EECS810
Spring 2012

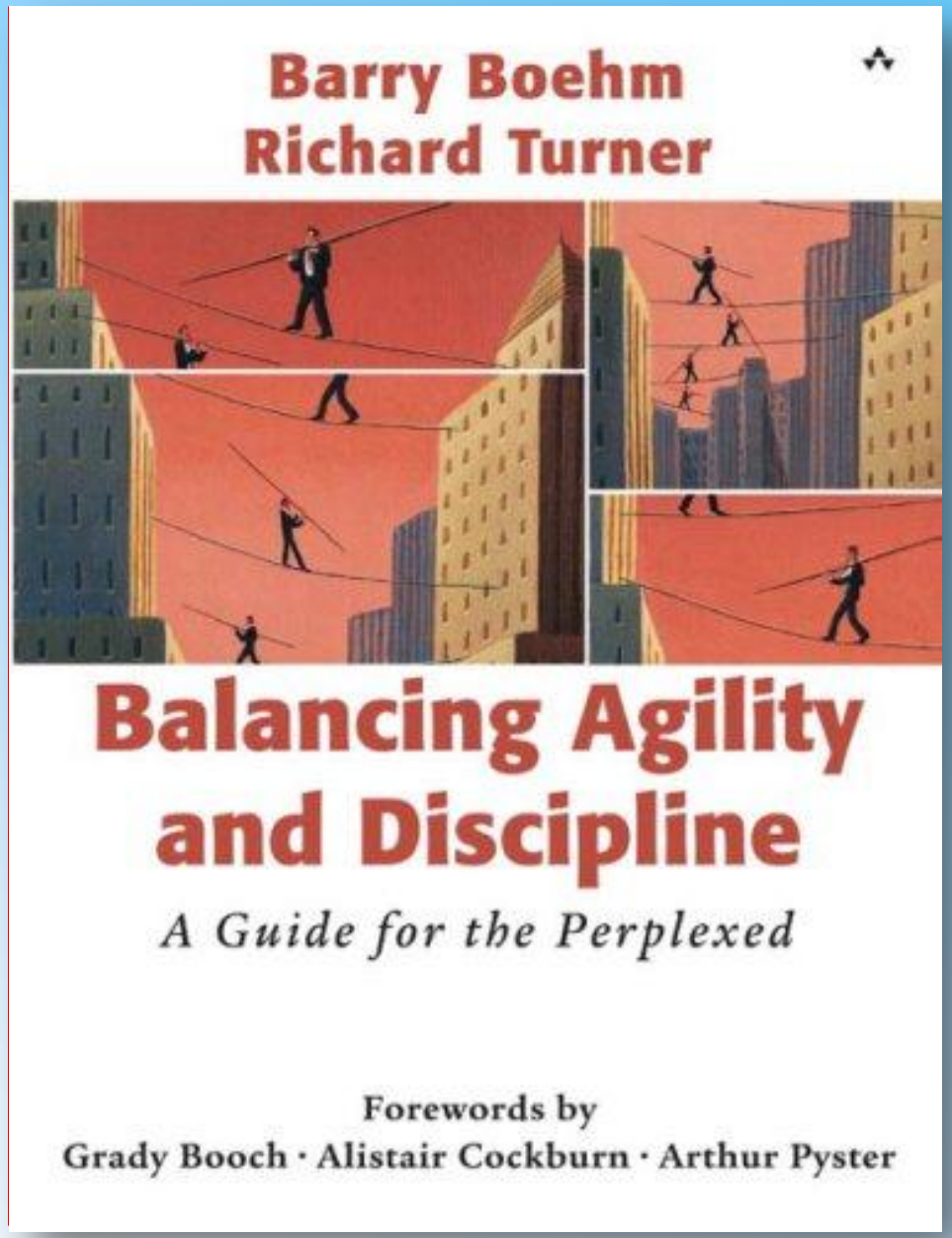
Balancing Agility and Discipline

Barry Boehm

Richard Turner

Presented by:

Bharath Padmanabhan



The Authors

Barry Boehm

TRW Professor of Software Engineering, Computer Science Department

Director, USC Center for Software Engineering

B.A. Harvard

M.S. UCLA

Ph.D. UCLA

Richard Turner

Distinguished Service Professor, School of Systems and Enterprises of Stevens Institute of Technology in Hoboken, New Jersey

Visiting Scientist, Software Engineering Institute of Carnegie Mellon University

BA Mathematics, Huntingdon College

MS Computer Science, University of Louisiana - Lafayette

DSc Engineering Management, The George Washington University

Abstract

- Promise of Agile development methodologies
- Promise of Plan-driven approaches
- Shortcomings in both
- Balancing the two approaches

Contents

What is Discipline?

Noun

discipline (*plural* **disciplines**)

1. A controlled behaviour; self-control
2. An enforced compliance or control
3. A systematic method of obtaining obedience

- Foundation for any successful endeavor
- Provides strength and comfort
- Creates well-organized memories, history, and experience

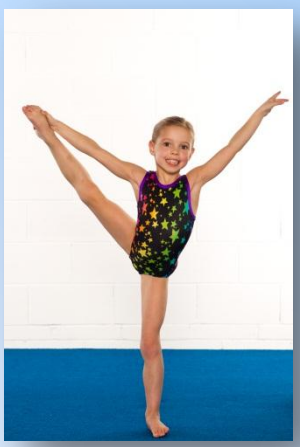
What is Agility?

Noun

agility (*countable and uncountable; plural* **agilities**)

1. (*uncountable*) The quality of being agile; the power of moving the limbs quickly and easily; nimbleness; activity; quickness of motion; as, strength and agility of body.
2. (*countable*) A faculty of being agile in body, mind, or figuratively.

- Counterpart of discipline
- Releases and invents as opposed to ingraining and strengthening
- Applies memory and history to adjust to new environments and, take advantage of new opportunities

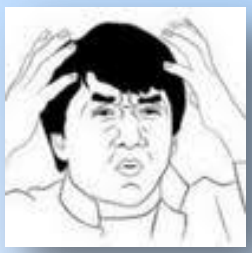


Successful Projects Need Both in a Changing Software Environment

- Discipline without agility leads to bureaucracy and stagnation
- Traditional Plan-driven development
  - "Feels like we're spending more time writing documents than producing software!"
- Agility without discipline leads to uncontrolled and fruitless enthusiasm
- Agile development
  - "Can we realistically scale to do big projects without comprehensive documentation?"

Sources of Perplexity

- Multiple definitions
  - Quality: Customer satisfaction or compliance?
- Distinguishing method use from method misuse
  - Claiming XP use when simply “not documenting”
  - “CMM Level 4 Memorial Library” of 99 2-inch binders
- Overgeneralization based on the most visible instances
  - XP is Agile; CMM is Plan-driven
- Claims of universality
  - Pace of IT change is accelerating and Agile methods adapt to change better than disciplined methods, therefore, Agile methods will prevail

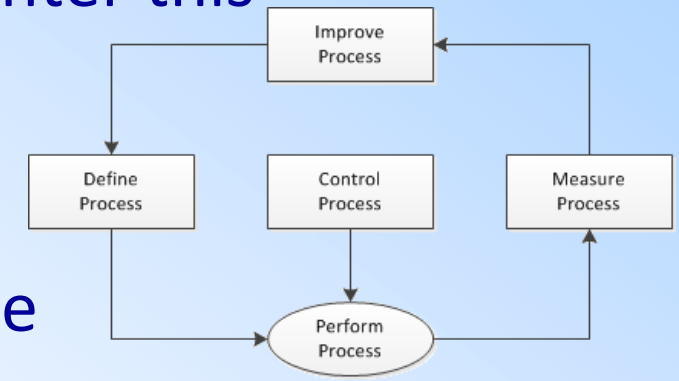


Sources of Perplexity (contd.)

- Early success stories
  - Chrysler project that successfully birthed XP was later cancelled
  - Cleanroom has never made it into the mainstream
- Purist interpretations (and internal disagreements)
  - "Don't start by incrementally adopting parts of XP. Its pieces fit together like a fine Swiss watch"
  - "An advantage of agile methods is that you can apply them selectively and generatively"
  - "If you aren't 100% compliant with SW CMM Level 3, don't bother to bid"
  - "With the CMMI continuous interpretation, you can improve your processes in any order you feel is best"

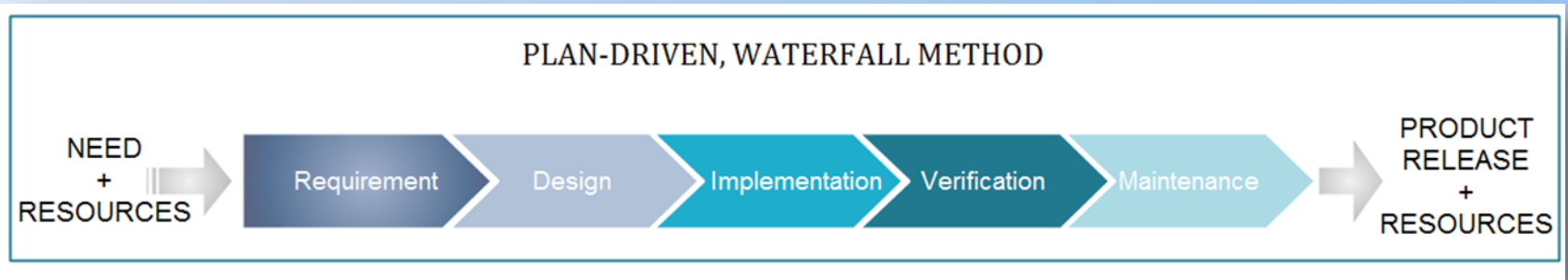
Plan-Driven Methods

- The “traditional” way to develop software
- Genesis lies in the systems engineering and quality disciplines
- While hardware components fit well in this paradigm, software proved difficult to manage
- Standards were introduced to counter this
- Definition and management of process is key as is the support from management & infrastructure



A Process Improvement Cycle

Some Important Plan-Driven Concepts



Colm O'hEocha, AgileInnovation Ltd. 2010: www.agileinnovation.eu

- Process Improvement
- Process Capability
- Organizational Maturity
- Process Group
- Risk Management
- Verification
- Validation
- Software System Architecture

Important Plan-Driven Concepts (contd.)

