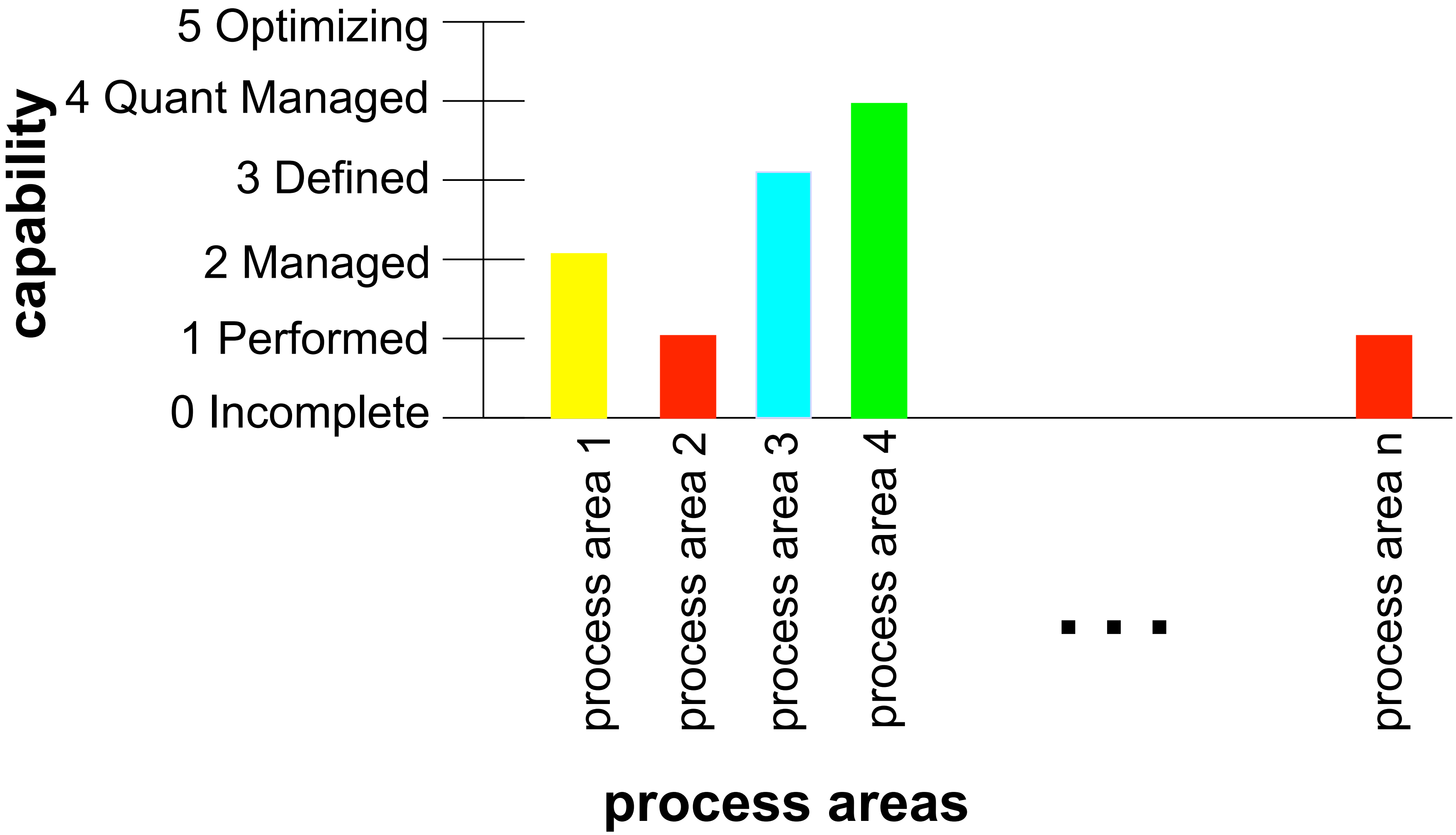
- Capability Maturity Model - Integrated
  - Basics
    - … gauges organizations' ability to predict and control software activities
      - capability = ability to build software
      - maturity = how well-defined and useful
    - identifies processes considered necessary for software production
    - provides a framework for measuring production capability
  - Views
    - continuous
      - Q: how well am I performing various software functions?
    - staged
      - Q: how well can I control cost, schedule, performance?
  - Misc
    - is called "integrated" because it is integrated with other CMMs

