PRINCIPLE-BASED PROJECT LEADERSHIP

SIMPLICITY AND OUTCOMES IN SOFTWARE-ENABLED PROJECTS

BY

ADAM C RUSSELL

BETA VERSION 0.90 - 31ST DECEMBER 2015

Principle-based Project Leadership – Beta Version

This book is a work of non-fiction. Names, characters, businesses, organizations, places, events and incidents have generally been written to obfuscate the actual events. Any resemblance to actual persons, living or dead, events, or locales is entirely regrettable.

For information contact; address www.adamonprojects.com

Book and Cover design by Adam Russell (temporary placeholder)

ISBN: (not yet applied for)

Beta Edition: 31st December, 2015

10 9 8 7 6 5 4 3 2 1

TABLE OF CONTENTS

Chapter 1 – Introduction
Notes on the Beta Version
The Context to this Book
Defining a Project Manager
About this book
Chapter 2 – Why do Information Technology Projects Fail?
Introduction
Projects & Failure: A Long-Term Affair
Cobbs Paradox
Knowing the causes of your own destiny
Methodology Capture
Complicatedness
Making problems “Wicked”
Projects fail because we make them Fail
Why have organizations not resolved "Cobbs Paradox"?
Methodology Structure, Selection and Use
Discipline
Undermining the Role of Project Manager
So the Solution is another Methodology?
Conclusion
Chapter 3 – Why is Leading Software Development Projects So Hard?
Introduction
Software Development is Hard
Managing Software Development is even harder
Fundamental Problems Elaborated
The Endless Search
Conclusion
Chapter 4 – The Failure of Methodologies
Introduction
What is a methodology?
Pre-Waterfall
Waterfall
Plan-based or Deterministic Methodologies
Alternative Approaches to Software Development
The Agile Movement
The Baby and the Bath Water
The Double-edged sword of Methodologies
Benefits of Methodologies

Problems with methodologies
The Great Methodological Paradox
Methodologies for Software Development Project Management
Using Methodologies
Who uses methodologies properly anyway?
The Agile Manifesto
Plan-based ("Waterfall") vs Agile
The "Paradigm Trap"
How do we avoid the Paradigm Trap?
Zombie Processes
Don't throw the baby out with the bathwater
Conclusion
Chapter 5 – A Bridge to the Future: Foundations of Principle-based Management
Introduction
The Conflict between Technology and Art
Principle-Centered Management
Managing Complexity
Hard and Soft Management Approaches
Complex Adaptive Systems
Holacracy
Self-Organizing Teams
Trusting People to do their jobs
Thinking fast and slow
Nudge Theory
What does this have to do with project leadership?
Principles lead to models of behavior
Principle-Based Leadership
Genesis of Principle-based Project Management
Reject Methodologies; Embrace "Hacks"
What is a Hack
Checklists
Conclusion
Chapter 6 – Principle-based Project Leadership
Introduction
Principle-based Project Leadership
Guided by higher-order principles
How does Principle-based Project Leadership work?
Project Mindfulness
Stand Outside your project
Project Management: What are our Values?

Building a delivery framework from the inside out
Building your own principles
Conclusion
Chapter 7 – The Project Action Principles
Introduction
Project Action Principles
Structure of the Project Action Principles
Dealing with Complexity: The dynamics of PAP's
Conclusion
Chapter 8 – Project Action Principle #1: Achieve Outcomes, Rapidly
Introduction
Project outcomes
Outcome-less interactions
Work-based outcomes
Bringing an outcome focus to your project
Strategies for PAP #1
Introduction
Models for PAP #1
Techniques for PAP #1
Conclusion
Chapter 9 – Project Action Principle #2: Enable Customer Value, Interactively
Introduction
Your customer is not your friend
A project delivery problem?
Definition of a customer
Definition of customer value
Customers decide the value that a product has: nobody else
Natural Customer Advocacy and the Badass User
Customer Value Perspective
What do customers want? It's not features
Direct customer contact
Embedding a Customer Value perspective into your project
Strategies Summary for PAP #2
Models Summary for PAP #2
Techniques Summary for PAP #2
Conclusion
Chapter 10 – Project Action Principle #3: Build Shared Models, Verifiably
Project Action Principle #3

