Let’s Build a Smarter Method

SDLC 3.0: A Complex Adaptive System of Patterns

Mark Kennaley

Perspective from Industry, on Industry

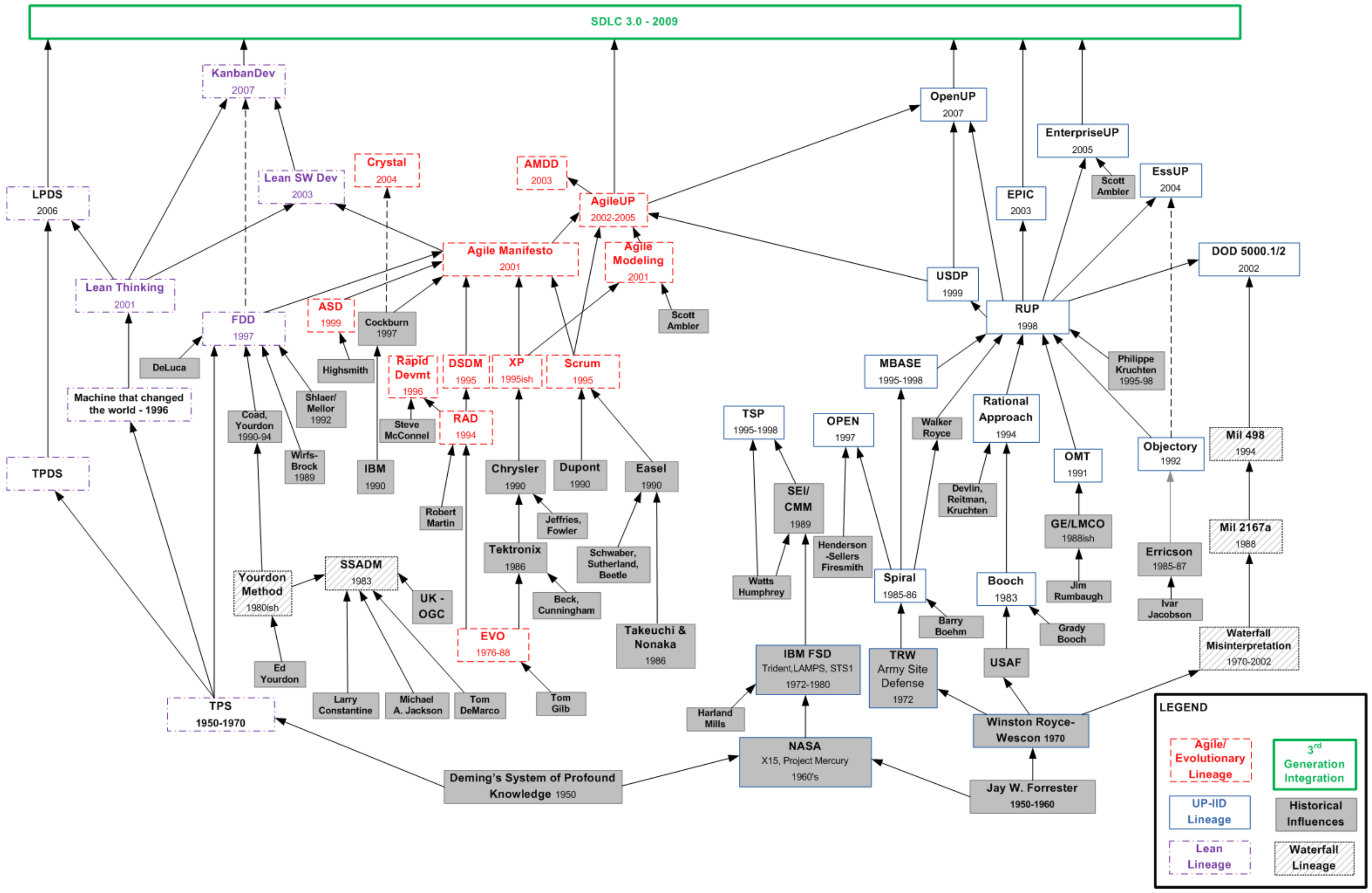
- Given the SEMAT Call for Action…3 key points

- **Why has the industry methodologists allowed a branching anti-pattern?**
  - Cascading branches, lack of integration;