CMMI Process Areas - Continuous

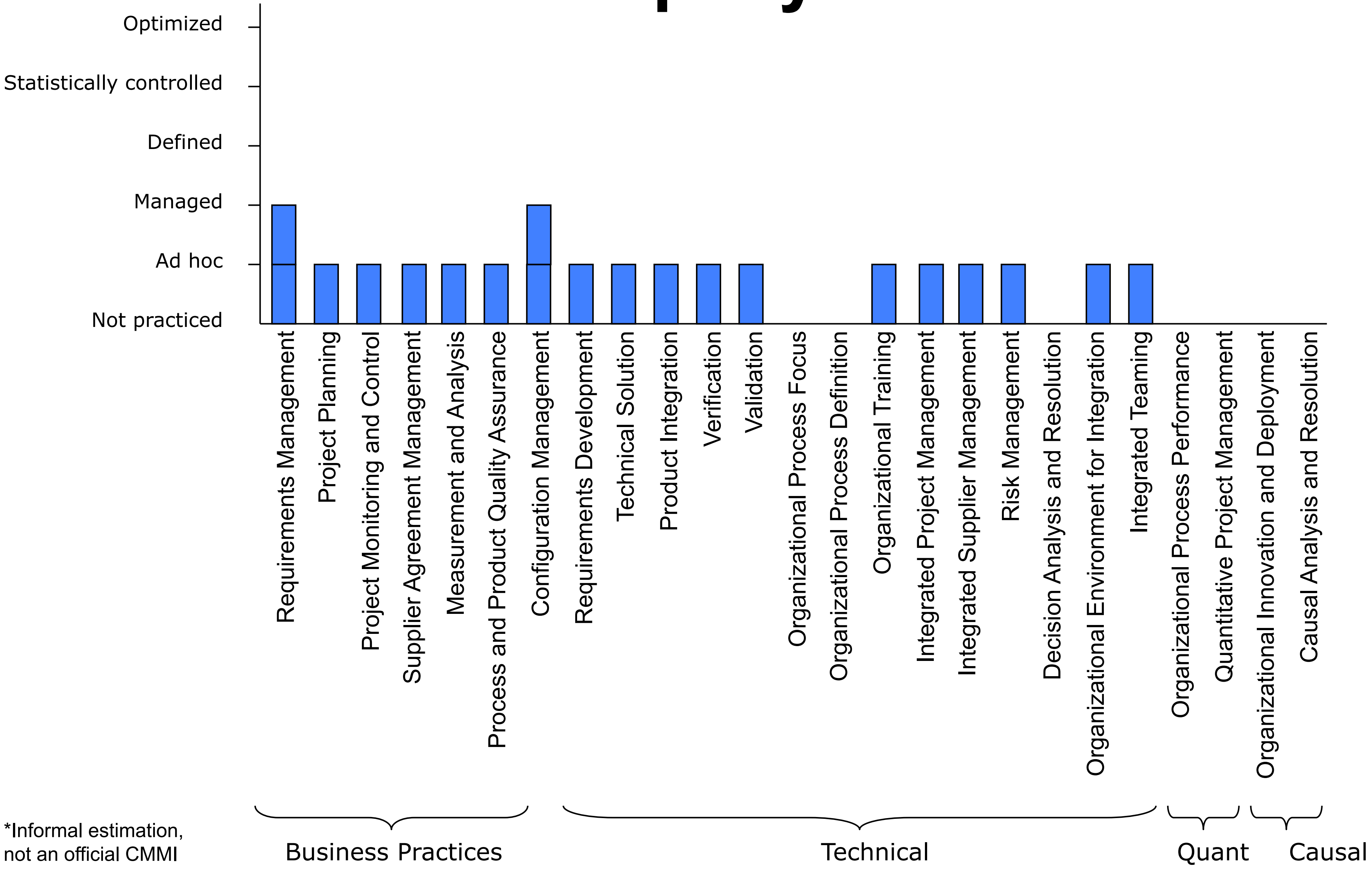
Category	Process Area
Project Management	Project Planning Project Monitoring and Control Supplier Agreement Management Integrated Project Management Risk Management Quantitative Project Management
Support	Configuration Management Process and Product Quality Assurance Measurement and Analysis Causal Analysis and Resolution Decision Analysis and Resolution
Engineering	Requirements Management Requirements Development Technical Solution Product Integration Verification Validation
Process Management	Organizational Process Focus Organizational Process Definition Organizational Training Organizational Process Performance Organizational Innovation and Deployment

CMMI (con't)

