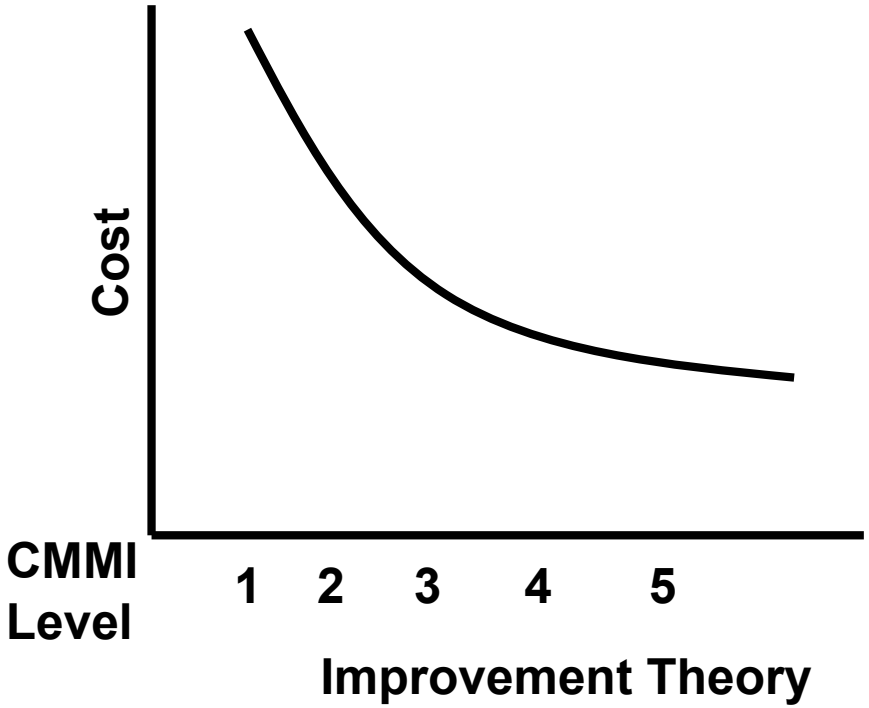
What Should Software Engineering Consist of?
An Industry Experience Perspective

