



Agile and Plan-Driven Methods Oil and Water?

**Barry Boehm, USC
Agile Universe 2002
August 5, 2002**



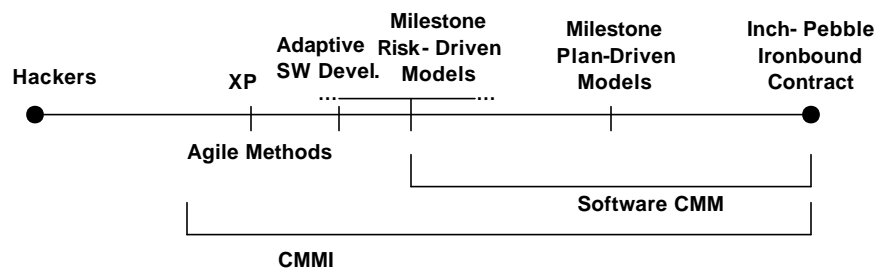
Clarifications

- **Plans include**
 - Process plans (tasks, milestones, org charts)
 - Product plans (architectures, designs)
- **Does not imply that agile methods do not involve planning**
 - But results largely become tacit interpersonal knowledge
 - Rather than explicit documented knowledge
 - And are valued less than responding to change

Outline

- The planning spectrum
- Agile and plan-driven home grounds
- How much planning is enough?
- Using plans to scale up agile methods
 - Thought Works lease management example
- Using agility to streamline plan-driven methods
 - TRW CCPDS-R example
- Reflections on the Agile Manifesto
- Hybrid agile/plan-driven methods

The Planning Spectrum





Agile and Plan-Driven Home Grounds

Agile Home Ground	Plan-Driven Home Ground
<ul style="list-style-type: none">• Agile, knowledgeable, collocated, collaborative developers• Above plus representative, empowered customers• Reliance on tacit interpersonal knowledge• Largely emergent requirements, rapid change• Architected for current requirements• Refactoring inexpensive• Smaller teams, products• Premium on rapid value	<ul style="list-style-type: none">• Plan-oriented developers; mix of skills• Mix of customer capability levels• Reliance on explicit documented knowledge• Requirements knowable early; largely stable• Architected for current and foreseeable requirements• Refactoring expensive• Larger teams, products• Premium on high-assurance



Agile Home Ground and People

- **Agile methods can work with a mix of people**
 - But need at least some agile people
- **49.9999% of software people are below average**
 - Slightly more precisely, below median
- **What’s an “average software person” like?**



“Average Software Person:” 1975 Survey

- 14 large industry and government software installations
- Profile of the average coder
 - Two years’ college level education
 - Two years’ software experience
 - Familiar with two programming languages
 - Familiar with two software products
 - Generally sloppy, inflexible, introverted, “in over his head,” and undermanaged
- Since 1975, demand has strongly exceeded supply of CS graduates
 - No comparable recent survey data found



How Much Planning Is Enough?

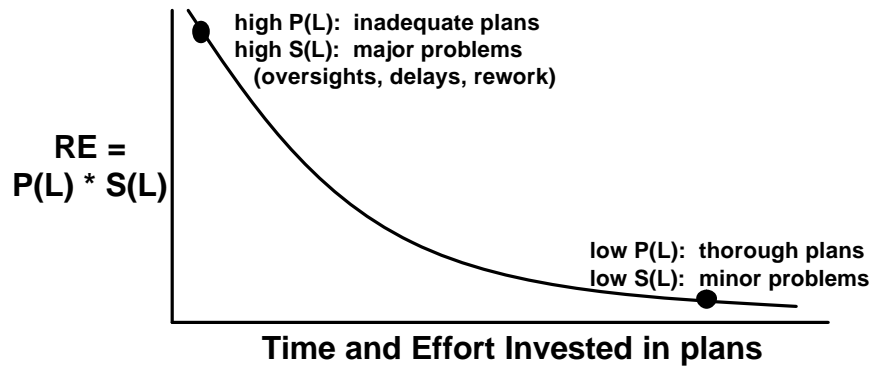
- A risk analysis approach

- Risk Exposure $RE = \text{Prob (Loss)} * \text{Size (Loss)}$
 - “Loss” – financial; reputation; future prospects, ...
- For multiple sources of loss:

$$RE = \sum_{\text{sources}} [\text{Prob (Loss)} * \text{Size (Loss)}]_{\text{source}}$$