- Continuous representation

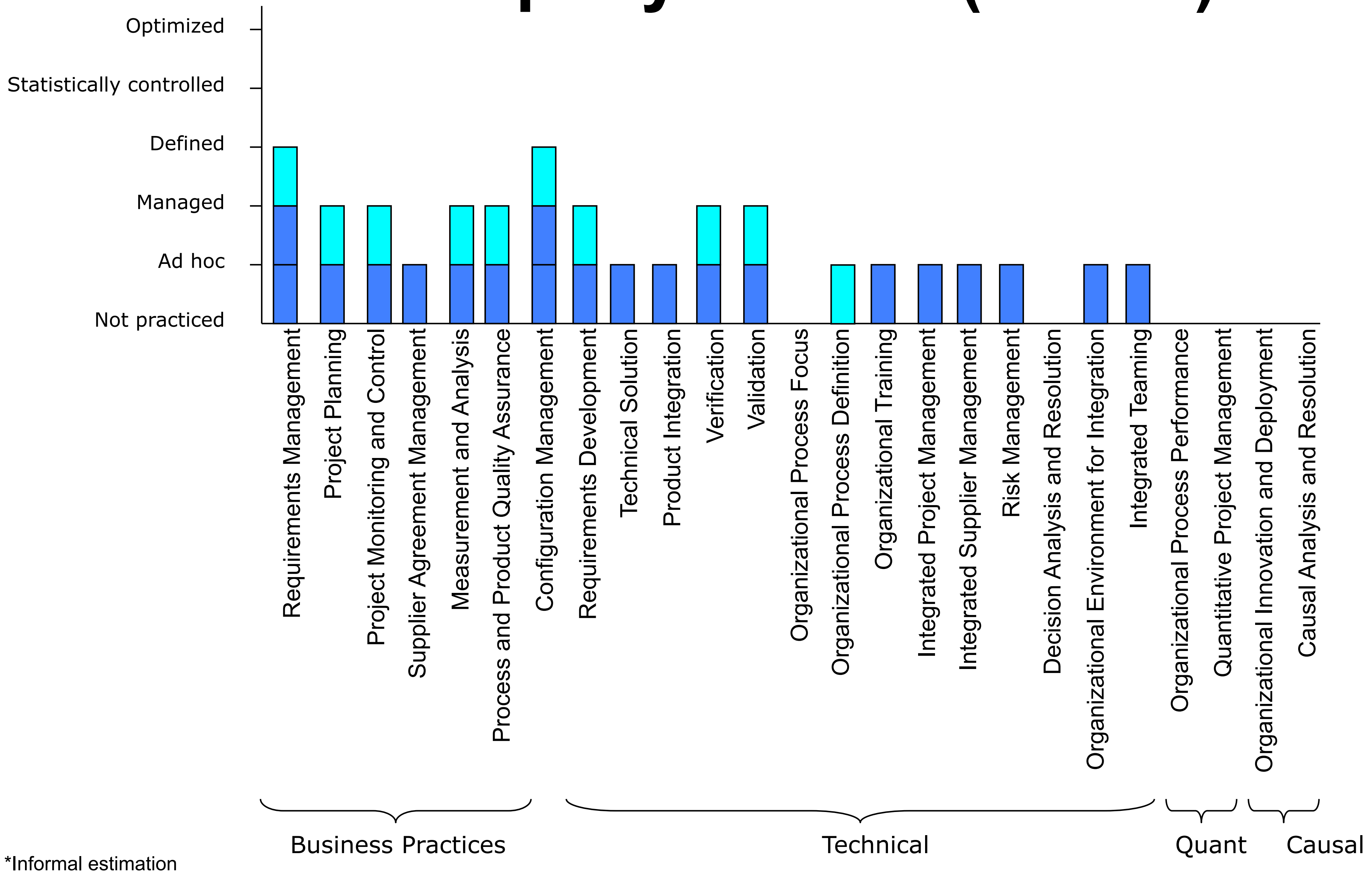


XXXX Company - As Is*



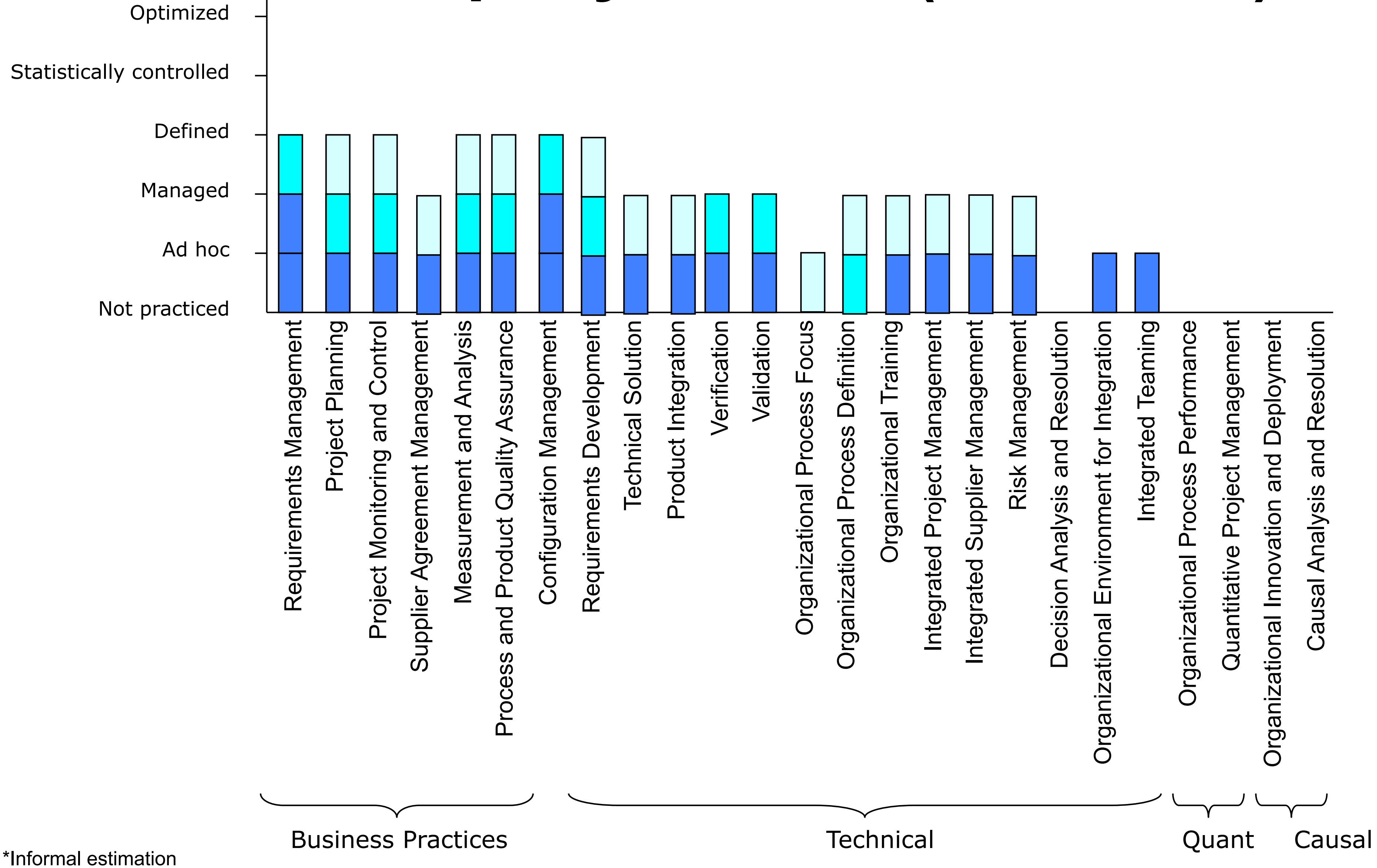
*Informal estimation, not an official CMMI assessment

XXXX Company - To Be (1 Year)