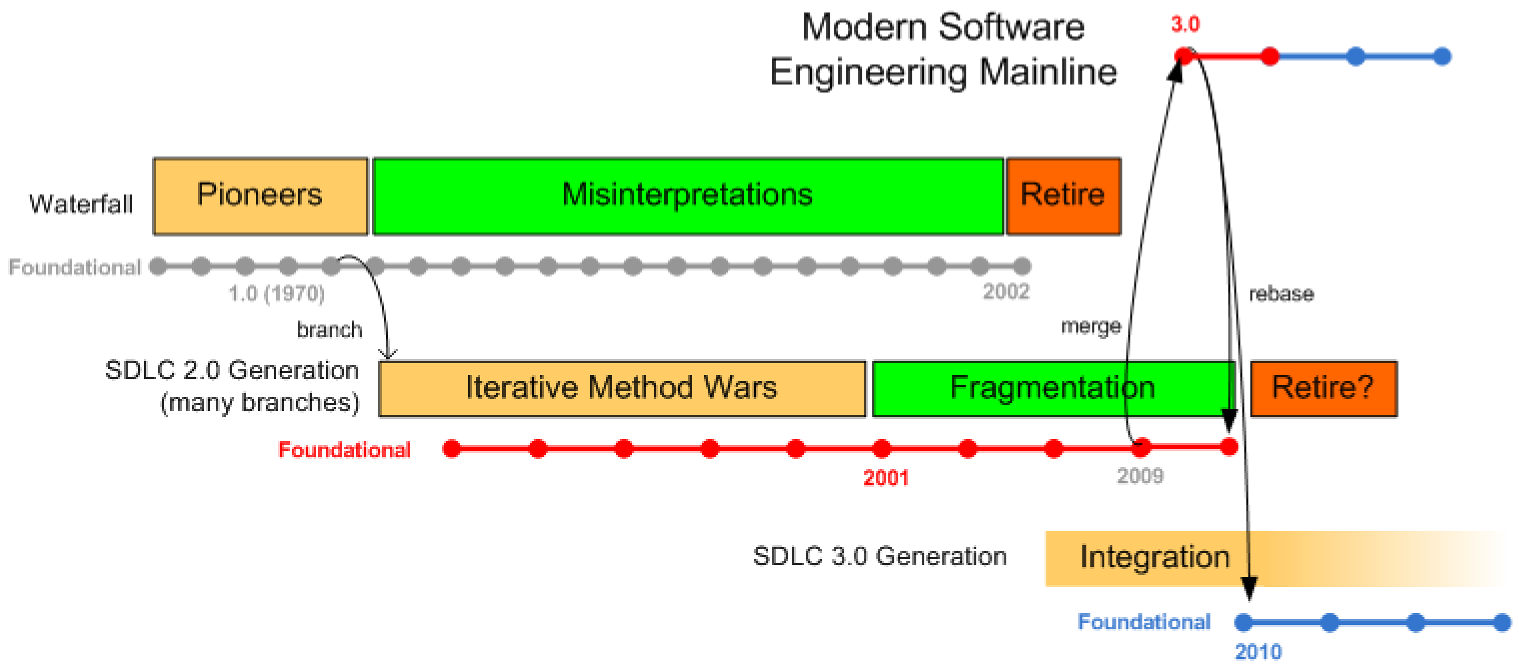
- **What happened to the (organizational) pattern movement?**
  - Pattern ≈ Practice
  - Pragmatic level of abstraction because it is experience based, generally agreeable between isolated branches
  - Pattern decomposition of commonplace methods enables integration

- **Why hasn't Control Systems Engineering been leveraged for a foundation of study of software product delivery dynamics?**
  - PID Control, Adaptive Control, Stochastic Control to study potential influences on CAS

End the Iterative Method Wars



SDLC 3.0: Time for a Re-base



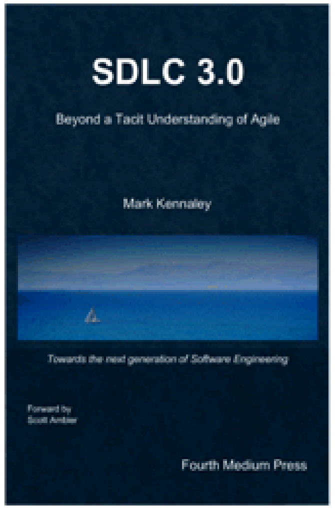
- **We know the longer we let a code-line drift, the harder it is to integrate**