- **Process Improvement**
  - A program of activities designed to improve the performance and maturity of the organization's processes, and the results of such a program.
- **Process Capability**
  - The inherent ability of a process to produce planned results.
- **Organizational Maturity**
  - By steadily improving its process capability, an organization is said to mature.

Important Plan-Driven Concepts (contd.)

- **Process Group**
  - A collection of specialists that facilitate the definition, maintenance, and improvement of the processes used by an organization.
- **Risk Management**
  - An organized, analytic process to identify uncertainties that might cause harm or loss, assess and quantify the identified risks, and develop and apply risk management plans to prevent or handle risk causes that could result in significant harm or loss.

Important Plan-Driven Concepts (contd.)

- **Verification**
  - Confirms that work products (specs, designs, models) properly reflect the requirements specified for them (building the product right).
- **Validation**
  - Confirms the fitness or worth of a work product for its operational mission (building the right product).
- **Software System Architecture**
  - Collection of software and system components, connectors, and constraints; stakeholders' need statements; and a rationale which demonstrates that the components, connectors, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.

Examples of Plan-Driven Approaches

- **Military Methods (DoD)**
  - DoD-STD-2167 was a document-driven approach that specified a large number of Data Item Descriptions for deliverables. Tailoring was encouraged, but was rarely effectively done.
  - MIL-STD-1521 details a set of sequential reviews and audits required.
  - MIL-STD-498 revised 2167 to allow more flexibility in systems engineering, planning, development, and integration.
  - MIL-STD-499B defines the contents of a systems engineering management plan.

Examples of Plan-Driven Approaches (contd.)

- **General Process Standards (ISO, EIA, IEEE)**
  - EIA/IEEE J-STD-016 was a generalization of MIL-STD-498 to include commercial software processes.
  - ISO 9000 is a quality management standard that includes software.
  - ISO 12207 and 15504 address the software life cycle and ways to appraise software processes.

- **Software Factories (Hitachi, GE, others)**
  - A long-term, integrated effort to improve software quality, software reuse, and software development productivity.
  - Highly process-driven, emphasizing early defect reduction.

Examples of Plan-Driven Approaches (contd.)

- **Cleanroom (Harlan Mills, IBM)**
  - Uses statistical process control and mathematically based verification to develop software with certified reliability.
  - The name comes from physical clean rooms that prevent defects in precision electronics.

- **Capability Maturity Model Software (SEI, Air Force, others)**
  - A process improvement framework, SW-CMM grew out of the need for the Air Force to select qualified software system developers.
  - Collects best practices into Key Practice Areas that are organized into five levels of increasing process maturity.

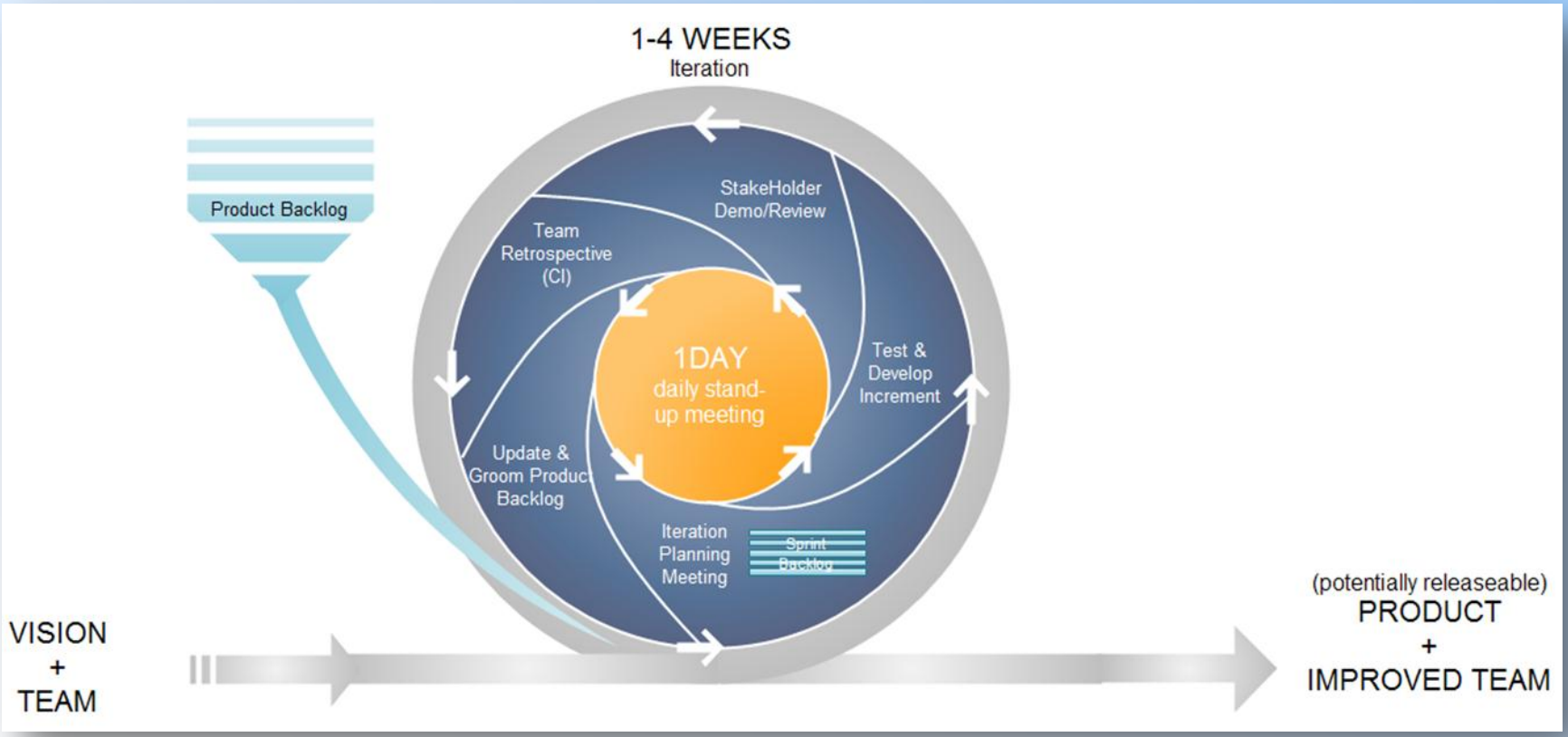
Examples of Plan-Driven Approaches (contd.)

- **CMM Integration (SEI, DoD, NDIA, others)**
  - CMMI was established to integrate software and systems engineering CMMs, and improve or extend the CMM concept to other disciplines.
  - Its a suite of models and appraisal methods that address a variety of disciplines using a common architecture, vocabulary, and a core of process areas.
- **Personal Software Process (PSP)/Team Software Process (TSP) (Watts Humphrey, SEI)**
  - PSP is a structured framework of forms, guidelines, and procedures for developing software. Directed toward the use of self-measurement to improve individual programming skills.
  - TSP builds on PSP and supports the development of industrial-strength software through the use of team planning and control.

Agile Methods

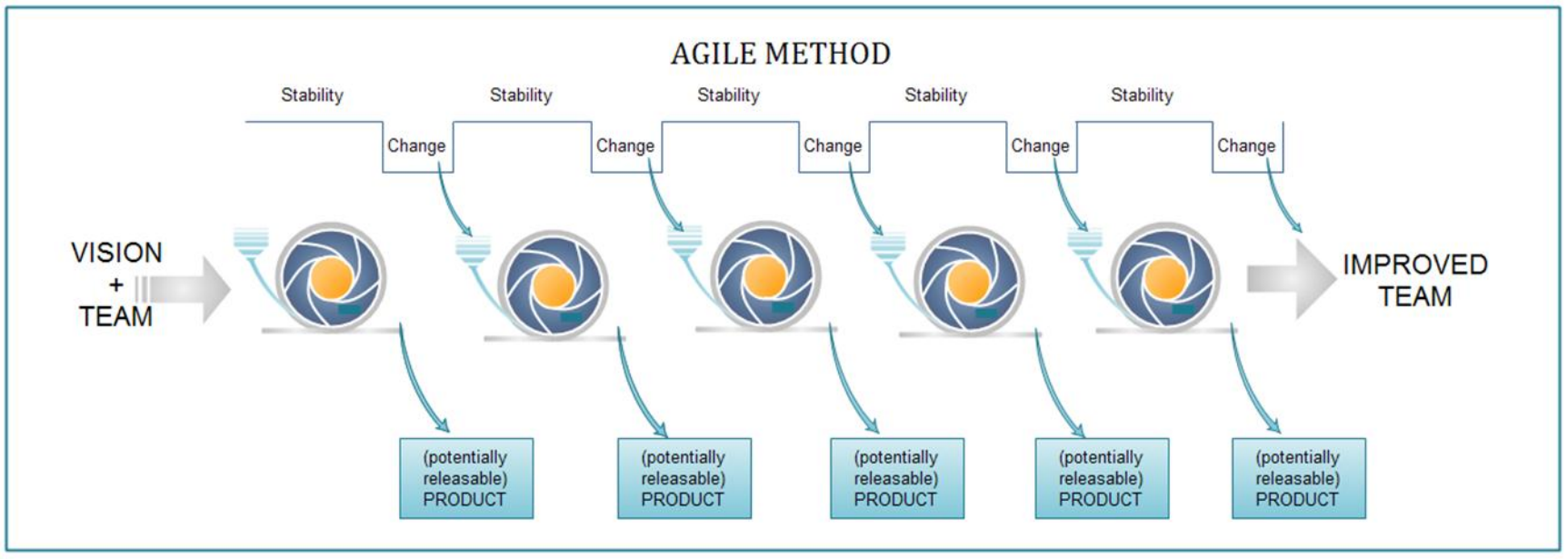
- Grew from rapid prototyping and the philosophy that programming is more a craft than mechanical process
- Rapid change and long development cycles don't mix
- Chaordic = Chaos + Order
- Agile Manifesto
  - **Individuals and interactions** over processes and tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan

Life of an Iteration



Colm O'hEocha, AgileInnovation Ltd. 2010: www.agileinnovation.eu

Some Important Agile Concepts



Colm O'hEocha, AgileInnovation Ltd. 2010: www.agileinnovation.eu

- Embracing Change
- Fast Cycle / Frequent Delivery
- Simple Design
- Refactoring
- Pair Programming
- Retrospective
- Tacit Knowledge
- Test-Driven Development

Important Agile Concepts (contd.)