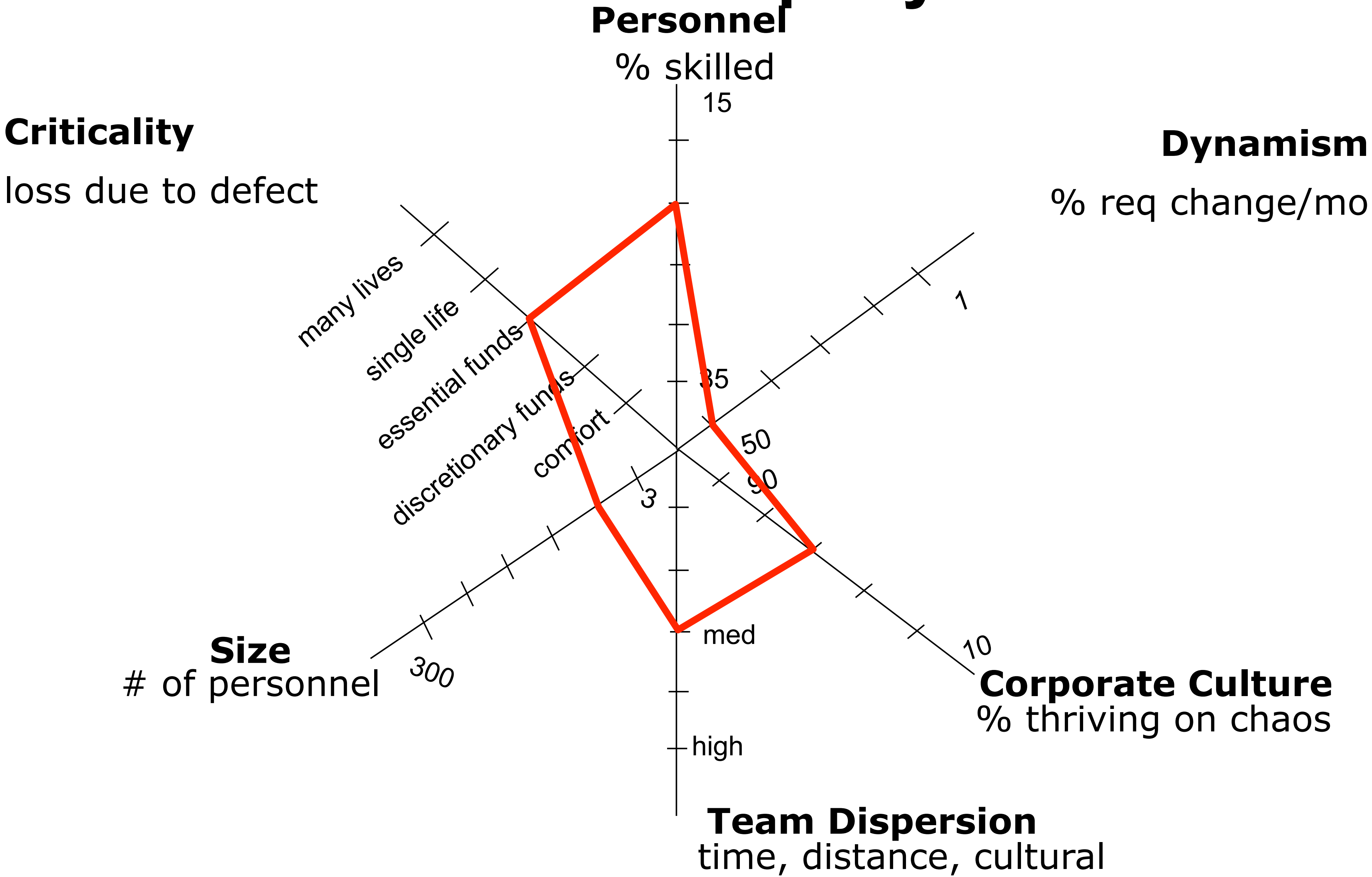
*Informal estimation

XXXX Company - To Be (2-3 Years)