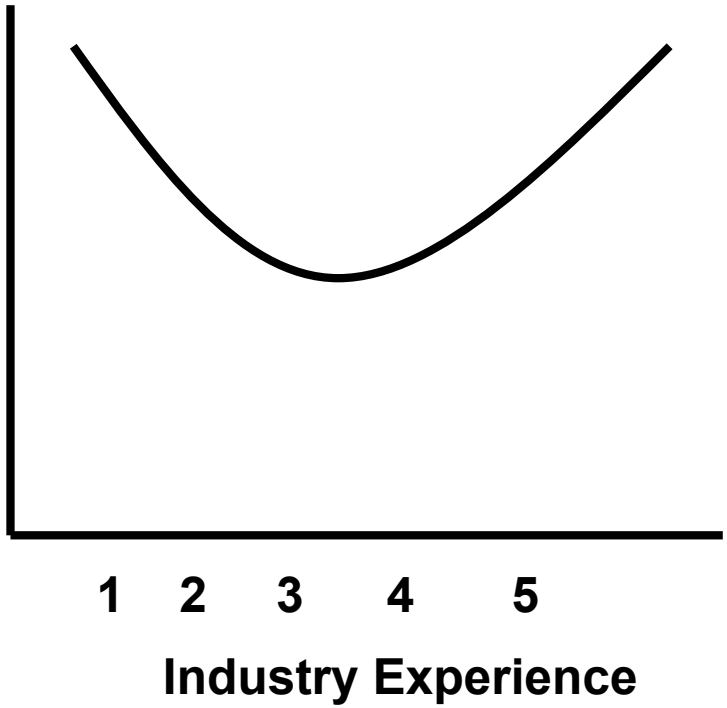
Paul E. McMahon, Principal

PEM Systems

pemcmahon@acm.org

Observation





Theory indicates as CMMI maturity rises so should productivity

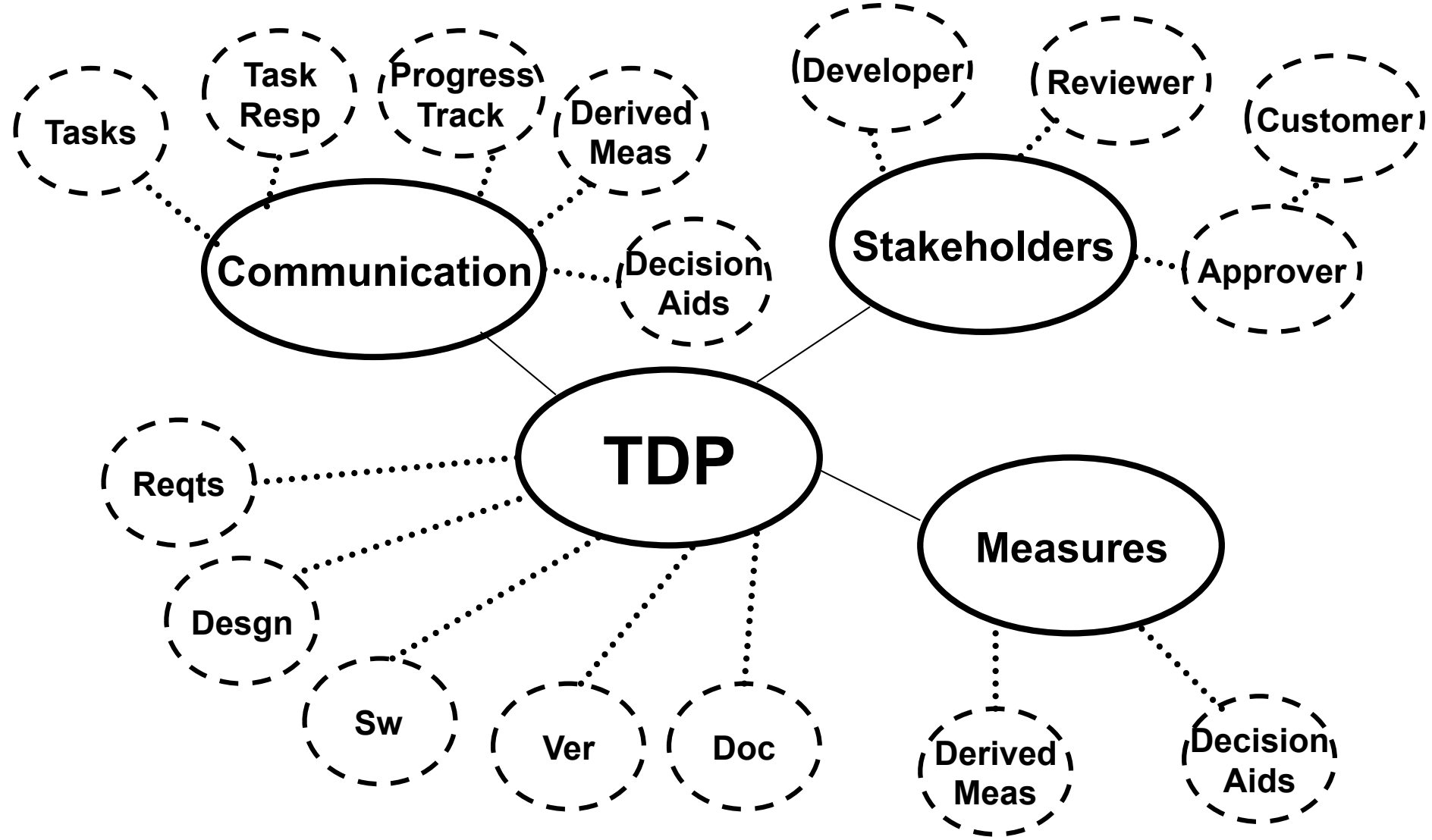
Experience beyond level 3 In some organizations is the reverse

Is theory underlying CMMI wrong?

Position and Challenge

- Position
  - CMMI model based on solid theory
  - Not doing adequate job bridging theory to practice
- Challenge:
  - This is where Software Engineering & SEMAT should help

Paper presents 4 Core element kernel supporting position



Focus here is Motivation..