- **Embracing Change**
  - Seeing change as an ally rather than an enemy. Change allows for more creativity and quicker value to the customer.
- **Fast Cycle/Frequent Delivery**
  - Scheduling many releases with short time spans between then; forces implementation of only the highest priority functions, delivers value to the customer quickly, and speeds requirements emergence.
- **Simple Design**
  - Designing for the battle, not the war. The motto is YAGNI ("You Aren't Going to Need It"). Strips designs down the cover just what is currently being developed. Since change is inevitable, planning for future functions is a waste of effort.

Important Agile Concepts (contd.)

- **Refactoring**
  - The restructuring of software to remove duplication, improve communication, simplify, or add flexibility without changing its behavior. Just-in-time redesign.
- **Pair Programming**
  - A style of programming in which two programmers work side by side at one computer, continually collaborating on the same design, algorithm, code, or test.
- **Retrospective**
  - A post-iteration review of the effectiveness of the work performed, methods used, and estimates. The review supports team learning and estimation for future iterations. Sometimes called “reflection”.

Important Agile Concepts (contd.)

- **Tacit Knowledge**
  - Agility is achieved by establishing and updating project knowledge in the participants' heads rather than in documents (explicit knowledge).

- **Test-Driven Development**
  - Module or method tests are incrementally written by the developers and customers before and during coding. Supports and encourages very short iteration cycles.

Examples of Agile Approaches

- **eXtreme Programming (XP) (Kent Beck, Ward Cunningham, etc)**
  - Regarded as perhaps the most famous agile method.
  - Refined from experience gained developing an information system for Daimler Chrysler corporation.
  - Fairly rigorous and initially expects all practices to be followed.
  - Practices include stories, pair programming, simple design, test first, and continuous integration.

- **Adaptive Software Development (ASD) (Jim Highsmith)**
  - Provides a philosophical base and practical approach.
  - Uses iterative development, feature-based planning, and customer focus group reviews within a leadership-collaboration management style.

Examples of Agile Approaches (contd.)

- **Crystal (Alistair Cockburn)**
  - Provides different levels of "ceremony" depending on the size of the team and the criticality of the project.
  - Practices draw from agile and plan-driven methods as well as psychology and organizational development research.

- **Scrum (Ken Schwaber, Jeff Sutherland, Mike Beedle)**
  - More of a management technique.
  - Projects are divided into 30-day work intervals ("sprints") in which a specific number of requirements from a prioritized list ("backlog") are implemented.
  - Daily 15-minute "Scrum meetings" maintain coordination.

Examples of Agile Approaches (contd.)

- **Feature-Driven Development (FDD) (Jeff DeLuca, Peter Coad)**
  - Lightweight architecturally based process that initially establishes and overall object architecture and features list.
  - Proceeds to design-by-feature and build-by-feature afterwards.
  - Maintains Chief Architect and Chief Programmer roles.
  - Use of UML or other object-oriented design methods is strongly implied.

Finding the Middle Ground

- Both approaches have "home grounds"
  - **Plan-driven:** Generally large, complex systems, often with safety-critical or other high-reliability attributes
  - **Agile:** Smaller systems and development teams, readily available customers and user, and changing requirements and environment
- **Risk is the key**; processes should be the right weight for the specific project, team, and environment
- *"Is it riskier for me to apply (more of) this process component or to refrain from applying it?"*

Contents

Software Project Characteristics

- Application Characteristics
  - Primary Goals, Size, Environment
- Management Characteristics
  - Customer Relations, Planning and Control, Project Communication
- Technical Characteristics
  - Requirements, Development, Testing
- Personnel Characteristics
  - Customers, Developers, Culture

Application Characteristics

Characteristics	Agile	Plan-Driven
Primary Goals	Rapid value; responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low-change; project/organization focused

Management Characteristics

Characteristics	Agile	Plan-Driven
Customer Relations	Dedicated on-site customers; focused on prioritized increments	As-needed customer interactions; focused on contract provisions
Planning and Control	Internalized plans; qualitative control	Documented plans, quantitative control
Communication	Tacit interpersonal knowledge	Explicit documented knowledge

Technical Characteristics

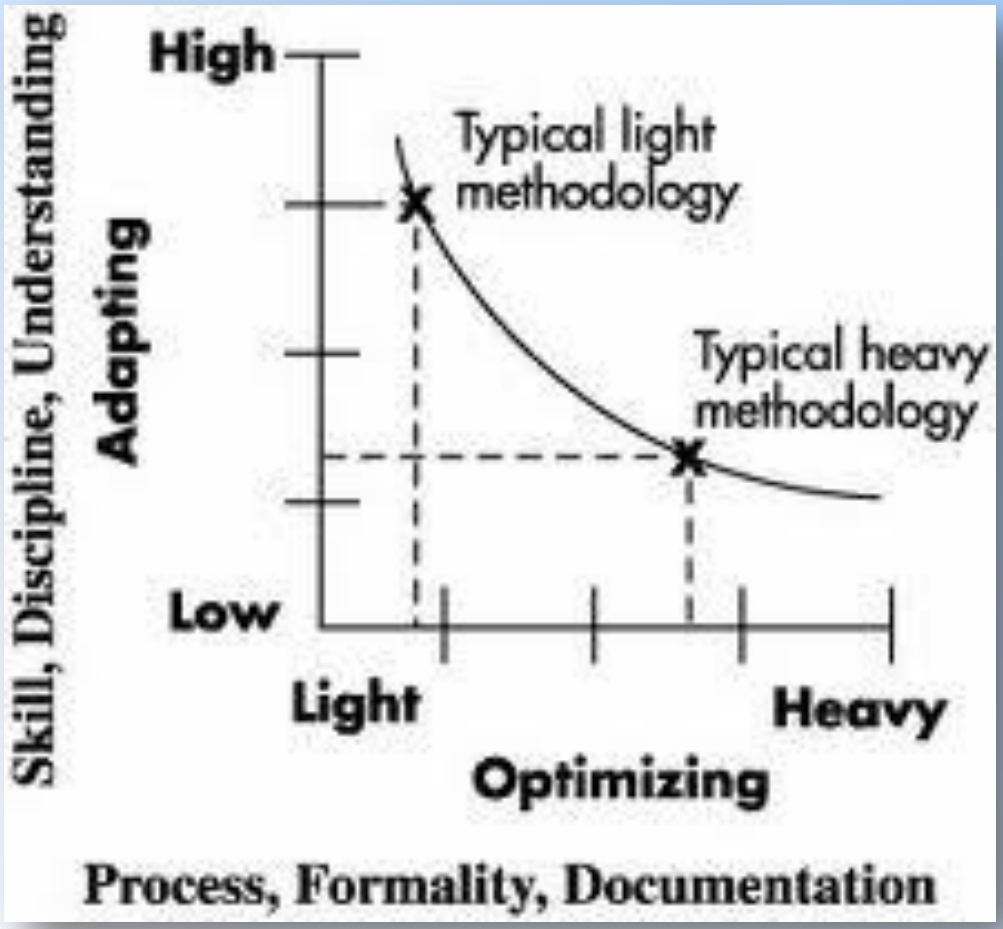
Characteristics	Agile	Plan-Driven
Requirements	Prioritized informal stories and test cases; undergoing unforeseeable change	Formalized project, capability, interface, quality, foreseeable evolution requirements
Development	Simple design; short increments; refactoring assumed inexpensive	Extensive design; longer increments; refactoring assumed expensive
Testing	Executable test cases define requirements	Documented test plans and procedures

Personnel Characteristics

Characteristics	Agile	Plan-Driven
Customers	Dedicated, collocated CRACK* performers	CRACK* performers, not always collocated
Developers	At least 30% full-time Cockburn Level 2 and 3 experts; no Level 1B or -1 personnel**	50% Cockburn Level 3s early; 10% throughout; 30% Level 1Bs workable; no Level -1s**
Culture	Comfort and empowerment via many degrees of freedom (thriving on chaos)	Comfort and empowerment via framework of policies and procedures (thriving on order)

* Collaborative, Representative, Authorized, Committed, Knowledgeable
** These numbers will permanently vary with the complexity of the application

Balancing Optimizing and Adapting Dimensions (Cockburn and Highsmith)



Levels of Software Method Understanding and Use (after Cockburn)

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation.
2	Able to tailor a method to fit a precedential new situation.
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience, can become Level 2.
1B	With training, able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience, can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Misconceptions and Realities About Plan-Driven Methods

Misconceptions	Realities
Plan-driven methods are uniformly bureaucratic	Overly bureaucratic cultures and methods can stultify software development
Having documented plans guarantees compliance with plans	Not necessarily
Plan-driven methods can succeed with a lack of talented people	Plan-driven methods can succeed with a smaller percentage of talented people
High maturity guarantees success	Explicit, documented plans provide more of a safety net than tacit plans
There are no penalties in applying plan-driven methods when change is unforeseeable	Plan-driven methods work best in accommodating foreseeable change

Misconceptions and Realities About Agile Methods

Misconceptions	Realities
Agile methods don't plan	Agile methods get much of their speed and agility through creating and exploiting tacit knowledge
Agile methods require uniformly talented people	Agile methods work best when there is a critical mass of highly talented people involved
Agile methods can make the slope of the cost-to-change vs. time curve uniformly flat	Agile methods can reduce the slope of the cost-to-change vs. time curve
YAGNI is a universally safe assumption, and won't alienate your customers	YAGNI helps handle unforeseeable change, but is risky when change is foreseeable