What is the secret to ‘herding cats’?
But the secret is not to herd the cats!
What are shared mental models?
The power of shared mental models
Verification of mental models
Representations of models
Strategies Summary for PAP #3
Models Summary for PAP #3
Techniques Summary for PAP #3
Conclusion
Chapter 11 – Project Action Principle #4: Eliminate Teaming Threats, Ruthlessly
Introduction
Why don’t we have better teaming?
Teaming threats
Removing teaming threats in your projects
Strategies Summary for PAP #4
Models Summary for PAP #4
Techniques Summary for PAP #4
Project Team Charter
Principle-based Self-Organizing Teamwork
Conclusion
Chapter 12 – Project Action Principle #5: Suppress Project Entropy, Selectively
Introduction
Project Entropy
Manage or suppress?
Entropy Management Leverage
Strategies Summary for PAP #5
Models Summary for PAP #5
Techniques Summary for PAP #5
Conclusion
Chapter 13 – Using the Project Action Principles
Introduction
This is not a typical “How To” guide
Why do the 5 Project Action Principles work?
Working with existing methodologies
Simplicity, always Simplicity
The Leverage and Extension of Human Capabilities
Tools Techniques and Methods
How to Use

Developing Your Project Management Principles
Conclusion
Appendices
Introduction
Summary
Appendix 1: Errors Created by Team Misfocus
Introduction
How to Use The Observations
Team Imbalances
Appendix 2: Pursuing the Perfect Project Manager Revisited
Rethinking “The Perfect Project Manager” by Tom Peters
The original post
The Honey in the Lion
Conclusion
Appendix 3: Customer Advocacy and Understanding Customer Value
Introduction
Customer Advocacy
A Taxonomy of Advocacy
Conclusion
Appendix 4: Are the Risk Management activities on your project a WOFTAM?

Chapter 1 – Introduction

Notes on the Beta Version

I am publishing this book before it is 100% complete; to participate in a process called "lean publishing". From a lean perspective, I am already way past the MVP for this book, but of course I wanted to get this volume to its best position before I pushed it out the door. Ah vanity! And now, of course, comes hubris.

I've gotten it to the point where it says pretty much what I want it to say, but perhaps not as cleanly, clearly and eloquently as desired.

A modestly experienced person who works in and around the cut and thrust of project delivery will be able to work through the content, but I'm demanding much more tolerance and forgiveness of my readers than I should.

Parts of the book have had some professional editing, but there is quite a bit of polishing and cross-checking to do in order that the points I want to make are clear and accessible. It needs a good book-butcher to slice it up so that I can put it back together again.

It needs some illustrations and extensive formatting to make it suitable for both e-book distribution via Kindle and also for printed book production.

I have released it for paid sales on leanpub.org quite frankly because sales and donations are how I'm going to fund the completion of this volume and its successors. Anyone buying the beta version will receive the final version in its e-book ready state, and also get an optional cost-price deal on the printed version.

For those of you reading this after purchase or donation, thank you very much.

I dearly want some feedback on this book so that I can continue to improve its contents.

For updates, additional content and other resources, visit http://www.adamonprojects.com/

Thank you again.

The Context to this Book

This book is based primarily on my own personal experience as a project manager working on software-enabled projects.

As to the much troubled term ‘project manager’, we will discuss what the term means later in the book, but for now just take this term to mean anyone charged, at any level of the corporate hierarchy, with any material part of orchestrating people in an initiative to deliver a software-enabled solution to end-customers.

The term ‘software-enabled’ is used to mean any project where software is the means to deliver primary capability to the end-customer or consumer of the project, even if there is a substantial component of computing infrastructure (e.g. hardware, networking equipment and system software) that is delivered by the same project. Examples of this type of project range from a stand-alone Windows app or mobile app on an iPhone through to a greenfield corporate website portal or email project.

I have been fortunate to have worked a very wide range of roles in the information technology industry over a long period. Starting as a software developer to managing the delivery of solutions, I’ve spent nearly 40 years building software-based solutions, starting with PC Kits in the 70’s moving to Distributed PC & Mini-computer and mainframe systems in the 80’s, all of which had very broad industry targets. Starting in the 90’s, I began focus on industry verticals, starting with Telco solutions in the 90’s and moving to Internet-enabled solutions and Digital Content solutions since the 2000’s.

Although the bulk of my experience has been in Project Management, I’ve also included roles in Software Development, Professional Services Sales, Startups, Business Development, Product Management, Solutions Lead and Resources Management, and probably a few I don’t remember

The common thread of this experience has been my interest in finding and delivering solution outcomes based primarily on software technology. And it has been about building products destined for use by an end-user, both consumers and enterprise

end-users.