*Informal estimation

XXXX Company

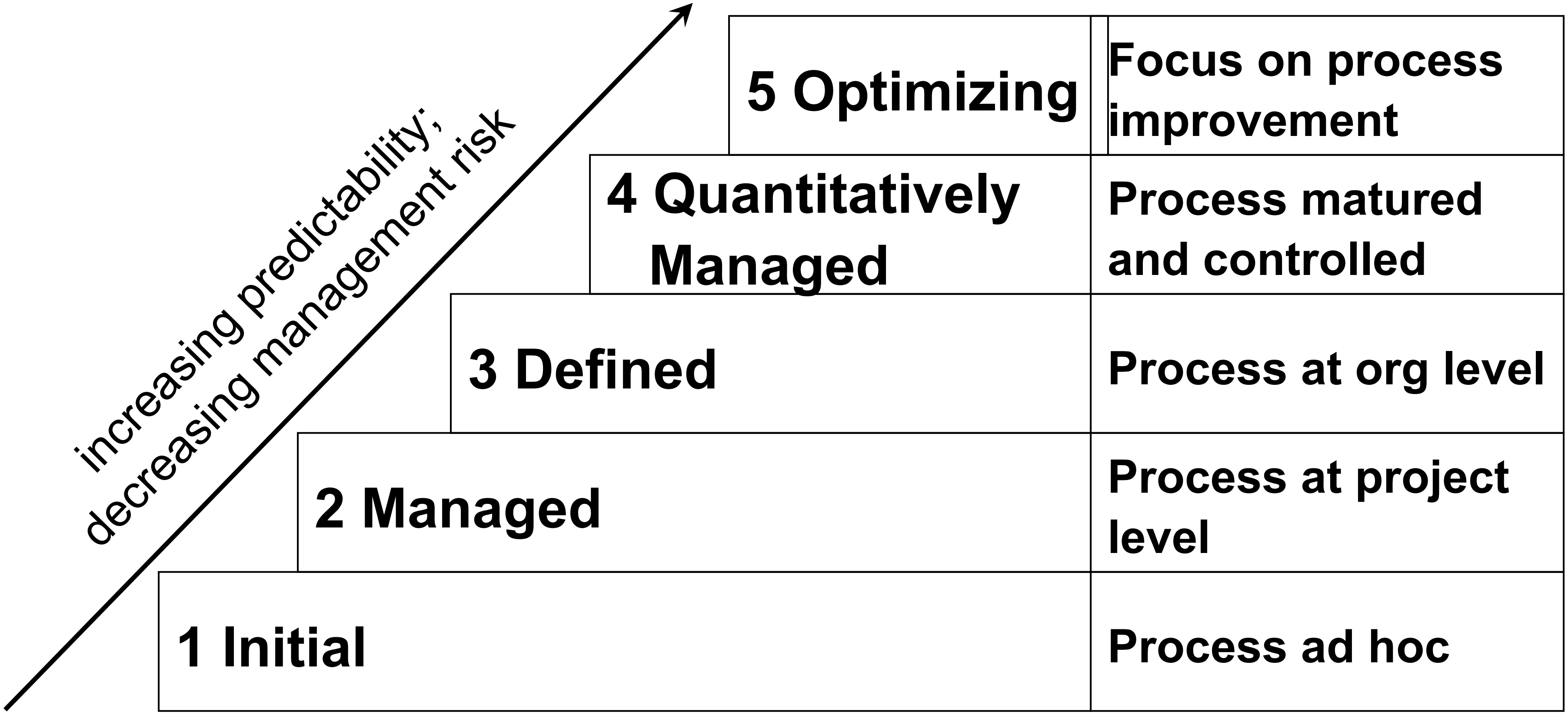


CMMI Process Areas – Staged

Level	Focus	Process Areas
5 Optimizing	*Continuous process improvement*	Organizational Innovation and Deployment Causal Analysis and Resolution
4 Quantitatively Managed	*Quantitative management*	Organizational Process Performance Quantitative Project Management
3 Defined	*Process standardization*	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Risk Management Decision Analysis and Resolution
2 Managed	*Basic project management*	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management
1 Performed		

CMMI (con't)