Pattern ≈ Practice

- Pattern = a solution to a problem in context
- Practice = technique effective at delivering effective outcome
- Each approach from the Iterative Methods has something to offer (in context)
- Instead of competing methods (wasteful), integrate practices (wise)
- Patterns are the key to integration – a Complex Adaptive System of Patterns

SDLC 3.0: Universals do Exist

- Domain model of patterns
- Common ground exists
- Community specific innovations, as well as synonyms and re-branding



SDLC 3.0

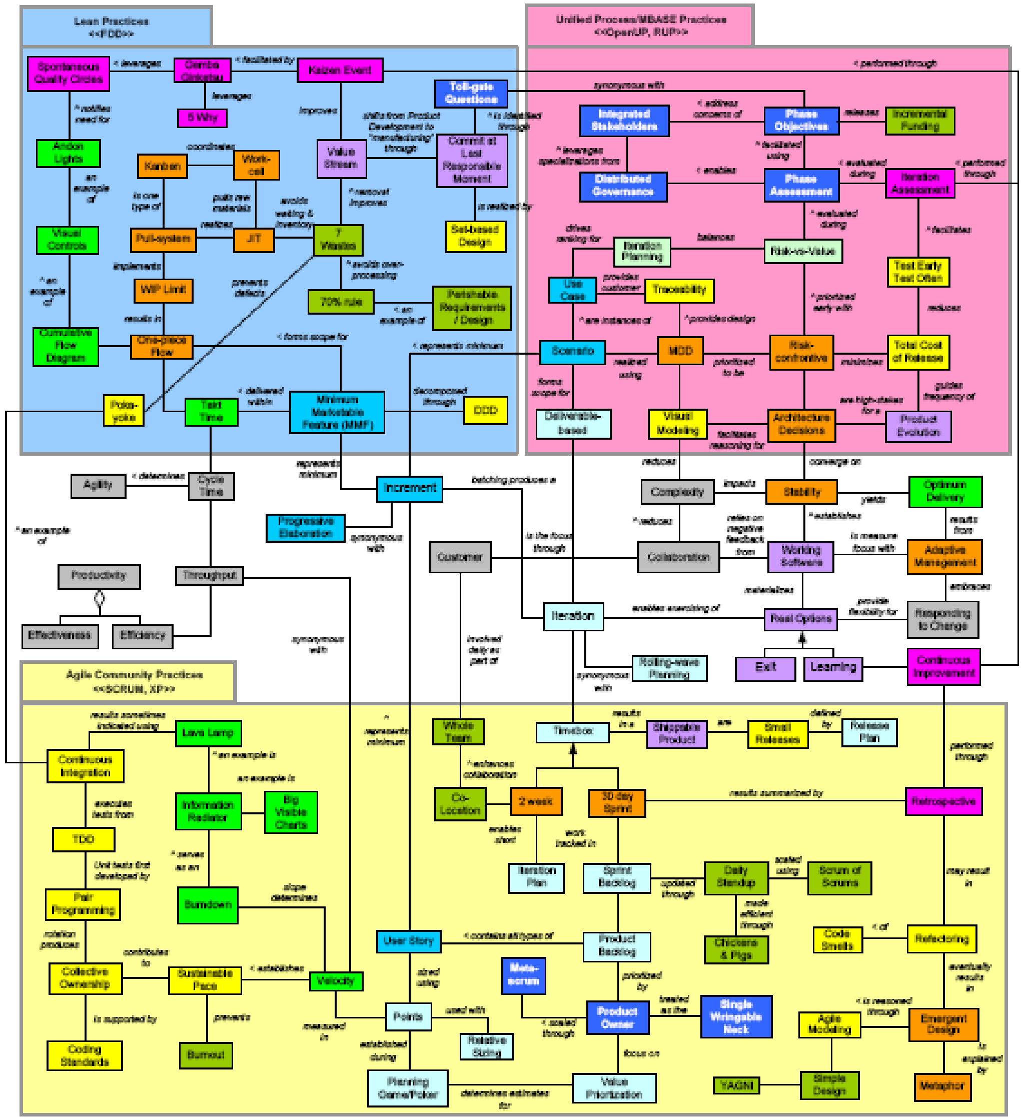
Beyond a Tacit Understanding of Agile

Mark Kennaley

Towards the next generation of Software Engineering

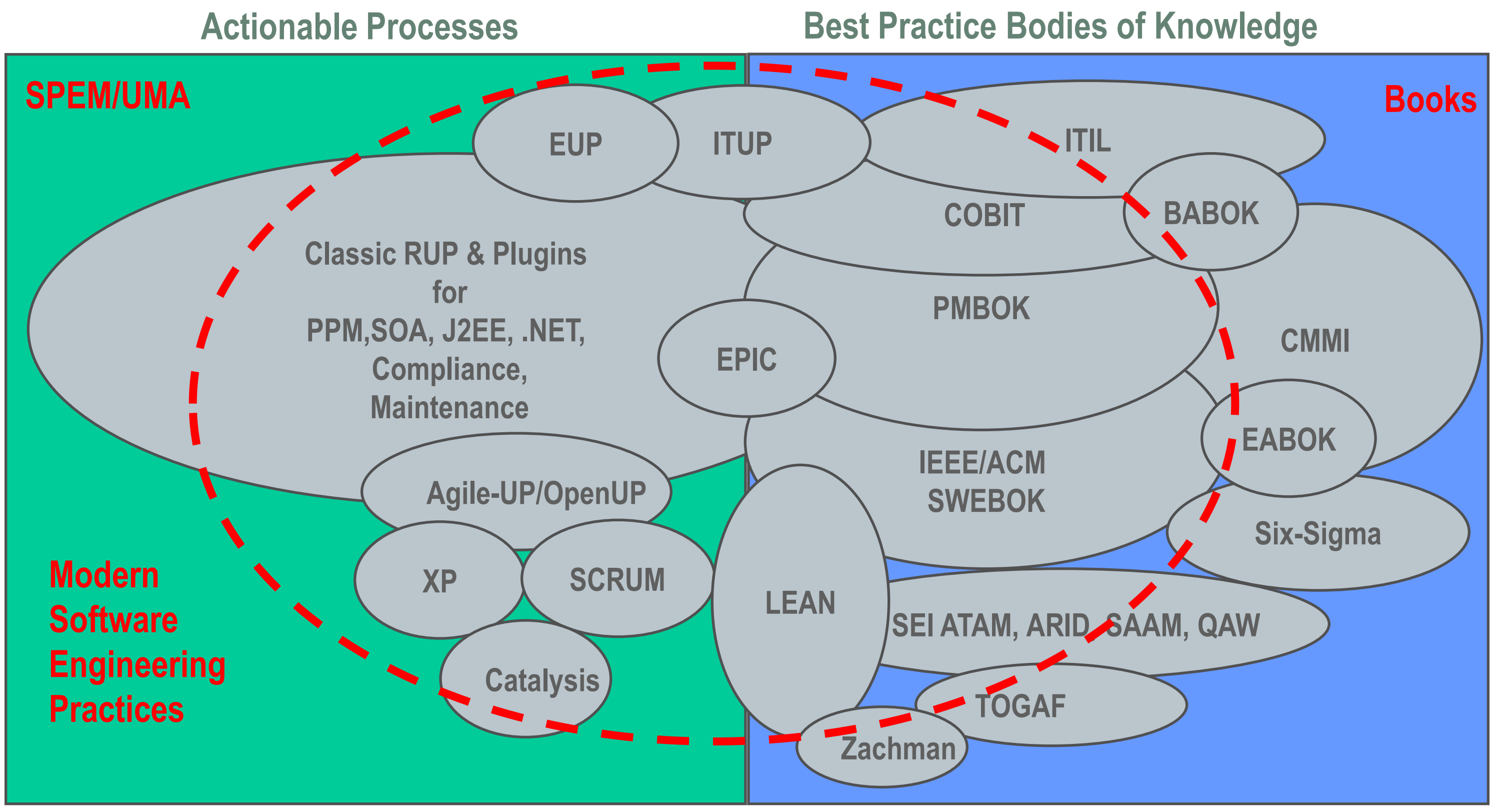
Forward by
Scott Ambler

Fourth Medium Press



State of Software Engineering – Many different Perspectives

- Modern Software Engineering landscape involves broad and diverse, sometimes overlapping and conflicting, and most of the time bloated bodies-of-knowledge