The context to this experience and learning has been about domains dominated by:

- High rates of change in the industry, either through massive growth (e.g. PC and Internet applications) or disruption (industry rationalization, strategic market shifts, etc.); and
- Development and delivery of largely “soft” or “intangible” deliverables, such as digital publishing, online and mobile applications, software development, and intellectual property development (e.g. content, IP or knowledge management).

In a nutshell, the context in which I have worked has always been “we need this product/project/thing developed in a ‘Big Damn Hurry’”. No schedule or launch date that I’ve ever proposed has been soon enough, the level of chaos has always been high, and new or revised requirements have cascaded down like hailstones on a tin roof.

Since I began my first attempt at programming (an old HP calculator, and then an Intel 8085 SDK Kit), I’ve been fascinated by the process of software development and the way in which people interact with each other in such a domain of "hurry up": when going slow is standing still or going backwards.

I’ve always been happy to use the mandated processes, e.g. the corporate project management methodology, but I become frustrated and unsatisfied in accepting ways of doing things that don’t work: the “mandatory” corporate methodology that has gaps and errors that have to be “filled in on the fly” (and I’ve seen my share of those). Methodologies that work, and that are applicable to the problem at hand can be a thing of beauty, but I admit to being seduced at some points in my career, of being able to introduce such perfect methodologies, and of pushing to implement a new solution or “silver bullet” at work.

I’ve been thinking critically about approaches to project management pretty much since my first “real” project with Wang Labs in Australia, developing tools and reading extensively on approaches to project management and software development.

The longer I work in this industry, the more it seems that the answers are not in technologically derived processes but in how people work together to produce outcomes. The more I learn about Project Management, the more I've come to realize that it is much more about 'management' than it is about 'project's.

And my view is that the endless debate about people vs methodology: the debate about "hard" versus 'soft' skills debate in many forums is not the right debate; the questions asked are not the right questions. Nor do I see guidance being given to new project managers coming through their project experiences as being much more than a reconstitution of existing ways of doing things. The same concepts promoted in different clothes, the same "rules" being dished out as fact.

I believe that there are much simpler and much more natural ways for people to work together to achieve great project outcomes, even in the face of high technology and "chaotic" environments. These ways may need changes of behavior or attitudes, but none any greater than demanded by Agile methodologies.

I think we can get projects delivered faster and cheaper, and have more fun doing it, than most people do under current regimes.

Hence the reason for this book to share my findings.

Defining a Project Manager

At the beginning of this introduction we touched on the meaning of the term 'project manager' we haven't yet agreed (in this book) on a definition of who or what we mean by a Project Manager

With all the changes and competition between methodologies that has occurred, and particularly in terms of the elimination or downgrading of the role, the question now becomes: "What is a project manager?"

Is it someone who actually leads the project? Or a person who executes project management processes?

The traditional methodological view (outside of Agile) is that a "project manager" is someone who manages a "project" according

to the processes and practices of the methodology. And the definition of project is "a unique initiative that …" – no need to repeat it here.

The self-referential view from within the methodology is that a Project Manager is someone who owns and exercises the methodology. When I first read these definitions, I thought that it seemed a pointless exercise, but later I realized it was the whole point.

Tom Peters, back in 1999, described the "perfect" project manager (see http://tompeters.com/columns/pursuing-the-perfect-project-manager/).

Peters was an early proponent (at least to my ears) of the "everything is a project" view of business operations, and his view was that if you broke down the relatively narrow definition of a project, the term 'project manager' was much harder to define in terms of activities, or outside the context of a project management methodology.

> ***"But as 'project' and 'network' become the norm, 'who's in charge?' becomes problematic. Everyone needs to learn to work in teams, 'with' multiple, independent experts, often from multiple, independent companies; each will be dependent upon all the others voluntarily giving their best. The new lead actor/'boss'—the project manager—must learn to command and coach; that is, to deal with paradox. Here are eight dilemmas she or he must master"*** **– Tom Peters in "Pursuing the Perfect Project Manager"**

Peters described 8 dilemmas that a 'perfect' project manager needed to resolve:

1. Total ego/no ego
2. Autocrat/delegator
3. Leader/manager
4. Tolerate ambiguity/pursue perfection
5. Oral/written

6. Acknowledge complexity/champion simplicity
7. Think big/think small
8. Impatient/patient

We'll come back to Peter's dilemmas later in the book (see Appendix 2), but for now the takeaway is that there is more, a lot more, to being a 'project manager' than executing a methodology.