Example RE Profile: Planning Detail

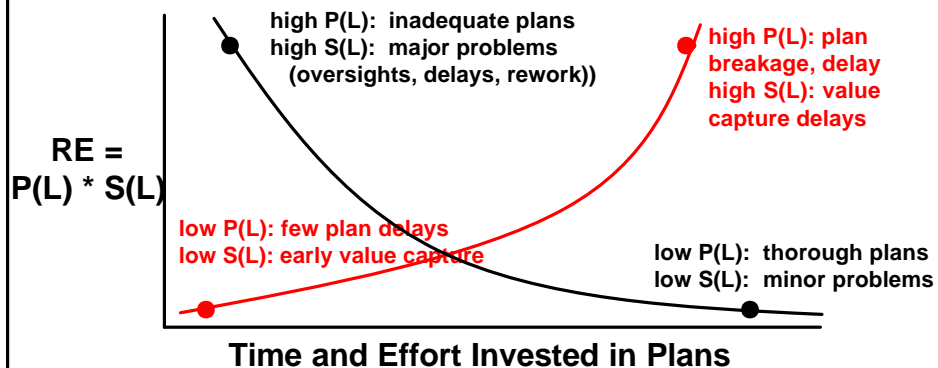
- Loss due to inadequate plans

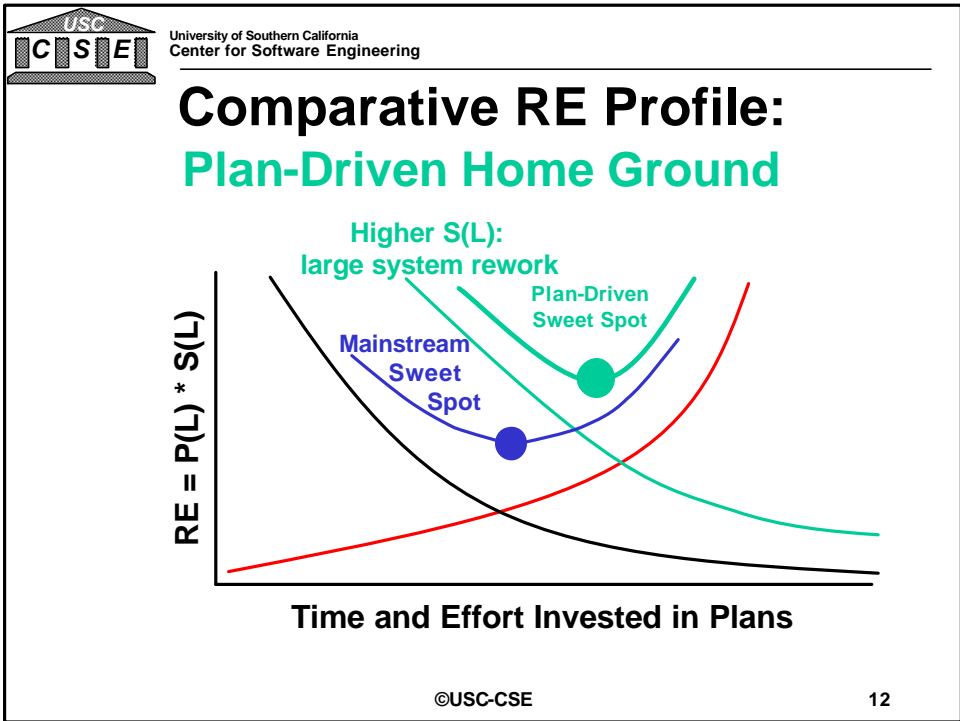
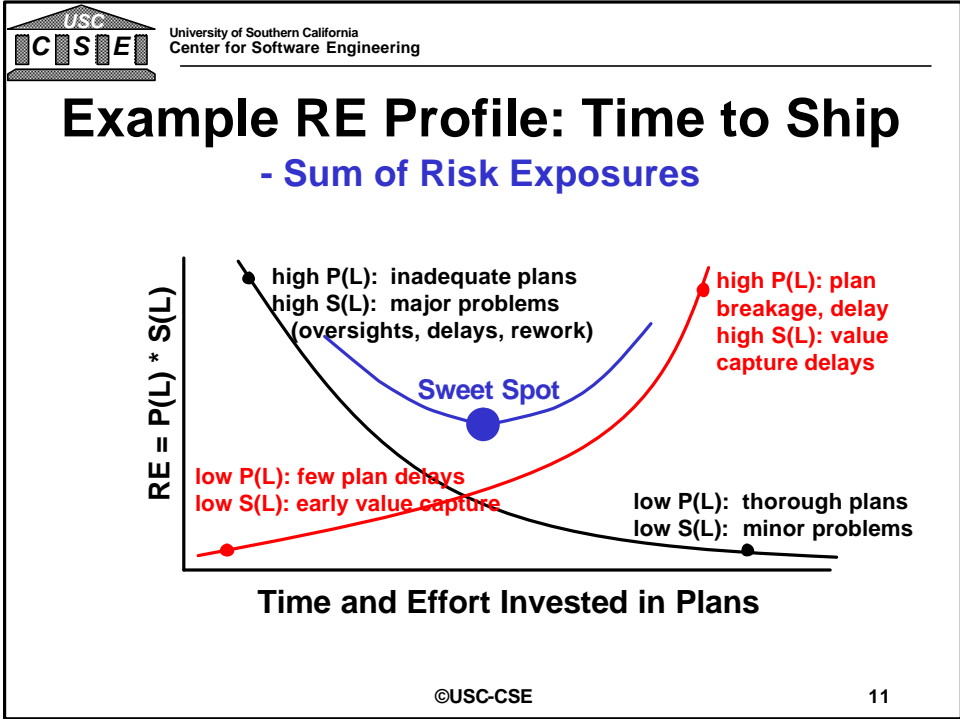