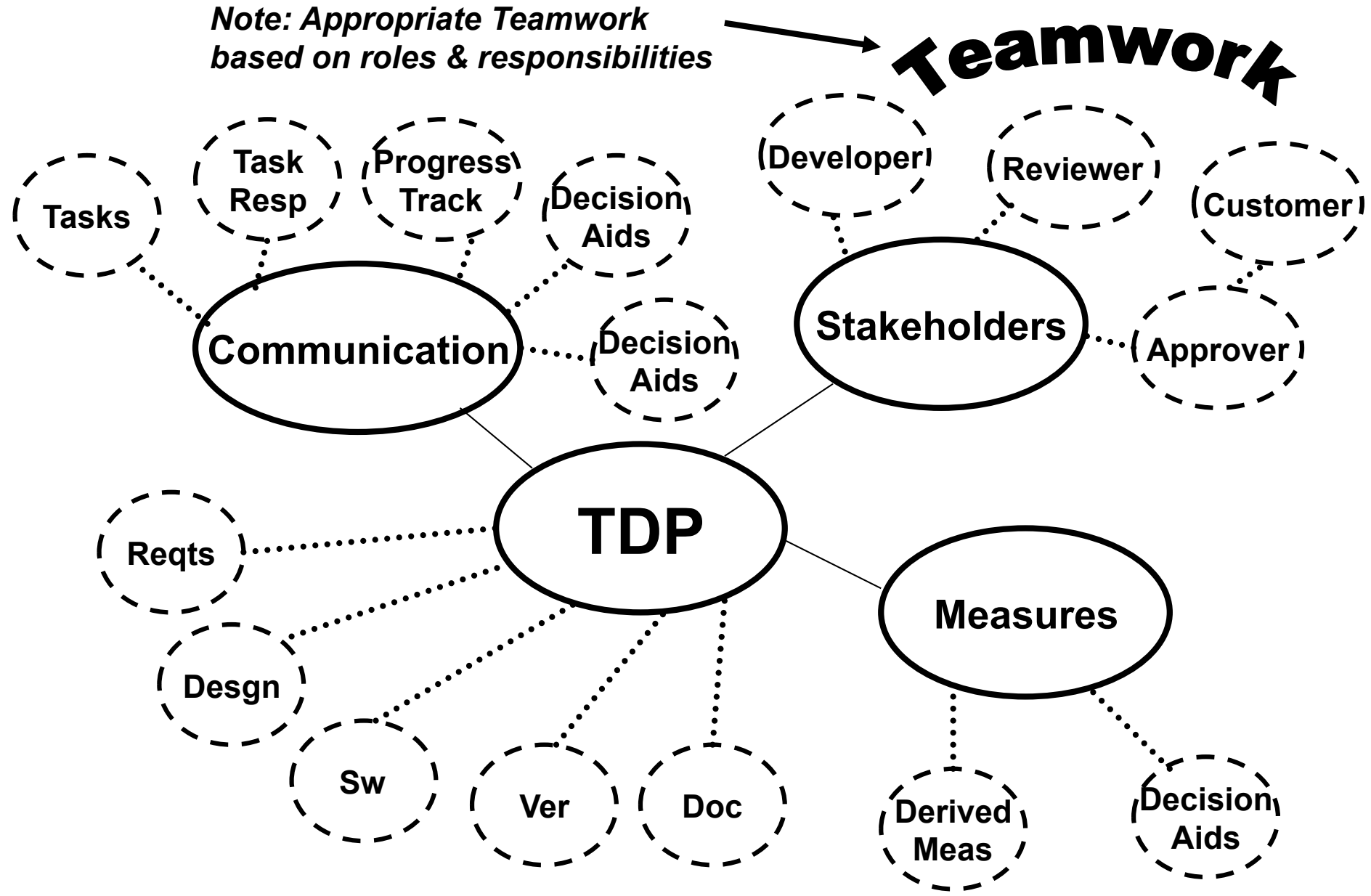
1st Kernel Element:
Technical Data Package (TDP)

- TDP is "center-piece" of the kernel
  - Think of this as whatever the customer is buying
  - Kernel must be "customer product centric"
- Motivation:
  - Goal to satisfy customer
  - Everything else must be justified in terms of contribution to goal

2nd :
Simple Stakeholder element

- Based on clearly defined roles & responsibilities

- Motivation:
  - Common root cause of immature software practices today is failure to involve right people at right time
  - This is where Teamwork fits in the kernel
  - People need help with “decisions” related to who to involve and when
  - This is where Software Engineering (and SEMAT) should help