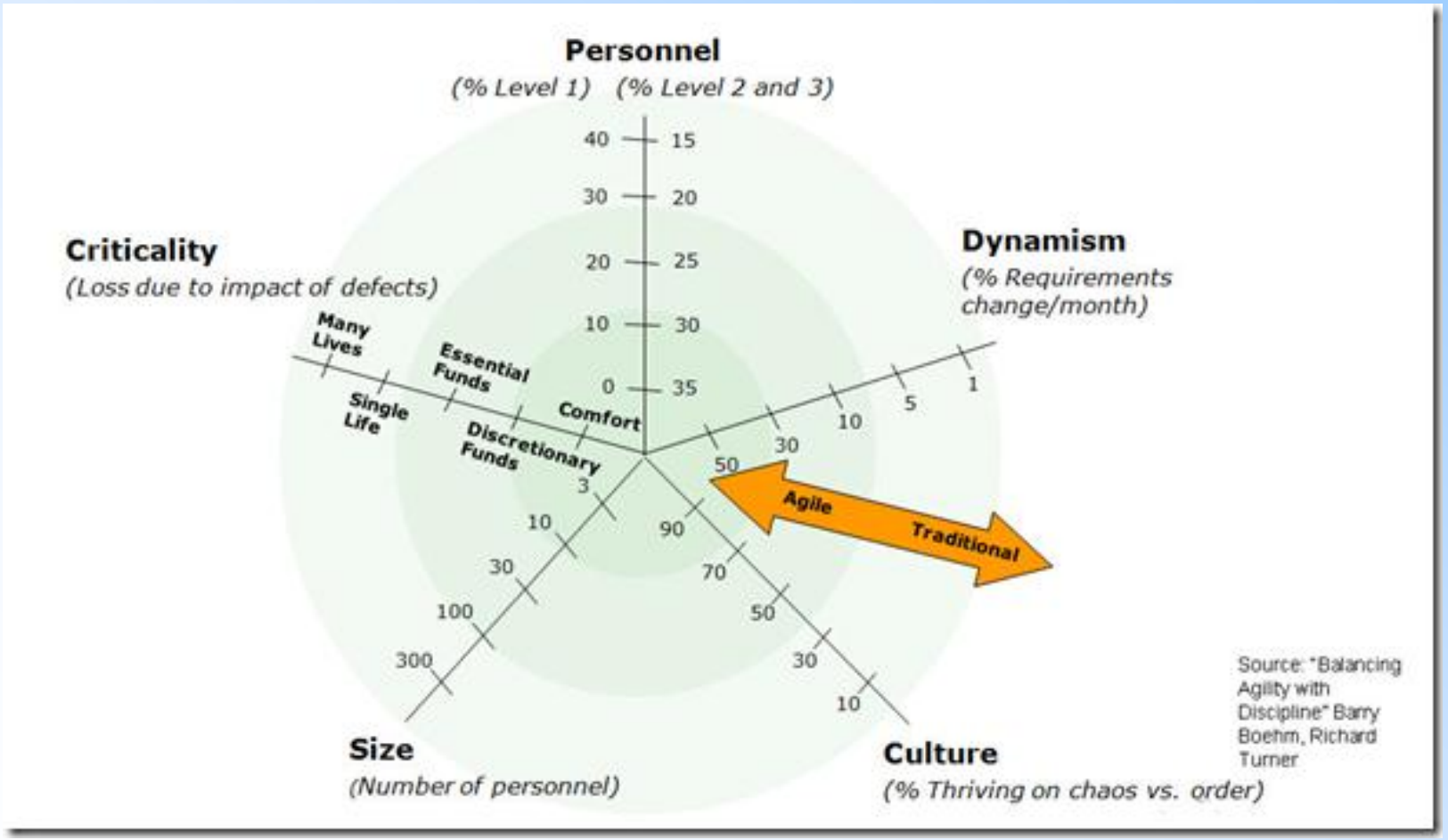
The Five Critical Agility/Plan-Driven Factors

Factor	Agility Discriminators	Plan-Driven Discriminators
Size	Well-matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to low-criticality products.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments, but a source of potentially expensive rework for highly stable environments.	Detailed plans and Big Design Up Front excellent for highly stable environment, but a source of expensive rework for highly dynamic environments.

The Five Critical Agility/Plan-Driven Factors (contd.)

Factor	Agility Discriminators	Plan-Driven Discriminators
Personnel	Requires continuous presence of a critical mass of scarce Cockburn Level 2 or 3 experts. Risky to use non-agile Level 1B people.	Needs a critical mass of scarce Cockburn Level 2 and 3 experts during project definition, but can work with fewer later in the project – unless the environment is highly dynamic. Can usually accommodate some Level 1B people.
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom. (Thriving on chaos)	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures. (Thriving on order)

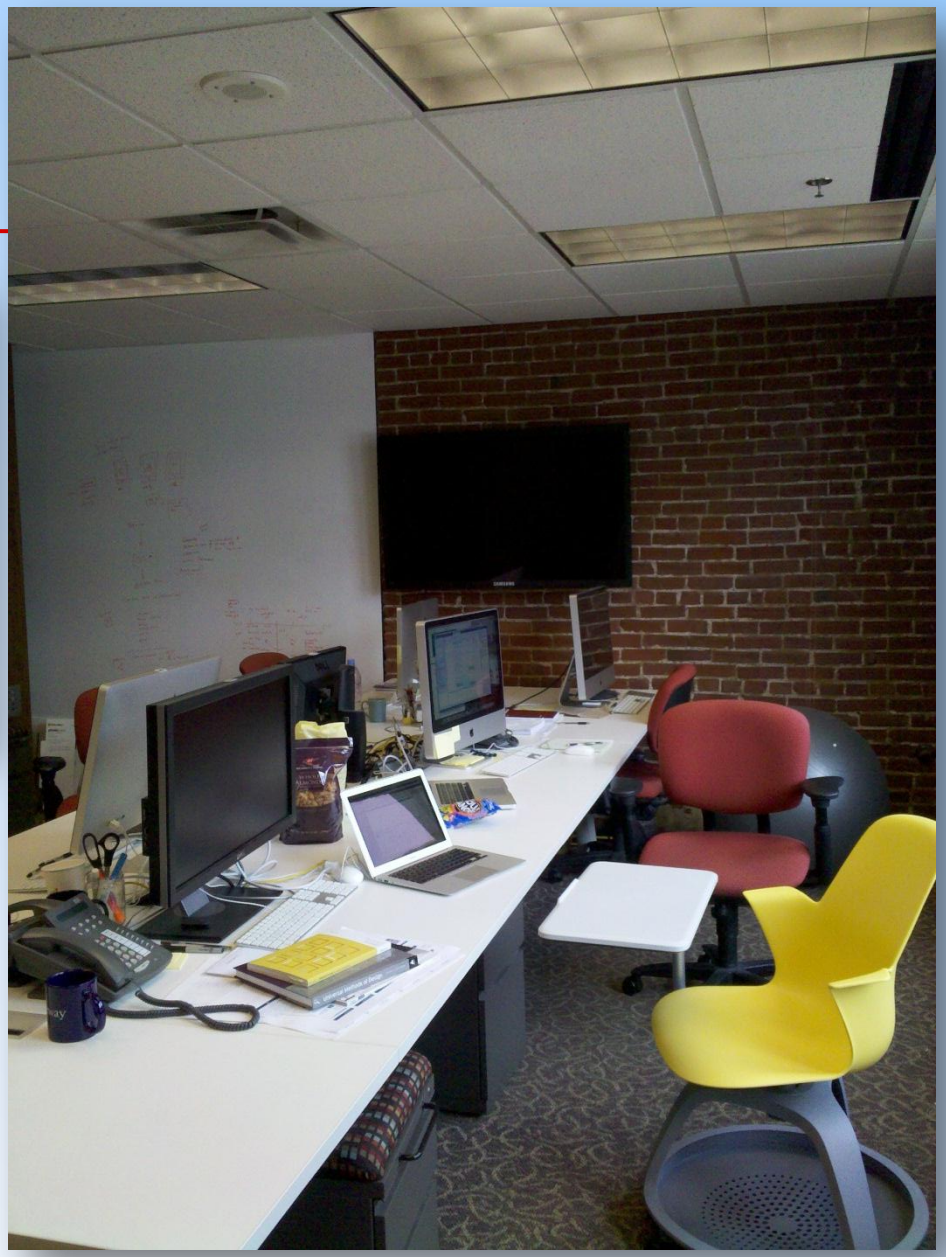
Dimensions Affecting Method Selection



Combining Agile and Plan-Driven Methods

Misconceptions	Reality
Agile and plan-driven methods are completely unmixable.	Agile and plan-driven methods have been successfully combined in a variety of situations.
There are one-size-fits-all process templates for balancing agile and plan-driven methods.	Variations in project risks and stakeholder value propositions lead to different balances of agile and plan-driven methods.
Balancing agile and plan-driven methods is a one-dimensional pure-technology, pure-management, or pure-personnel activity.	Balancing agile and plan-driven methods involves multidimensional consideration of technology, management, and personnel factors.

Preference?