Example RE Profile: Planning Detail

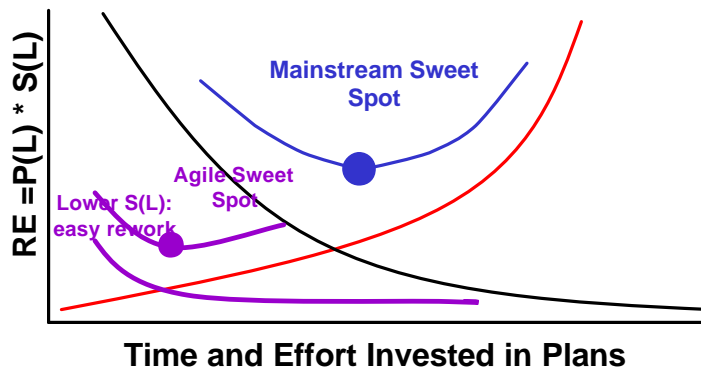
- Loss due to inadequate plans

- Loss due to market share erosion





Comparative RE Profile: Agile Home Ground



Outline

- The planning spectrum
- Agile and plan-driven home grounds
- How much planning is enough?
- • Using plans to scale up agile methods
 - Thought Works lease management example
- Using agility to streamline plan-driven methods
 - TRW CCPDS-R example
- Reflections on the Agile Manifesto
- Hybrid agile/plan-driven methods



Thought Works Lease Management Example

- J2EE application; over 500 KSLOC
 - Over 1000 story cards
- Standard XP approach adopted after failure of 18-month traditional approach
- About 30 developers, 50 people overall
- Encountered and coped with = 8 XP “bad-smells”
 - Parental metaphor that something in code or process is wrong
 - Largely resolved by adapting plan-driven methods

Ref: A. Elssamadisy and G. Schalliol, "Recognizing and Responding to 'Bad Smells' in XP," Proceedings, ICSE 2002, ACM/ IEEE, May 2002, pp. 617-622.
©USC-CSE

15



Bad Smells #1,3 and 7:

The effort to develop or modify a story eventually does increase with time and story number

- **BS #1:** Effort per story increases, but iteration length stayed the same
 - Fix: plan more granular and modular stories
- **BS #3:** Functions satisfied unit tests, but did not integrate well
 - Fix: develop and evolve an easy-to-modify architectural plan
- **BS #7:** Complex functions (unbook/rebook) had high rework growth
 - Fix: more upfront architectural planning

©USC-CSE

16



Bad Smells #2,4 and 6:

Trusting people to get everything done on time does not fit fixed schedules and diseconomies of scale