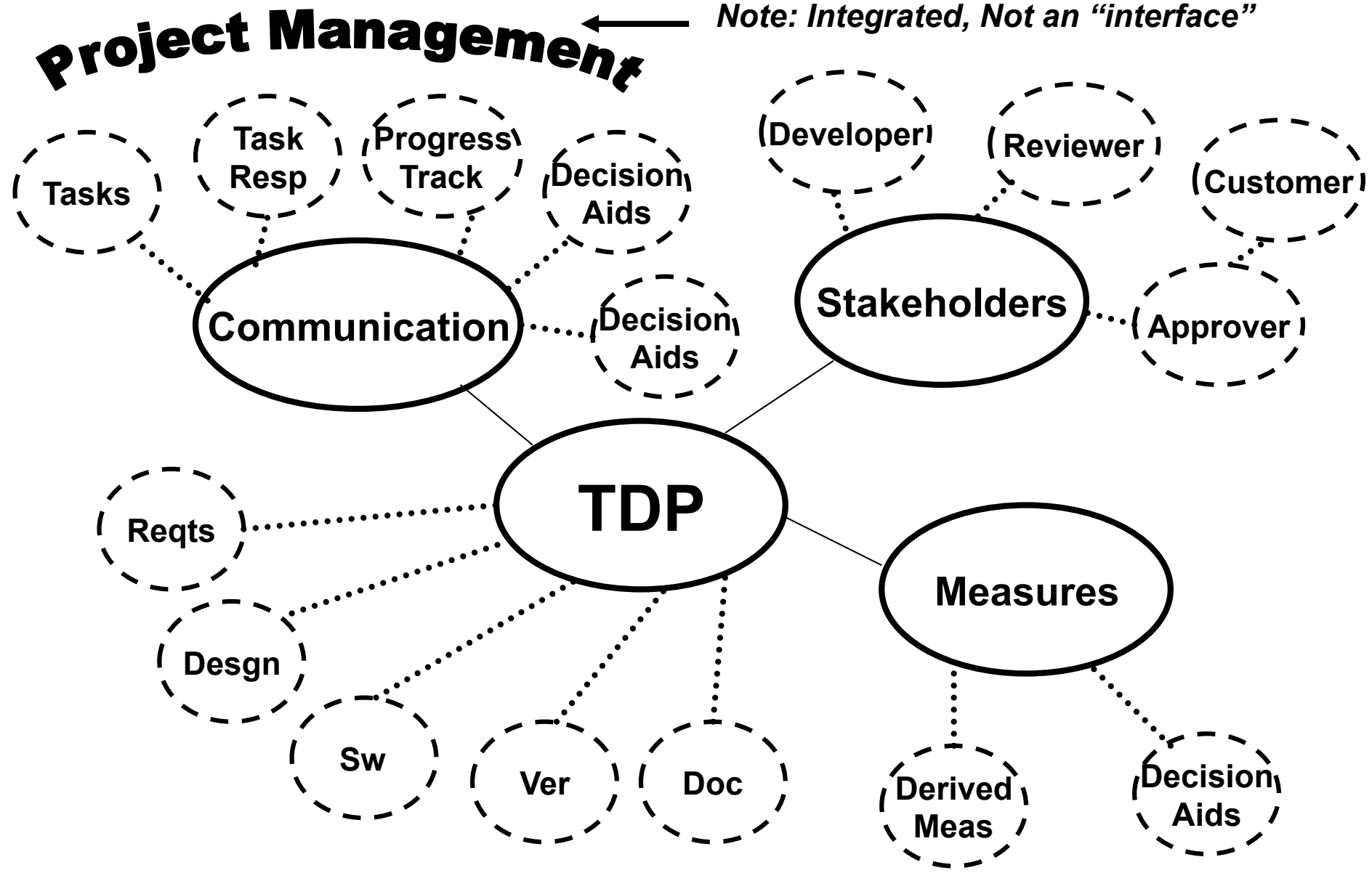


..In context of product..

3rd : Communication

- Think of this element as project management
  - Must be “integrated”, not an interface

- Motivation:
  - Another common cause of “immature practices” is failure to communicate current accurate task status, including decisions faced and risks
  - People need help articulating options, potential consequences & rationale for decisions
  - This is another area where Software Engineering (and SEMAT) can help



..In context of product..