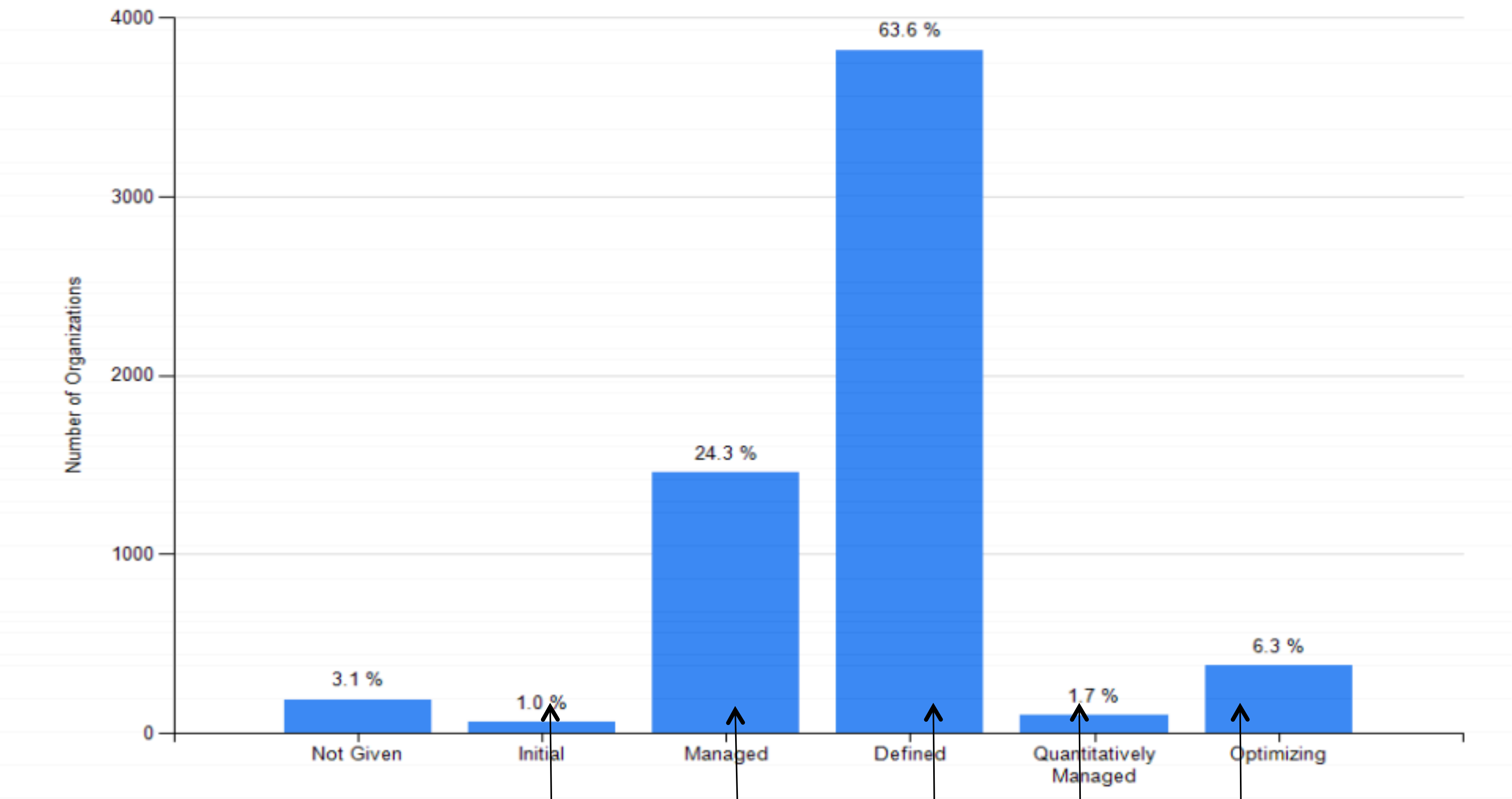
- Staged representation



increasing predictability; decreasing management risk

5 Optimizing	Focus on process improvement
4 Quantitatively Managed	Process matured and controlled
3 Defined	Process at org level