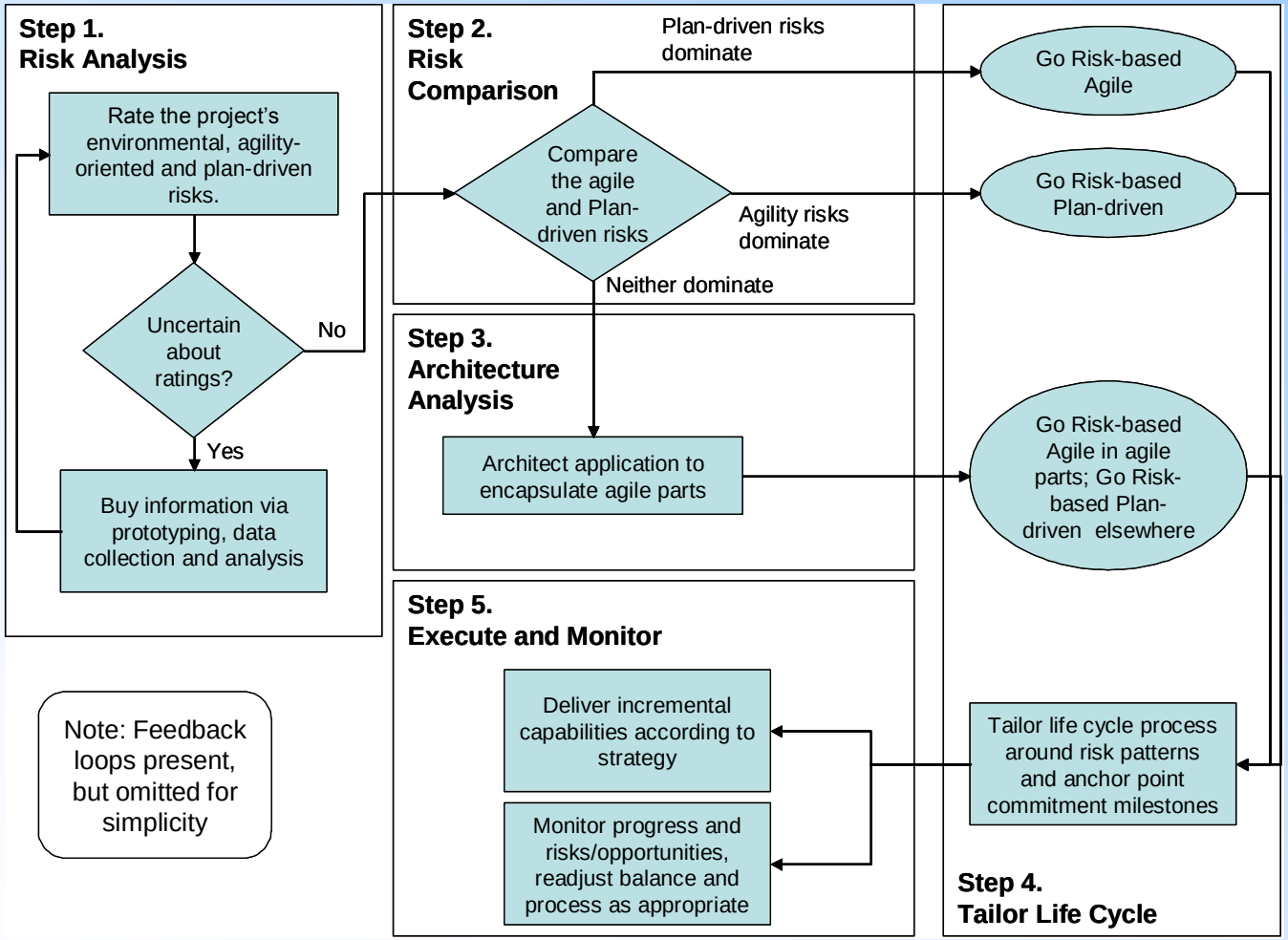


Contents

A Tailorable Method for Balancing Agile and Plan-Driven Methods

Step 1	Rate the project's environmental, agile, and plan-driven risks. If uncertain about ratings, buy information via prototyping, data collection, and analysis.
Step 2a	If agility risks dominate plan-driven risks, go risk-based plan-driven.
Step 2b	If plan-driven risks dominate agility risks, go risk-based agile.
Step 3	If parts of the application satisfy 2a and others 2b, architect the application to encapsulate the agile parts. Go risk-based agile in the agile parts and risk-based plan-driven elsewhere.
Step 4	Establish an overall project strategy by integrating individual risk mitigation plans.
Step 5	Monitor progress and risks/opportunities, re-adjust balance and process as appropriate.

Summary of Risk-Based Method



Categories of Risk

- Environmental
  - Risks that result from the project's general environment
- Agile
  - Risks that are specific to the use of agile methods
- Plan-Driven
  - Risks that are specific to the use of plan-driven methods

Candidate Risks for Environment

Environmental risk: risks that result from the project's general environment

- **E-Tech.** Technology uncertainties
- **E-Coord.** Many diverse stakeholders to coordinate
- **E-Cmplx.** Complex system of systems

Candidate Risks for Agile

Agile risks: risks that are specific to the use of the agile methods

- **A-Scale.** Scalability and criticality
- **A-YAGNI.** Use of simple design or YAGNI
- **A-Churn.** Personnel turnover or churn
- **A-Skill.** Not enough people skilled in agile methods

Candidate Risks for Plan-Driven

Plan-driven risks: risks that are specific to the use of plan-driven methods

- **P-Change.** Rapid change
- **P-Speed.** Need for rapid results
- **P-Emerge.** Emergent requirements
- **P-Skill.** Not enough people skilled in plan-driven methods

Example Family of Applications: Agent-Based Planning Systems

- Software agents search for and locate information across a network
  - Provide significant improvements
  - Have significant risks
  - Good examples
- Example applications
  - Small, relatively noncritical
  - Intermediate
  - Very large, highly critical

Characteristics of SupplyChain.com (Intermediate Application)

An agent-based planning system for supply chain management across a network of producers and consumers (based on the risk patterns discussed in ThoughtWorks' experience scaling up XP to a 50-person project in a lease management application).

Team size	50
Team type	Distributed; often multiorganization
Failure risks	Major business losses
Clients	Multiple success-critical stakeholders
Requirements	Some parts relatively stable, others volatile, emergent
Architecture	Mostly provided by small number of COTS packages
Refactoring	More expensive with mix of people skills
Primary objective	Rapid value increase, dependability, adaptability

SupplyChain.com (Intermediate Application)

Step 1: Rate the project risks

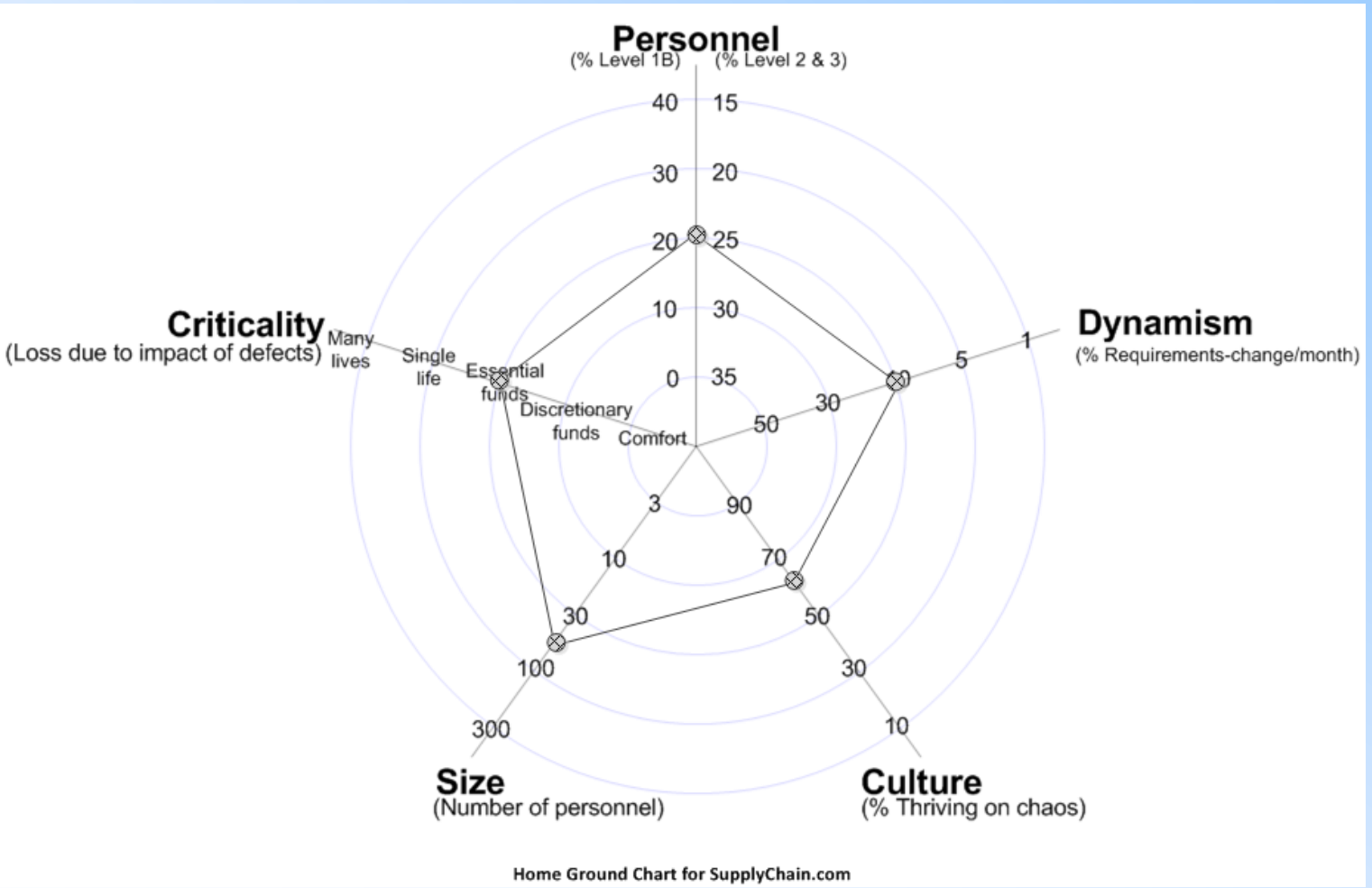
- Environmental
  - E-Tech. Uncertainties with agent-based system
  - E-Coord. Multiple suppliers and distributors
- Agile Risks
  - A-Scale. 50 person development team
  - A-YAGNI. Keeping it simple vs. over-architecting
  - A-Churn. Relatively stable workforce
- Plan-driven Risks
  - P-Change. Slow response to changes in technology, organizations, and market conditions
  - P-Speed. Keeping pace with competition; loss of market share
  - P-Emerge. Effect of changing requirements on architecture

Risk Ratings for SupplyChain.com (Intermediate Application)

E-Tech. Technology uncertainties	2
E-Coord. Many stakeholders	1
E-Cmplx. Complex system of systems	1
A-Scale. Scalability and criticality	2
A-YAGNI. Use of simple design	2
A-Churn. Personnel turnover	1
A-Skill. Not enough people skilled in agile methods	1
P-Change. Rapid change	2
P-Speed. Need for rapid results	2
P-Emerge. Emergent requirements	2
P-Skill. Not enough people skilled in plan-driven methods	1

Risk Scale:
0–Minimal; 1–Moderate; 2–Serious but manageable; 3–Very serious but manageable; 4–Showstopper

Home Ground Chart for SupplyChain.com (Intermediate Application)



SupplyChain.com (Intermediate Application)

Step 2: Compare the Agile and Plan-driven risks

- Agile and plan-driven risks can be seen and mitigated
  - Agile risks (scalability, criticality, simple design) can be addressed by selective application of plan-driven planning and architecting techniques
  - Plan-driven risks (rapid change, rapid response, emerging requirements) can be addressed by selective application of agile methods within an overall plan-driven framework
- **Risk-based agile approach is chosen** based on project culture and environment
- No need for Step 3

SupplyChain.com (Intermediate Application)

Step 4a: Individual risk resolution strategies

- E-Tech: Technical uncertainties
  - Use a combination of reference checking, analysis, benchmarking, and prototyping
  - Technology-watch activity is needed to assess emerging technology risks and opportunities
- E-Coord, E-Cmplx: Many separately evolving networks of Suppliers and Distributors
  - Involve the stakeholders
  - Ensure CRACK stakeholder representatives
  - Negotiate and document interfaces

SupplyChain.com (Intermediate Application) (contd.)