4th :
Measurement

- Measurement is another example where <u>failing to "bridge"</u> theory to practice

- Watts Humphrey provided solid theory in "*A Discipline for Software Engineering"* written over 15 years ago

  – Importance of "***<u>deriving" context specific</u>*** measures emphasized

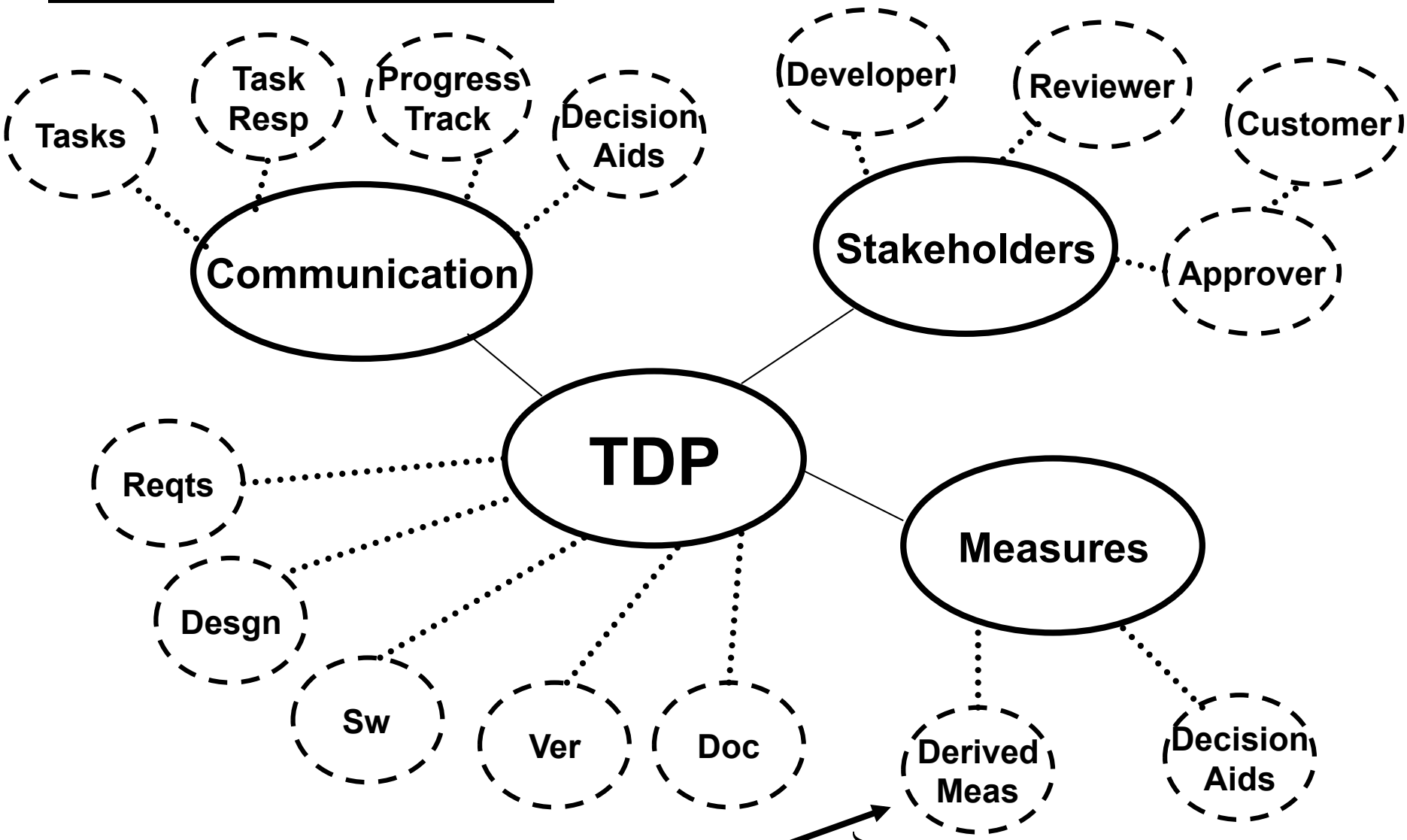
- Yet today <u>some</u> CMMI level 5 organizations continue to collect "***<u>standard measures</u>***" that are not providing the intended value

We Can Do Better & We Are Better

- There exist great organizations that are doing it right today

  - E.g. some have automated parts of their software process through domain-specific solutions and tools simplifying their most common occurring decisions

..*In context of product*..