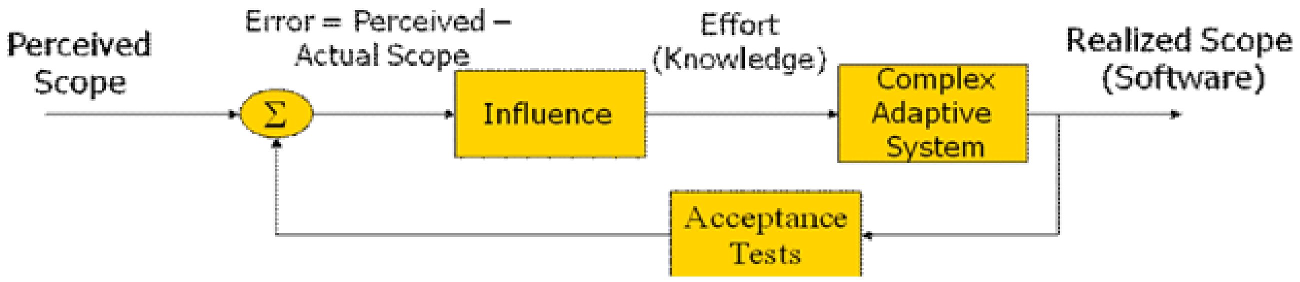

Actionable Processes
Best Practice Bodies of Knowledge
SPEM/UMA
Books
EUP
ITUP
ITIL
COBIT
BABOK
Classic RUP & Plugins for PPM,SOA, J2EE, .NET, Compliance, Maintenance
EPIC
PMBOK
CMMI
EABOK
IEEE/ACM SWEBOK
Agile-UP/OpenUP
Six-Sigma
XP
SCRUM
LEAN
SEI ATAM, ARID, SAAM, QAW
Catalysis
TOGAF
Zachman
Modern Software Engineering Practices

Making Change Happen

- **Everyone "Knows" - Tacit Knowledge**
  - Yet lack of Trust – no wonder there is so much "us-vs-them" in software engineering!
  - Biggest waste in IT

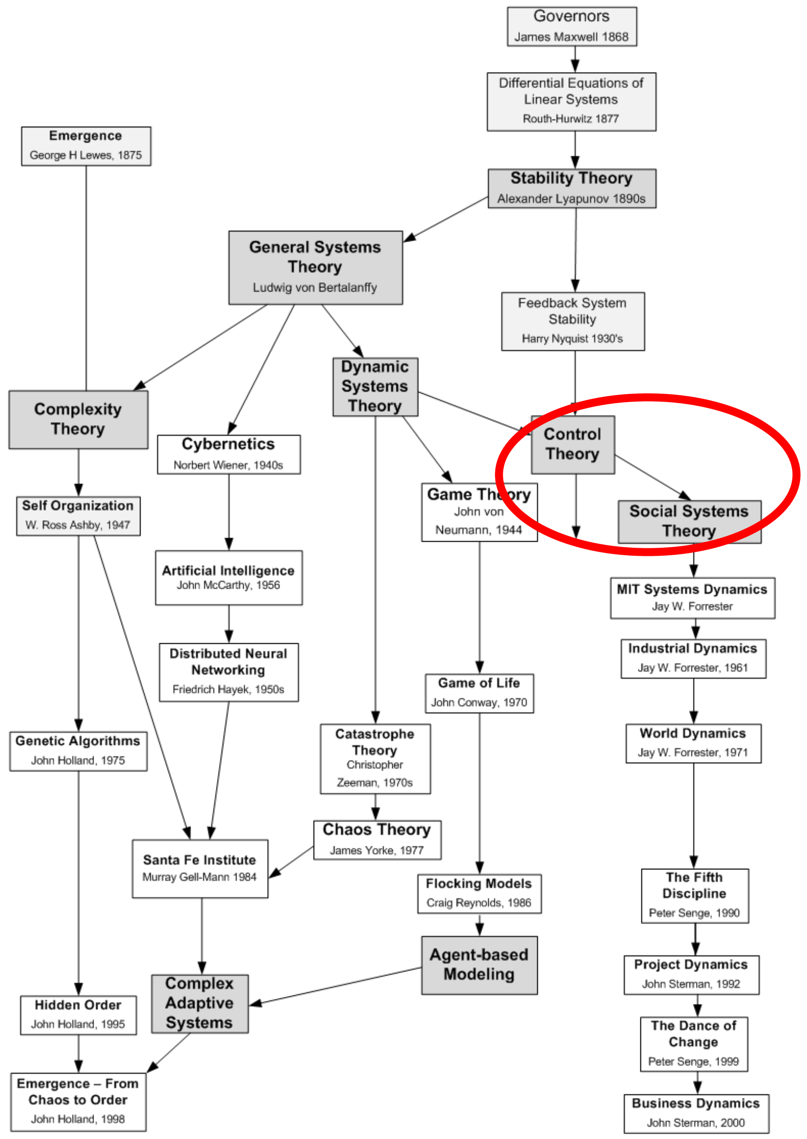
- **Consolidate the centrists – pull an Obama**
  - "The change we seek is the change we need – yes we can"