Step 4a: Individual risk resolution strategies

- A-Scale: Diseconomies of scale
  - Variable length iterations; timeboxing to stabilize intervals
- A-YAGNI: Foreseeable change vs. Simple design and YAGNI
  - Architecture and design patterns (encapsulation)
- A-Churn: Loss of tacit knowledge via personnel turnover
  - Alternate representatives and completion bonuses
- Plan-driven risks (P-Change, P-Speed, P-Emerge)
  - Mitigated by using more agile techniques

Risk Mitigation Strategies for SupplyChain.com (Intermediate Application)

Environmental risks

Risk Items	Mitigation Approaches
E-Tech. Technology uncertainties	Risk-driven technology prototypes; Technology and COTS watch
E-Coord. Many stakeholders	Results chain; Stakeholder win-win; CRACK representatives
E-Cmplx. Complex system of systems	Architecture determination; Early Commitments on validated interfaces

Risk Mitigation Strategies for SupplyChain.com (Intermediate Application) (contd.)

Risks of using Agile methods

Risk Items	Mitigation Approaches
A-Scale. Scalability and criticality	Subsettable stories; Longer iterations as size/complexity grows
A-YAGNI. Use of simple design	Balance with high-level change-prescient architecture; Design patterns
A-Churn. Personnel turnover	Personnel backups; Pair programming; Project completion bonuses
A-Skill. Not enough people skilled in agile methods	Low-risk

Risk Mitigation Strategies for SupplyChain.com (Intermediate Application) (contd.)

Risks of using Plan-driven methods

Risk Items	Mitigation Approaches
P-Change. Rapid change	Short iterations; Balanced simple design and change-prescient architecture
P-Speed. Need for rapid results	Short iterations; Pair programming; Timeboxing
P-Emerge. Emergent requirements	Short iterations; Dedicated customer
P-Skill. Not enough people skilled in plan-driven methods	Risk management team with agile and plan-driven method skills

SupplyChain.com (Intermediate Application)

Step 4b: System development

- Three major participant groups are critical to success
  - Operational stakeholders
    - CRACK representatives from Manufacturing, Supplier, and Distributor
  - Risk/Opportunity management team
    - Project manager and three senior (Level 2 or 3) staff
    - Keep watch and take appropriate response
  - Agile feature teams
    - Team leaders (Level 2) and team members (Level 2 and Level 1A)

SupplyChain.com (Intermediate Application) (contd.)

Step 4b: System development

- Three-phase development strategy
  1. Inception
     - Rapid team-building
     - Development of shared system vision
  2. Elaboration
     - Establish overall operational concept and life cycle strategy
     - Establish a set of key COTS, reuse, and architecture decisions
     - Prioritized features for first increment
  3. Implement system incrementally
     - Feature teams develop successive increments
     - Development, testing, lessons-learned

Characteristics of Event Managers (Small, Non-Critical Application)

An agent-based planning system for managing events such as conferences or conventions (based on the risk patterns observed in small, campus Web services applications).

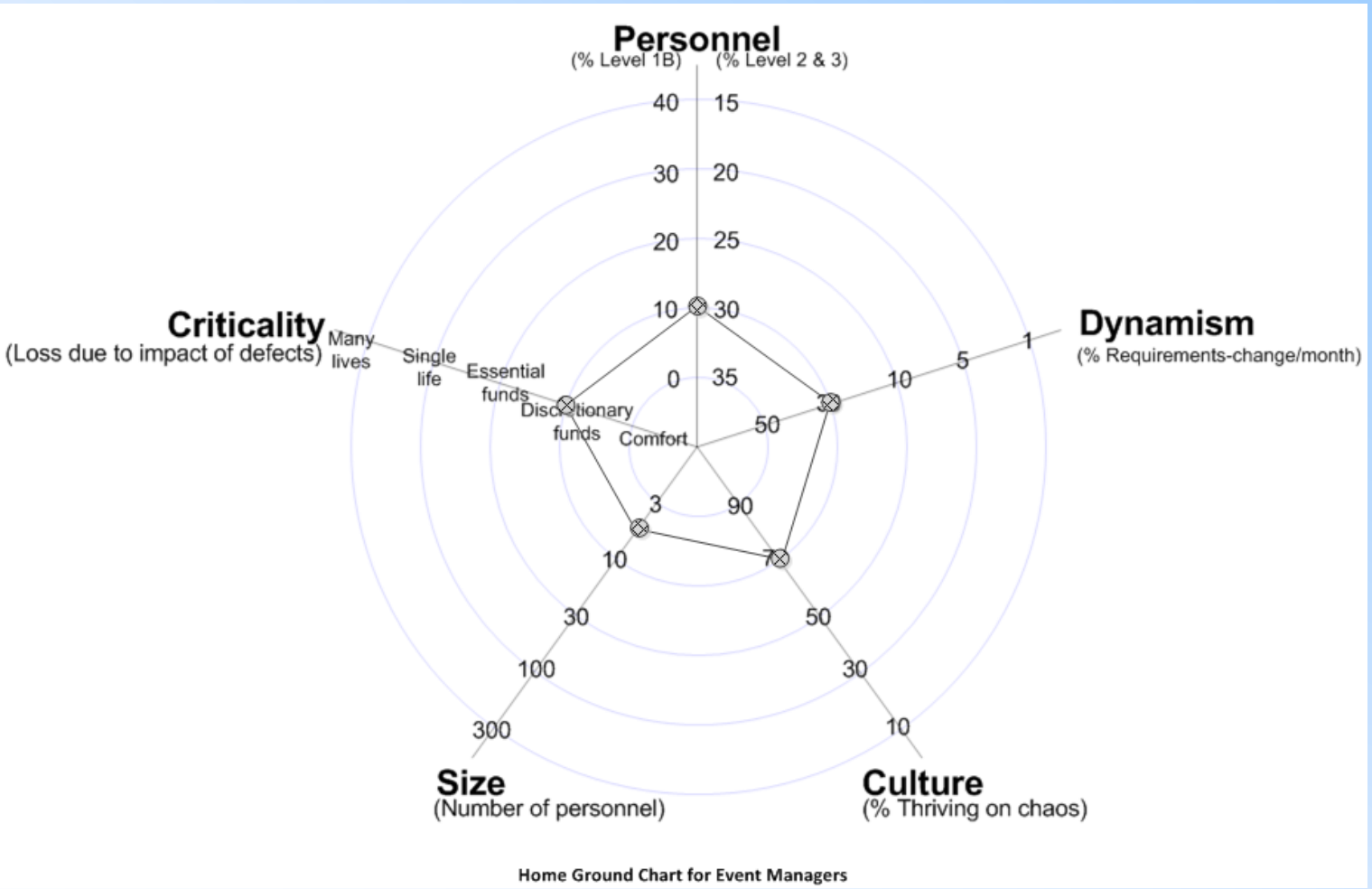
Team size	5
Team type	In-house/venture startup; collocated
Failure risks	Venture capital; manual effort
Clients	Single collocated representative
Requirements	Goals generally known; details emergent
Architecture	Provided by single COTS package
Refactoring	Inexpensive with skilled people
Primary objective	Rapid value

Risk Ratings for Event Managers (Small, Non-Critical Application)

E-Tech. Technology uncertainties	1
E-Coord. Many stakeholders	0
E-Cmplx. Complex system of systems	0
A-Scale. Scalability and criticality	0
A-YAGNI. Use of simple design	0
A-Churn. Personnel turnover	2
A-Skill. Not enough people skilled in agile methods	0
P-Change. Rapid change	3
P-Speed. Need for rapid results	3
P-Emerge. Emergent requirements	3
P-Skill. Not enough people skilled in plan-driven methods	0

Risk Scale:
0–Minimal; 1–Moderate; 2–Serious but manageable; 3–Very serious but manageable; 4–Showstopper

Home Ground Chart for Event Managers (Small, Non-Critical Application)



Home Ground Chart for Event Managers

Characteristics of NICSM (Very Large, Highly Critical Application)

An agent-based planning system for national crisis managements (based on risk patterns observed in the US DARPA and US Army Future Combat Systems program).