Note: Domain-Specific, Product-Specific, Project-Specific

What Does it Mean to "Engineer" Software?

- Alistair Cockburn has suggested when we think of "engineering" software we should think about the "decisions" and "tradeoffs"

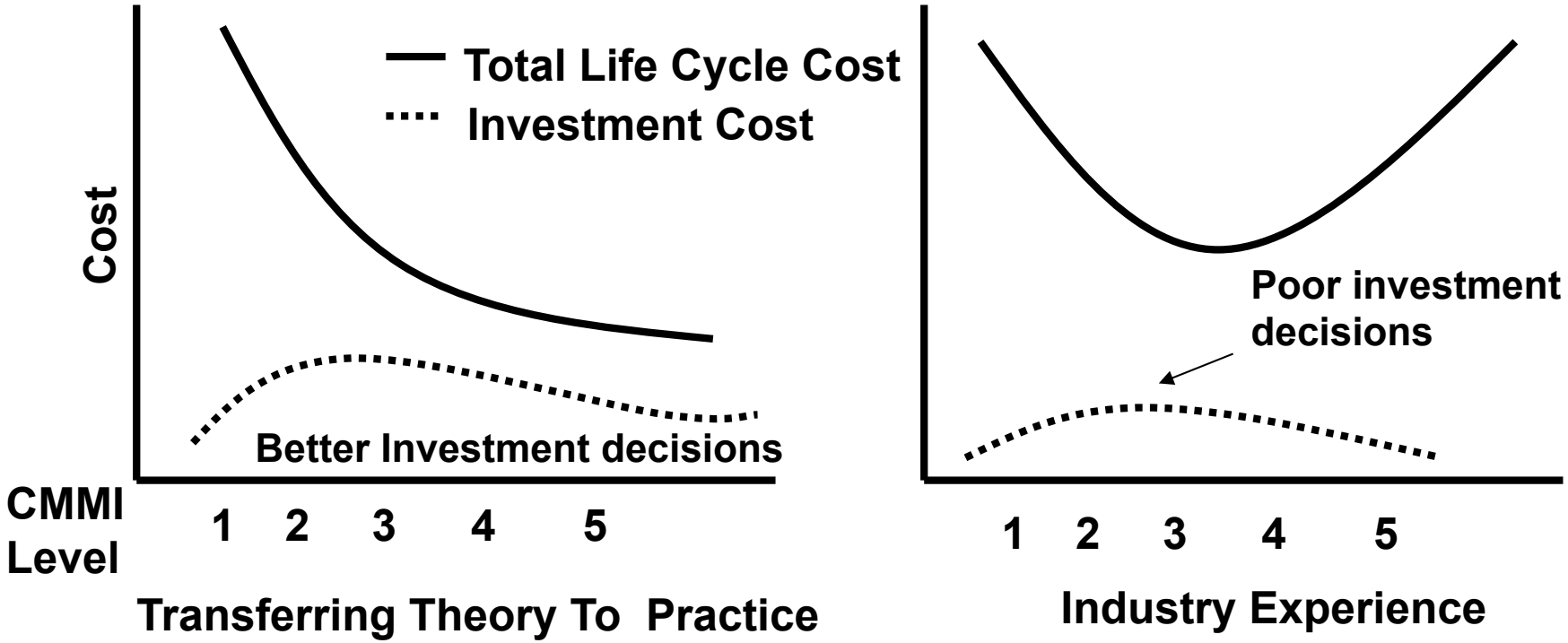
- Software Engineering should provide more help in "how-to engineer software"

  - Not to give you the answers (e.g. object-oriented)

  - But to help with the "reasoning process" to find the right answer given your specific project and product conditions

    - People need better "decision-guidance"

How SEMAT Can Help



Need Better "Decision-Guidance" based on Real Factors Specific to Environment faced

Position Summary

- Caution against developing a “new theory”
- We are further ahead than many realize
- Challenge: Extract what we know works along with the reasoning process behind it & then institutionalize it as part of what it means to “engineer software” by making better decisions