Steve Berkun addresses this in his book 'Making Things Happen: Mastering Project Management' (2003) in the following way:

> ***"Refer to whoever is involved in project leadership and management activity… leading the team in figuring out what the project is (planning, scheduling, and requirements gathering), shepherding the project through design and development work (communication, decision making, and mid-game strategy), and driving the project through to completion (leadership, crisis management, and end-game strategy)".***

Berkun also says, and I expect that it is your experience as it is mine, that such a role may be the "person doing project management tasks, even though it's not her primary job" or "person thinking about the project at large." and says that it is less about role or title than it is about the activity that they are performing.

There has been much publicity about different organizations who have done away with the role of Project Manager completely, assigning whatever outcomes a PM is supposed to achieve, either to the dustbin (as being too much nit-picking process) or to other roles, such as the Scrum Master in Scrum. The Scrum methodology uses the term "Scrum Master" for the role, but the term "project management" to describe part of what the Scrum Master actually does.

Other management approaches use different terminology but with the same intent: the 'person' instantiation of a project manager is not required, just the skillset and attitudes.

The concept of considering a PM in terms of the role not the person does little to address the problems that the function of project management has experienced in the progressive erosion of its value.

We often joke that if everyone had the hustle to get things done on or ahead of time – if everyone worked as their own project manager – we PM's wouldn't have jobs. This is true to the extent that in most teams the gaps are so large that it requires dedicated effort from a single person (or even more) to achieve the desired outcomes.

Even if the individual accountability and team drive reached its maximum levels, the issues that face PM's would just face other people.

So bottom line: we need the outcomes that PM's can provide but we dispute the means with which they are delivered: a dilemma that manifests itself every day in many development shops around the world.

What we need, and what this book attempts to convey, is a way to resolve this dilemma.

About this book

I like to think that this is the first real book on agile project management, because it doesn't include a single prescriptive step. I'm sure that others would disagree, but, as we will cover later, there are no degrees of prescriptiveness, even a little will trigger or promote a style of thinking that is inimical to agile objectives.

The world is full of consultants, managers and others who talk about non-prescriptive approaches but finish up by saying "do this", and even if they don't, teams or individuals can easily adopt prescriptive implementations of methodologies that were not intended to be prescriptive.

Agile, for example, tells you that the best way to keep communications going with the team is to have a daily "Stand-up". There are plenty of ways to keep communications going, but try to run an agile project without having a standup, and you will most likely find out how prescriptive this can be.

All humor aside, what this book does is to teach you how to think flexibly and with discipline about how to tackle any project, at least in the digital product and IT space.

I have discovered that in order to let people excel, you really have to be non-prescriptive in terms of how they actually do things. I mean **really** non-prescriptive.

The methodology "wars" between plan-driven and agile promote the concept that plan-driven methodologies are prescriptive and agile methodologies are not. The problem is that this is not correct, at least not in practice.

Agile methodologies in practice seems to be just as prescriptive as any other, and bad-agile practice even more so.

I have come to believe that what drives us to do excellent work is not practices but **values**; and that set of values that works for one team or organization may not work for another. Indeed, as teams and organizations change rapidly, it is possible that no set of values is in place for more than one, or at most a few, projects.

Principles seem to me to be values encoded as statements that can be absorbed by people. Principles are powerful but simple statements that build on the way people perceive the world around them, and therefore drive behavioral outcomes.

The agile principles for software development are such values encoded as statements. This book is about how principle-based management works, and how to modulate those principles in the quest for effective end-to-end project delivery.

This principle-based approach originates not only on my experience but is also founded on extensive research performed by the world's leading writers and practitioners of project management.

I've called this approach Principle-based Project Leadership (PBPL)

Principle-based Project Leadership (PBPL) will give you the intended agile perspective: that of the ability to work dynamically with people on tasks of importance and mutual enjoyment, knowing that you are not wasting your time or producing a dud.

PBPL can add value in any domain which enjoys high rates of change, require quick time to complete, and has all (or at least the majority) of the deliverable outcomes as intangible products such as software, services or information.