- **BS #2: Customer complaint rates increase as operational cutover nears**
 - Fix: coach customers to get earlier realistic end-user feedback
- **BS #4: Story cards reported as “finished,” but have weeks of work left**
 - Fix: create a precise list of completion tasks, police it rigorously and honestly
- **BS #6: Large refactorings stink: nobody had time to refactor unbook/rebook**
 - Fix: no simple solution
 - Sometimes works: lengthen later iterations; architect for foreseeable change



Bad Smells #5 and 8:

Simple Design and YAGNI are increasingly risky on large projects

- **BS #5: Continuous expensive refactoring for foreseeable inventory format changes**
 - Fix: Architect to accommodate foreseeable changes (factory pattern)
- **BS #8: Need test drivers reflecting different points in business object’s lifetime**
 - Fix: Violate test-driver version of Simple Design: use factory pattern variant

Outline

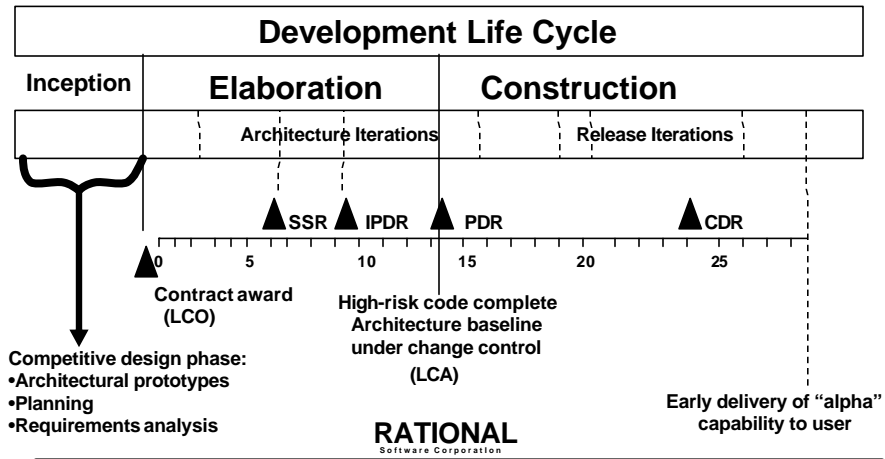
- The planning spectrum
- Agile and plan-driven home grounds
- How much planning is enough?
- Using plans to scale up agile methods
 - Thought Works lease management example
- • Using agility to streamline plan-driven methods
 - TRW CCPDS-R example
- Reflections on the Agile Manifesto
- Hybrid agile/plan-driven methods

Case Study: CCPDS-R Project Overview

Characteristic	CCPDS-R
Domain	Ground based command & control
Size/language	1.15M SLOC Ada
Average number of people	75
Schedule	75 months (48 month IOC), 1987-1994
Process/standards	DOD-STD-2167A Iterative development
Environment	Rational host DEC host DEC VMS targets
Contractor	TRW
Customer	USAF
Outcome	Award-winning, On-budget, On-schedule

Ref: Walker Royce, Software Project Management, Addison Wesley, 1998, pp. 299-362.

CCPDS-R Iterative Spiral Process



Individuals and Interactions over Processes and Tools: CCPDS-R

- Stakeholder win-win negotiation of budget and schedule
 - Using Ada COCOMO cost model
- Processes reinterpreted to avoid wasted effort
 - Working high-risk code by Preliminary Design Review
- Processes, architecture oriented around people's strengths, weaknesses
 - Experts do Ada tasking, novices do sequential code
- Performers empowered with top-of-the-line Ada tools
 - Rational R-1000: incremental syntax, semantics checking
- Award fee flowdown to project performers
 - High motivation, low turnover



Working Software over Comprehensive Documentation: CCPDS-R

- **Early investment in message-passing middleware**
 - Enabled sequential programming of applications software
- **Also enabled test-as-you-go**
 - Via applications software stubs
- **Ada package specs, working software development files as main documentation**
 - With big-picture architecture information
 - Some mil-standards documents generated from these



Customer Collaboration over Contract Negotiation: CCPDS-R

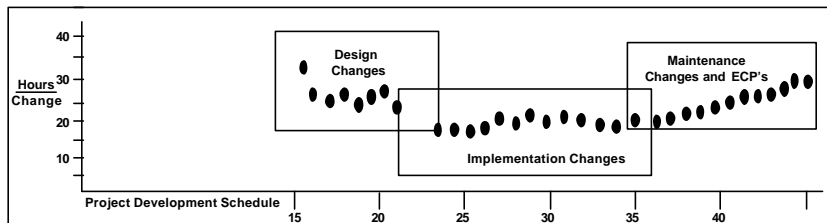
- **Contract negotiation an integral part of customer collaboration**
 - Expectations management via Ada COCOMO
 - Upfront investment in architecture for three foreseeable variants
- **Award fee flowdown a win for customers and developers**
- **Early prototype exercise; collaborative increment planning**
- **Mutual progress visibility via automated metrics**