- **We need a foundation to help people understand "why", and in what context**

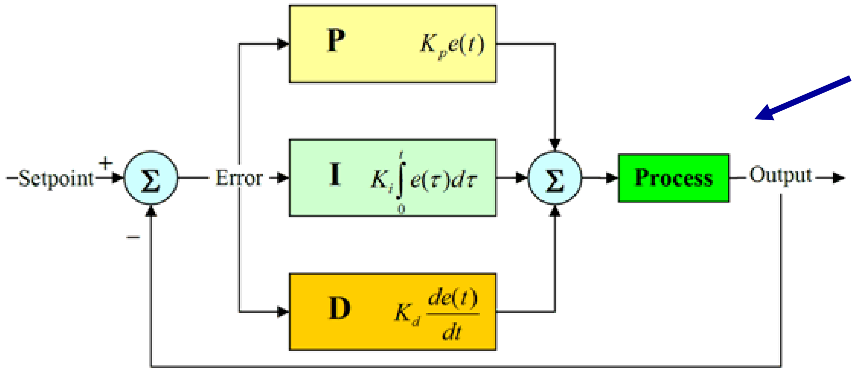


Beyond Superficial Systems Thinking

- Alistair Cockburn proposes Game Theory
- Others identify with CAS / Chaordic Systems
- The very popular Scrum approach mentions control theory in passing
- **Why not Control Systems Theory as a foundation?**



PID Control – A foundation for study



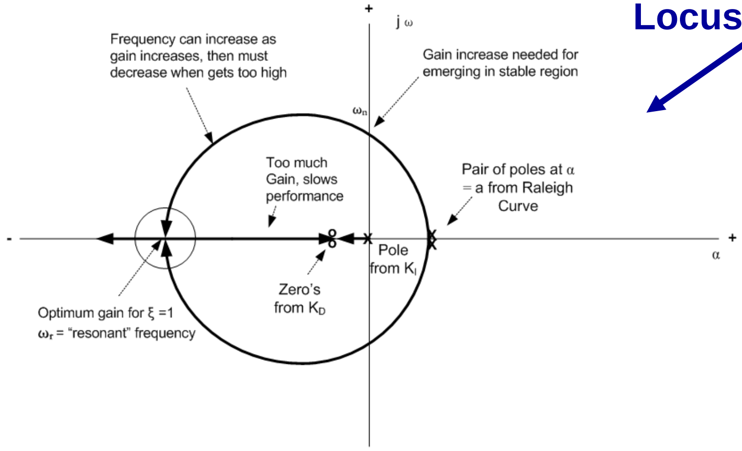
$$p'(t) = 2Kate^{-at^2}$$

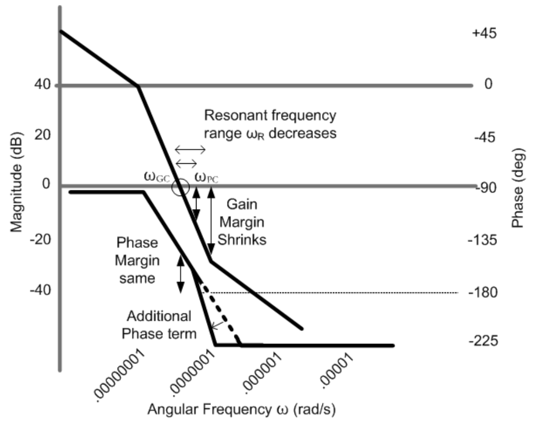
Laplace / Fourier Transforms

$$\mathbf{L}[f(t)] = F(s) = \int_0^\infty f(t)e^{-st}dt$$

Root Locus

Bode


j ω
+
-
+
α
Frequency can increase as gain increases, then must decrease when gets too high
Gain increase needed for emerging in stable region
ωn
Too much Gain, slows performance
Pair of poles at α = a from Raleigh Curve
Pole from K_I
Zero's from K_D
Optimum gain for ξ =1
ω_r = "resonant" frequency


Magnitude (dB)
Phase (deg)
Resonant frequency range ω_R decreases
ω_{GC}
ω_{PC}
Gain Margin Shrinks
Phase Margin same
Additional Phase term
Angular Frequency ω (rad/s)