And lastly, PBPL can be used as a supporting guide to any existing methodology, or it can be simply used as the essential guide on how to manage any project. There is no "either-or" choice, just a graded evolution of how you approach and understand your given methodology, and how you employ its knowledge and apply it to your projects.

Even if you make no changes in how you run your projects, this book spends some time looking at how methodologies work (or don't) in helping to manage projects and this will give you a useful perspective on that whole relationship.

Amongst other things, Principle-based Project Leadership (PBPL) will explain how you can make rapid starts to projects when it counts the most, focus on the most important things ahead of distractions and waste, deliver real value to enable customers, and build up a harmonious team

How can this all work? By distilling everything that everyone has ever told you that is right about how to manage software projects and letting your brain do the rest. PBPL will free the creative and powerful cognitive engines of you and your team to do the right thing at the right time.

All in all it is about simplicity: adopting the simplest approach and the simplest solution for any given project that will achieve the project's goals

PBPL absolutely does not reject methodology as its first foundation to existence: if a given methodology is working for you and you believe that it will continue to do so, then why change?

PBPL does not say you'll never use another tool or pre-defined process, but it does allow you to use whatever elements of those existing stores of learning that are appropriate and productive.

It is not a silver bullet. It may well require you to exercise even more discipline that you may be used to, but only that much that is

necessary for each particular project.

It will not be easy to open up this process and allow it to help you work simpler, faster, but it is absolutely worth it, in my opinion.

Good luck.

Chapter 2 – Why do Information Technology Projects Fail?

Introduction

Information Technology (IT) projects generally have a very high rate of under-performance, either a complete failure to achieve any of the planned outcomes or only a partial delivery of the stated outcomes. In addition we have to consider those projects that suffer from a high-level of "Project Entropy" and that are late, over budget and / or create an unreasonable level of stress in the project participants (i.e. any of the project stakeholders).

There have been many studies that assess the failure rates of software and technology projects. One, the "McKinsey Oxford Reference Class Forecasting for IT Projects Study" indicated that 64% of projects experienced cost overruns and 78% experienced schedule overruns. This study was probably the most comprehensive, covering 3,607 projects worth a combined USD84 billon.

Standish Report cites 488 Major US Federal Government IT Programs over the past decade of which only 4% were rated as "successful".

There is little point in enumerating these studies or statistics in much more detail, the information is readily accessible and the website points to resources as they are found. Most of these studies identify reasons for the project failures and recommend actions to be taken to avoid these problems in the future. Literally thousands of books and online articles quote these studies and/or attempt to identify the various factors that cause these failures.

Even if projects don't fail, as covered in these studies, many projects still become "impaired" in some way. Even though they deliver, projects may still suffer the impacts of "Project Entropy", my term for all the negative project impacts such as design problems, technical defects, requirements changes, office bureaucracy, risk events, and all those events that arise from the human factor.

Impaired projects can still produce a result, but execute sub-

optimally and produce solutions that may be ill-suited to the problem, be late or too expensive.

"Impaired" projects may simply be an unpleasant place to work. Unpleasant places to work produce unhappy people, which over time decreases morale and engagement.

Projects & Failure: A Long-Term Affair

Experiencing serious problems with the delivery of IT projects has a long-standing history. Endless reasons are given for project failures at many different levels of practice: from bad strategic choices to poor individual engagement, the post-fact explanation of failure covers every reason that you could ever think of, and probably a few more. All of us who have worked on IT or digital projects would recognize a depressing share of these stated problems.

Although we touch on the analyzed delivery problems and recommendations, in this book I want to look at this issue from another angle. I want to look, not at what caused the problems, but why we let these problems impact our projects. The question should not be "what are the causes of project failure", rather the question should be "why do we still let these causes occur and generate project failures at these rates?".

Quantitative industry studies on project failure have been published since at least the late 1990's if not earlier, not to mention the individual and organizational learning of the last 50 years.

The landmark book on this issue is Fred Brooks "Mythical Man Month" (actually series of essays) originally published in 1975, which was triggered primarily by issues that he encountered whilst building large systems programming products at IBM as far back as the 1960's: the very early days of large-scale program development.