2 Managed	Process at project level
1 Initial	Process ad hoc

Process Maturity Profile by All Reporting Organizations



Based on the most recent appraisal of 6,010 organizations

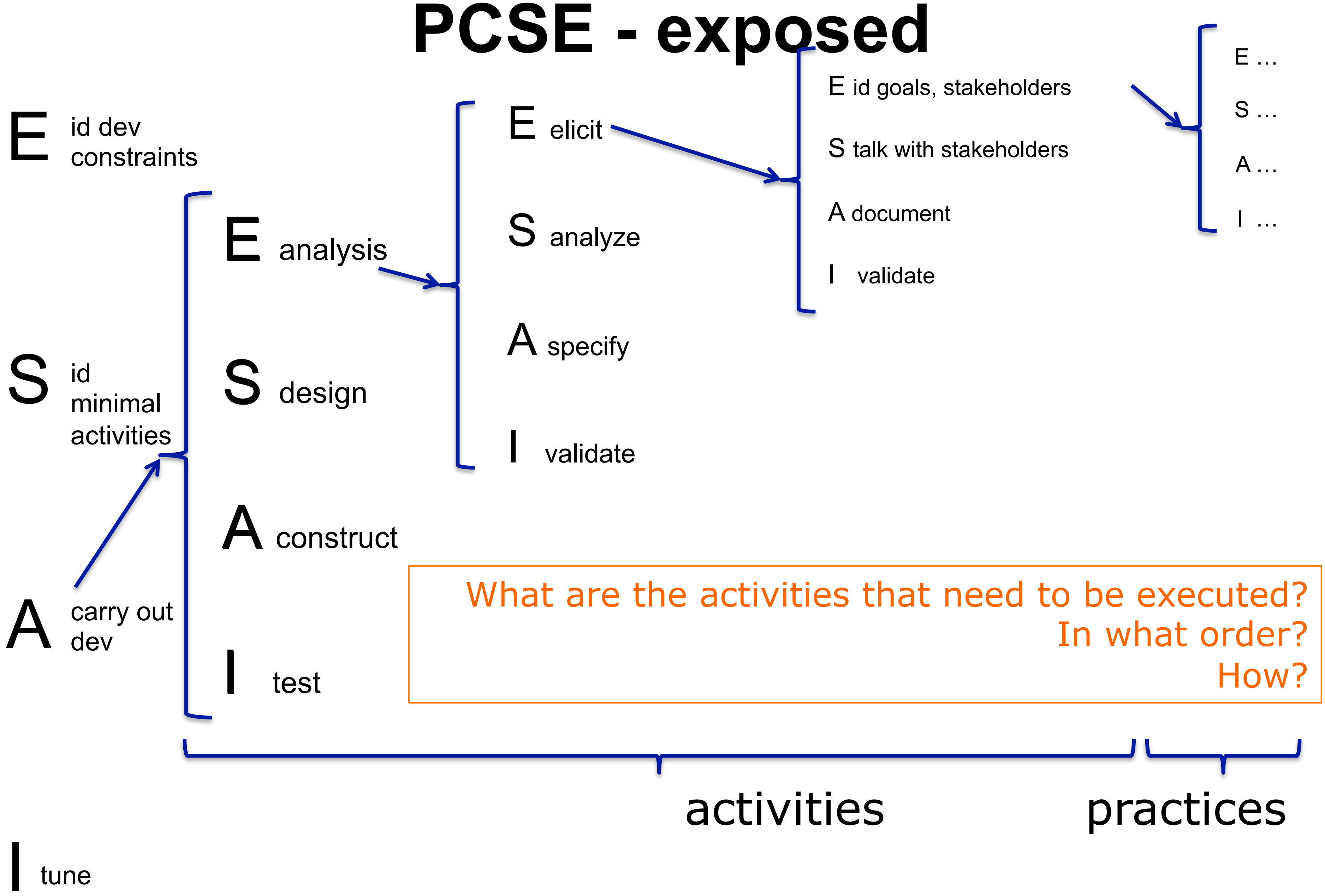
In 1987-91 timeframe:	80.2%	12.1%	7.6%	0%	0%