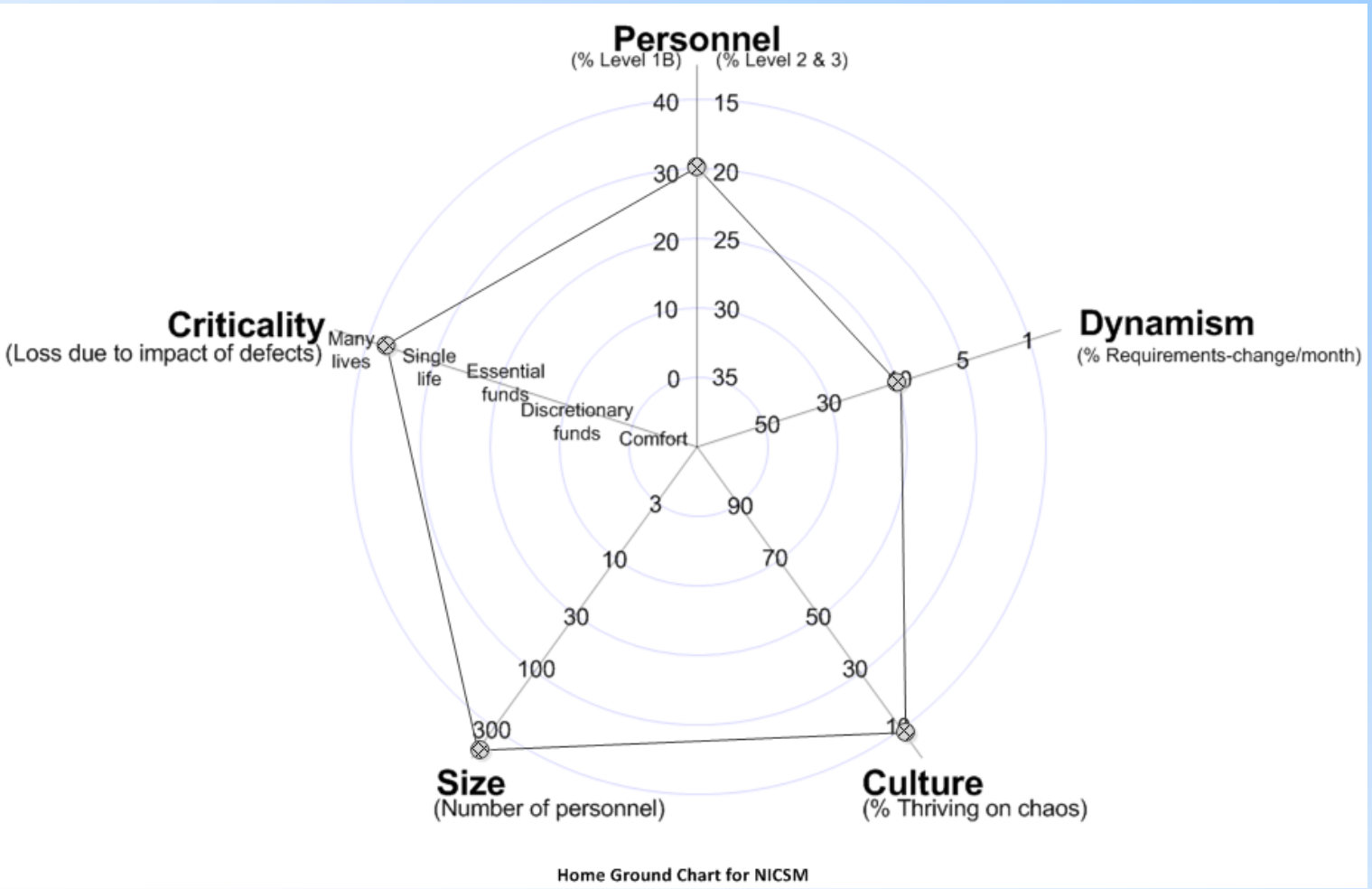
Team size	500
Team type	Highly distributed, multiorganization
Failure risks	Large loss of life
Clients	Many success-critical stakeholders
Requirements	Some parts relatively stable, others volatile, emergent
Architecture	System of systems; many COTS packages
Refactoring	Feasible only within some subsystems
Primary objective	Rapid response, safety, security, scalability, adaptability

Risk Ratings for NICSM (Very Large, Highly Critical Application)

E-Tech. Technology uncertainties	3
E-Coord. Many stakeholders	3
E-Cmplx. Complex system of systems	3
A-Scale. Scalability and criticality	2-4
A-YAGNI. Use of simple design	2-4
A-Churn. Personnel turnover	2
A-Skill. Not enough people skilled in agile methods	2-3
P-Change. Rapid change	2
P-Speed. Need for rapid results	2
P-Emerge. Emergent requirements	2
P-Skill. Not enough people skilled in plan-driven methods	2

Risk Scale:
0–Minimal; 1–Moderate; 2–Serious but manageable; 3–Very serious but manageable; 4–Showstopper

Home Ground Chart for NICSM (Very Large, Highly Critical Application)



Home Ground Chart for NICSM

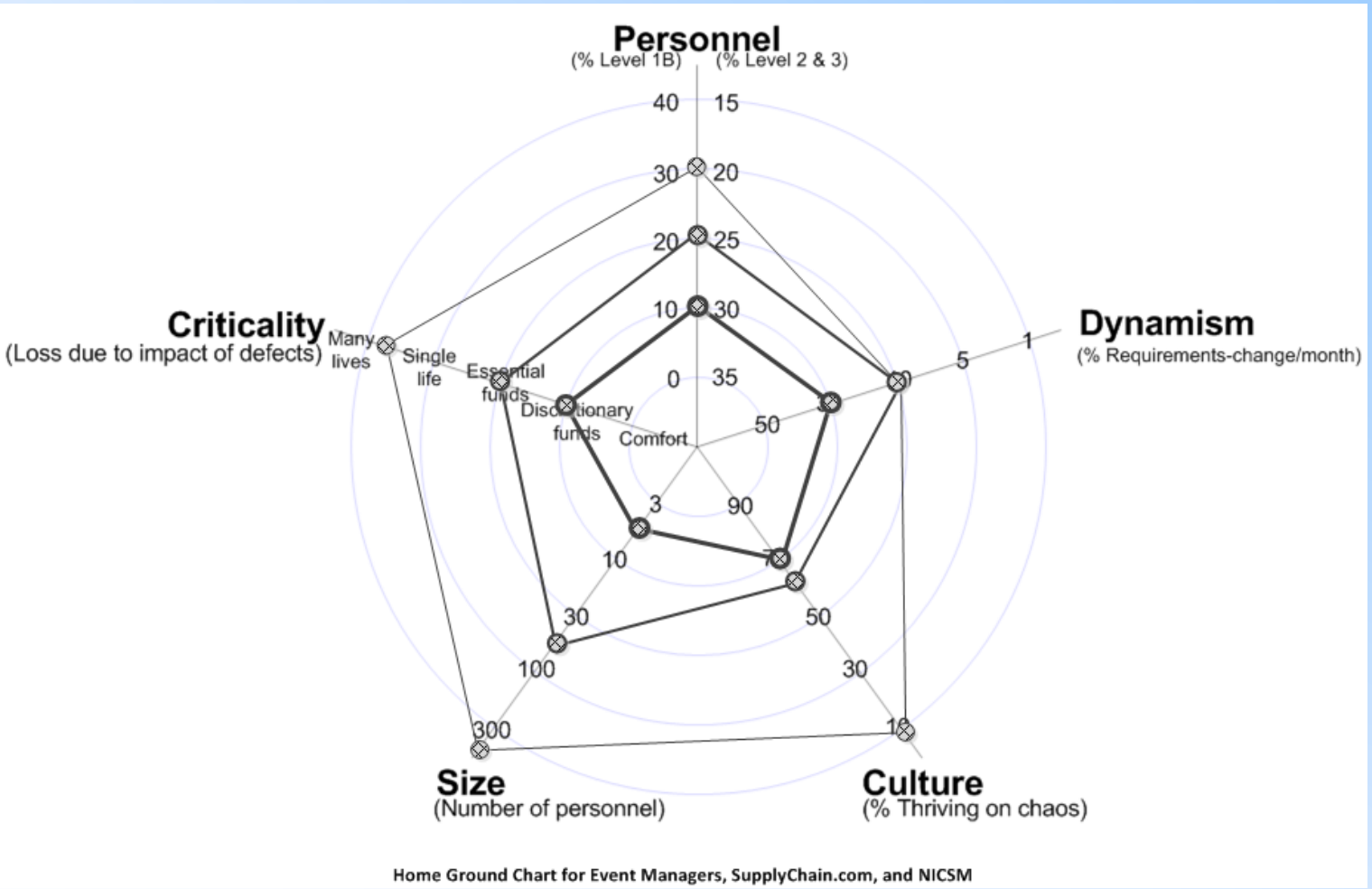
Summary of Example Application Characteristics

Application	Event Planning	SupplyChain Mgmt	National Crisis Mgmt
Team size	5	50	500
Team type	In-house/venture startup; collocated	Distributed; often multiorganization	Highly distributed, multiorganization
Failure risks	Venture capital; manual effort	Major business losses	Large loss of life
Clients	Single collocated representative	Multiple success-critical stakeholders	Many success-critical stakeholders
Requirements	Goals generally known; details emergent	Some parts relatively stable, others volatile, emergent	Some parts relatively stable, others volatile, emergent
Architecture	Provided by single COTS package	Mostly provided by small number of COTS packages	System of systems; many COTS packages
Refactoring	Inexpensive with skilled people	More expensive with mix of people skills	Feasible only within some subsystems
Primary objective	Rapid value	Rapid value increase, dependability, adaptability	Rapid response, safety, security, scalability, adaptability

Summary of Example Application Risk Ratings

Application	Event Planning	SupplyChain Mgmt	National Crisis Mgmt
E-Tech. Technology uncertainties	1	2	3
E-Coord. Many stakeholders	0	1	3
E-Cmplx. Complex system of systems	0	1	3
A-Scale. Scalability and criticality	0	2	2-4
A-YAGNI. Use of simple design	0	2	2-4
A-Churn. Personnel turnover	2	1	2
A-Skill. Not enough people skilled in agile methods	0	1	2-3
P-Change. Rapid change	3	2	2
P-Speed. Need for rapid results	3	2	2
P-Emerge. Emergent requirements	3	2	2
P-Skill. Not enough people skilled in plan-driven methods	0	1	2