Responding to Change over Following a Plan: CCPDS-R

- Stakeholders in three user systems involved in planning commonalities, variabilities, priorities
 - Platforms, displays, GUI's, algorithms
- Architecture developed to accommodate reuse and change adaptation
 - Avoided previous off-nominal architecture-breakers
 - Off-nominal performance demonstrated early
- Numerous changes handled efficiently
 - Measured flattening of cost-to-change vs. time curve

Reducing Software Cost-to-Change: CCPDS-R

- Architecture first
 - Integration during the design phase
 - Demonstration-based evaluation
- Risk Management
- Configuration baseline change metrics:





Outline

- The planning spectrum
- Agile and plan-driven home grounds
- How much planning is enough?
- Using plans to scale up agile methods
 - Thought Works lease management example
- Using agility to streamline plan-driven methods
 - TRW CCPDS-R example
- • Reflections on the Agile Manifesto
- Hybrid agile/plan-driven methods



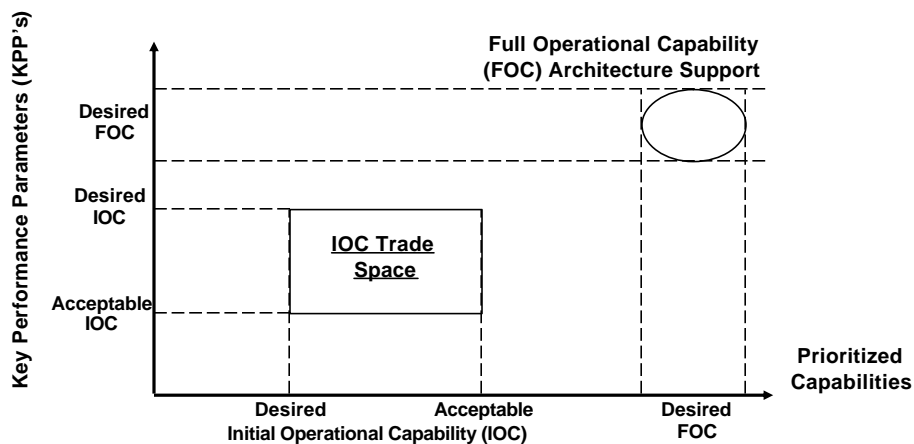
Reflections on the Agile Manifesto

- The four value preferences are not exclusive-or's
 - Processes and tools can empower individuals and interactions
 - Documentation can help people understand complex software
 - Contract negotiation can be an integral part of customer collaboration
 - When changes are foreseeable, following a plan can be the best way to respond to change
- Each pair of alternatives can reinforce each other
 - Can use risk to help find best balance
- People and their value propositions are top priority
 - Not just developers and customers, but end users, maintainers, interoperators, general public, ...

Hybrid Agile/Plan-Driven Methods

- Early agility to determine emergent requirements
 - Agile user-interface prototyping
- Early planning to avoid surprises, point solutions
 - Plans for test data, drivers, oracles, tools, facilities
 - Architectures for assuring scalability, interoperability
- Early hybrid approach for complex COTS selection
 - Planned evaluation criteria, approach, priorities
 - Agile evaluations and iteration of approach
- Main development tailored to agile, plan-driven home grounds
 - Architect to modularize around sources of rapid change
 - Apply agile methods to rapidly changing lower-criticality modules
 - Planned approach to multi-site relativity stable, higher-criticality modules

Large System Life Cycle Planning/Agility Framework



KPP's: cost, schedule, performance, dependability, interoperability, ...



Can't We All Get Along Together?

- **Agile methods bring a lot to the party**
 - Breath of fresh air for many plan-driven people
 - Attracting many erstwhile cowboy hackers
 - Increasingly valuable as pace of change accelerates
- **Plan-driven methods bring a lot to the party too**
 - Particularly as you approach large, critical systems with foreseeable change
- **Collaboration more likely to make progress than “duking it out”**