* Note: not a totally accurate comparison, but fair enough for discussion

Our Approach This Semester

- PCSE
  - – Practice-Centered Software Engineering
    - marriage of Personal Software Process, Scrum, Extreme Programming, Feature-driven Development
    - instantiation of CMMI at personal level, scalable to small teams
  - – COMP 5700/6700/6706 approach
    - introduce PCSE in several upwardly compatible steps
    - write small programs at each step
    - gather and analyze data on work
    - use data and analyses to gain insight into process management

PCSE - exposed



E id dev constraints

S id minimal activities

A carry out dev

I tune

E analysis

S design

A construct

I test

E elicit

S analyze

A specify

I validate

E id goals, stakeholders

S talk with stakeholders

A document

I validate

E …

S …

A …

I …

What are the activities that need to be executed?
In what order?
How?

activities

practices

Minimal Guiding Indicators

A statement of project goals and the means by which we know if we've achieved them.

Typically expressed as cost, schedule, performance.

Cost indicators
- Get Rich
  - $ income > $ outgo

MSAs describe how objectives will be achieved

Minimally Sufficient Activity

The least possible set of conceptual lifecycle activities needed to produce software of a given quality.

- Engineering
  - Envision
    - Req Analysis
  - Synthesize
    - Design
  - Articulate
    - Code
  - Interpret
    - Test
- Operational
  - Plan
  - Monitor
- Business
  - Market
  - Sell
- Legal
  - Sue

Minimally Viable Process

Orders MSAs so as to structure the effort to be as non-invasive as possible, yet provide enough structure to be viable.

Done by listing the lowest-level MSAs and determining work flow.

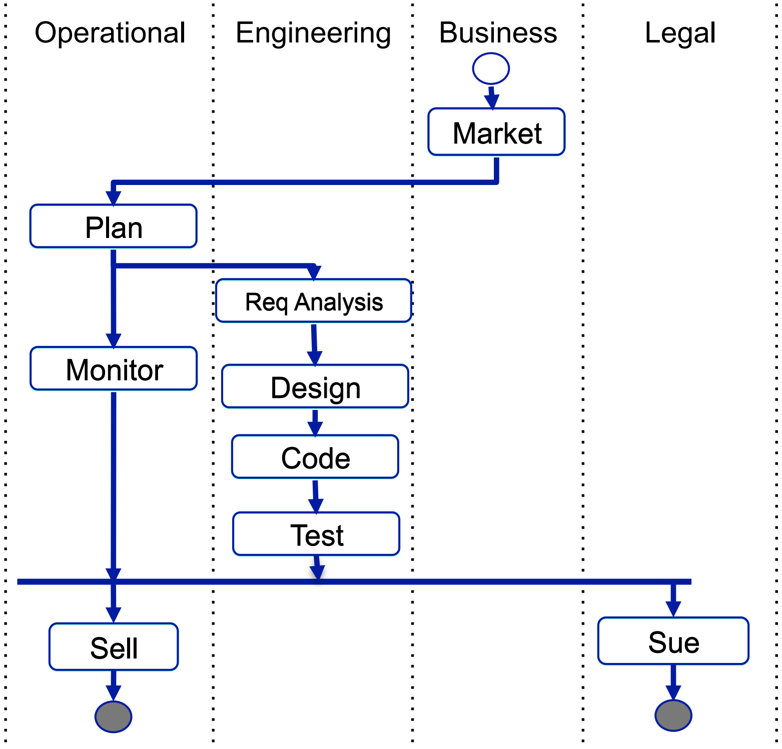
MVP identifies the relationship of the MSAs

MEPs describe how MVAs will be carried out

Minimally Effective Practice

Instructs developers in what to do.

Done by specifying lowest-level MSAs



MSA	MEP
Req Analysis	ad hoc
Design	Larman
Code	ad hoc
Test	Smoke
Plan	ad hoc
Monitor	ad hoc
Market	bill board
Sell	Google Play
Sue	ad hoc

Having identified this information, we have a basis for alleviating software development pain points.

Summary

Topics

- Process foundations
- Processes a la SwE
- Processes explored
- Samples

Next time: Common process elements

Key Points

- The manufacturing community discovered processes long ago
- SwE then = technical activities SwE now = process orientation
- A process is a set of tasks
- Processes have benefits: most notable is management containment
- Process models describe what to do; "branded" processes describe how to do it. Many exist