Summary of Example Application Home Ground Charts



Home Ground Chart for Event Managers, SupplyChain.com, and NICSM

Contents

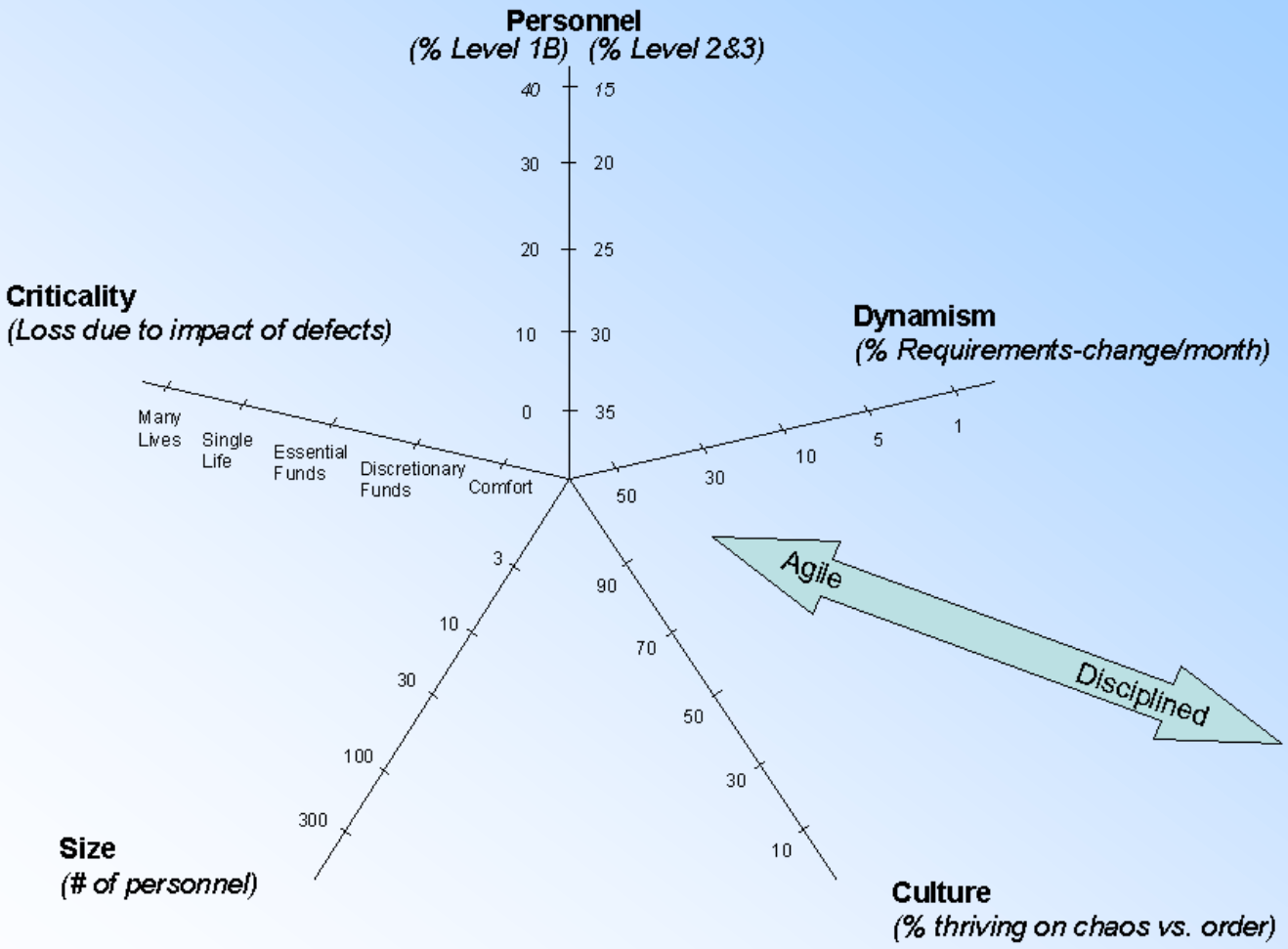
The Top Six Conclusions

1. No Agile or Plan-driven method silver bullet
2. Agile and Plan-driven method home grounds
3. Future applications will need both Agility and Discipline
4. Balanced Agility-Discipline methods are emerging
5. Build your method up – Don't tailor it down
6. Focus less on methods – More on people, values, communication, and expectations management

No Agile or Plan-Driven Silver Bullet

- Failures of Agile methods:
  - Complexity
  - Conformity
  - Scaling-up
- Failures of Plan-driven methods:
  - Complexity
  - Changeability (documentation rework)

Agile and Plan-Driven Home Grounds



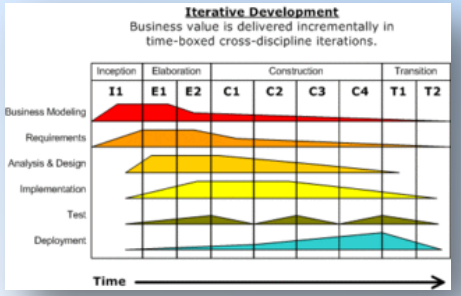
Future Applications Will Need Both Agility and Discipline

- Both agility and discipline are critical to future software success
- Not much crossover between the two home grounds in the past but things have changed
- Small projects need to be able to scale-up
- Large projects need to be more nimble

Balanced Agility-Discipline Methods are Emerging

- Agile methods have emerging approaches for achieving balance
  - Crystal Orange
  - DSDM
  - FDD
  - Lean Development

- Plan-driven methods that are more lightweight
  - RUP
  - UPEDU



Build Your Method Up – Don't Tailor It Down

- Build your method up
  - Start with relatively minimal sets of practices
  - Add extras only if necessary and justified
- Don't tailor it down
  - Tailoring-down all-inclusive methods requires highly experienced staff
  - Leads to unnecessary expenses in time and resources

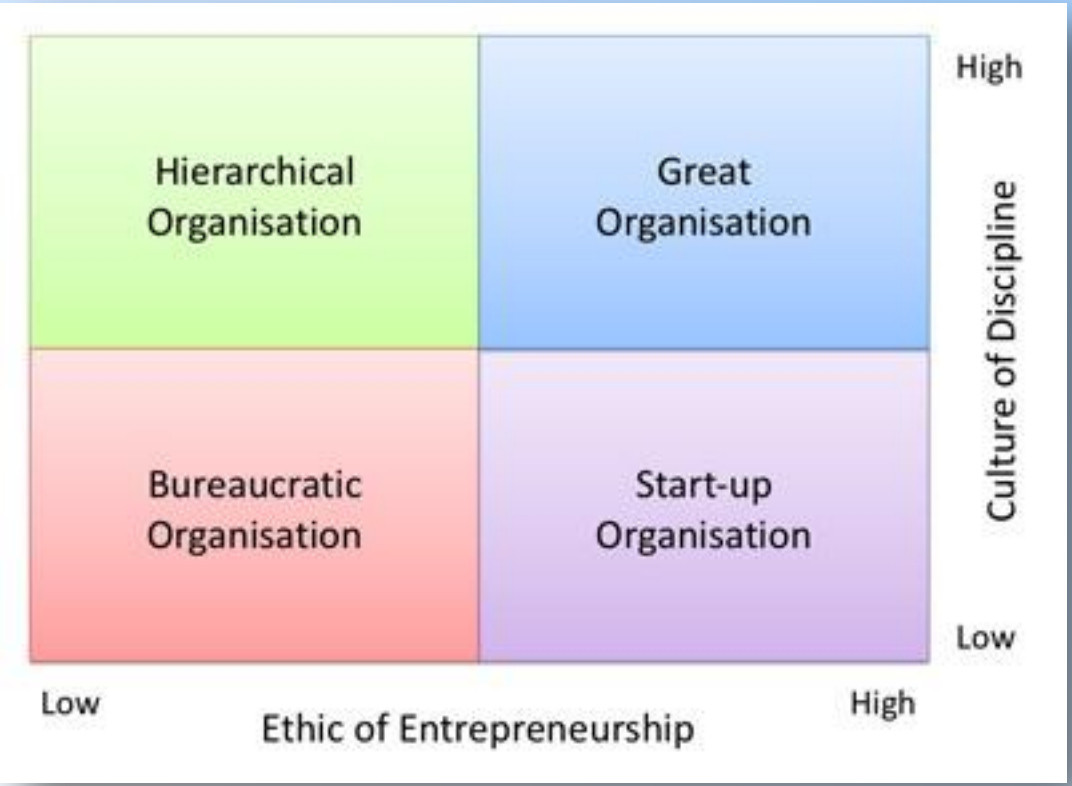
Focus Less on Methods – More on People, Values, Communication, and Expectations Management

- Good people and teams trump other factors
- Software engineering is done "of the people, by the people and, for the people."
- Challenge of reconciling value propositions of various stakeholders (user, customer, developer, etc.) into a win-win for all
- Challenge of communication among teams
- Poor expectations management causes many software project failures

What Can You Do Next about Balancing Agility and Discipline?

- Start with a self-assessment
- Key Stakeholders: Users, customers, developers, suppliers, strategic partners
- Key Trends:
    - Increased pace of change and need for agility
    - Software dependability and need for discipline
    - Ability to quickly satisfy stakeholders' evolving value props
    - Increasing gap between supply & demand for Cockburn Levels 2 and 3 people
    - Ability to cope with existing and emerging technical challenges (COTS integration, web, mobile, etc.)

What Can You Do Next about Balancing Agility and Discipline? (contd.)



Collins' Good-to-Great Matrix of Creative Discipline

Steps Toward Balancing Software Development Agility and Discipline

- Graph your organization's "typical project" on 5D charts
  - Current; Future ('n' years from now)
- Identify major needs for organizational change
  - With respect to balancing agility and discipline
  - With respect to future trends and environmental risks
- Identify most critical first steps for improvement
- Identify pilot project for applying first steps
  - Application, staffing
  - Needs for education, teambuilding, culture change
- Summarize in bulleted strategic plan

Final Takeaways

- Both Plan-driven and Agile methods aim to:
  - Satisfy customers
  - Meet cost and schedule parameters
- Home grounds exists but the need/opportunity for integration is expanding
- Self-assessment is recommended for your organization
  - Locate your place in the home ground space
  - Identify areas of change and how they might move your location
  - Look for ways to integrate methods to better respond to change
- People concerns dominate method concerns

References

Barry Boehm

http://en.wikipedia.org/wiki/Barry_Boehm

http://sunset.usc.edu/Research_Group/barry.html

Richard Turner

http://en.wikipedia.org/wiki/Richard_Turner_(software)

http://sse.stevens.edu/nc/people/faculty/faculty/1122/

Observations on Balancing Discipline and Agility

http://agile2003.agilealliance.org/files/P4Paper.pdf