> ***"Large-system programming has over the past decade been such a tar pit, and many great and powerful beasts have thrashed violently in it. Most have emerged with running systems—few have met goals, schedules, and budgets. Large and small, massive or wiry, team after***

***team has become entangled in the tar. No one thing seems to cause the difficulty— any particular paw can be pulled away. But the accumulation of simultaneous and interacting factors brings slower and slower motion. Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it. But we must try to understand it if we are to solve it."* -- Fred Brooks in 'The Mythical Man-Month: Essays on Software Engineering' (Anniversary Edition - 2nd Edition) Pearson Education**

Brooks attributes the first edition of "Mythical Man Month" as a 'belated answer to Tom Watson's probing questions as to why programming is hard to manage.' There were many important themes to take out of this book, but the most relevant one to our narrative here is that there is no simple way to develop software. The 'werewolf' of software development was never going to be killed with a 'silver bullet'.

It seems that from 1975 following "Mythical Man Month" declaration that there was no 'silver bullet', software development leadership has been trying to disprove Brooks in that assertion. Certainly from an external or historical perspective, the primary response has been to search for the "silver bullet" at least in part by the development of various methodologies that attempt to codify all or part of the end-to-end project development process.

Cobbs Paradox

With so much knowledge of past failure summarized so clearly, we have so many reasons in place to not fail. Particularly when we have so many methodologies, tools and techniques that are designed to prevent that failure.

Martin Cobb, CIO for the Secretariat of the Treasury Board of Canada in 1995, coined the so-called "Cobbs Paradox in this light, saying:

"We know why projects fail; we know how to prevent their failure – so why do they still fail?" - Martin Cobb

The implication of Cobbs Paradox is that we know, right from day one of any given project, the reasons why it may become impaired, and that some form of impairment was a high probability...

We know this and yet we still allow it to happen. Why? If we already know the causes of your own destiny, why do we not ameliorate them?

Knowing the causes of your own destiny

The domain of project management has a multiplicity of tools, techniques and methodologies. Project managers have access to numerous service offerings for training, certification and consulting: there are courses of study in Project Management that confer Masters and Doctoral-level degrees.

Industry organizations such as PMI marshal considerable resources to analyze, define and propagate project management skills, tools and methodologies to a wide constituency

Online, we have an endless number of blogs offering advice, templates, "how-to" guides, analysis and discussions via social media. And we've seen an explosion of a variety and change in approach in the Agile era. There are some who have criticized Agile as being a consulting-led industry. There is certainly no shortage of Agile consultants.

Despite all this support and prescription on how to deliver projects, our projects still fail.

Other industries (e.g. Transportation) and professions (e.g. Medical) have semi-independent organizations that monitor failures, regulate individual and organizational practice and orchestrate initiatives to improve the profession, but in the IT industry this approach is left to many individual groups, usually who have vested interests in the outcomes. IT projects leave it to individual organizations themselves to monitor these trends. The problem with this is reporting bias. Anecdotally, it appears that

self-reported data on project success is materially higher than relatively independent analyses and studies. [Reference needed]

Even though we know that these methodologies cannot be the "silver bullet" that we want to believe them to be, we still use and promote their use.

Methodology Capture

Project Management places a high level of reliance or faith in the benefits of adopting and following a pre-existing methodology to improve success. Some reports of project failures single out the use of a formal methodology as a differentiator in terms of project success rates.

We will cover methodologies in more detail in Chapter 4, what is relevant at this point is that often the methodology chosen to help solve the problem of failure or impairment is not the "silver bullet" that everyone expects, but actually makes it harder to execute the project effectively.

Many common groups of problems appearing in projects, and event portfolios of projects, can be attributed back to the methodology selected being inappropriately selected or employed in the organization and/or on the project itself.

Organizations with larger portfolios typically have big investments in any given methodological underpinning of their product delivery processes, and so are reluctant to change, but in so doing are "captured" by the methodology.

Chapter 4 will cover the issues of methodologies in more detail.

Complicatedness

A focus on methodologies invariably results in a focus on control; even Agile methodologies are in essence an attempt to control the team members and stakeholders into certain kinds of actions and behaviors.

This focus on control has been in contrast to expanding variability and complexity in the project context. For example, consider the market context for the project, the internal business

context of an organization and the rapidly evolving technology context inside and outside the organization. There are multiple degrees of freedom in the evolution of complexity.

Businesses have become significantly more complicated in the past 15 years, and our problem domains have accelerated in their complexity as the rate of solution change increases. Boston Consulting Group researchers Yves Morieux and Peter Tollman estimate that business complexity has multiplied six-fold since 1955.

> ***"over the past 15 years, the number of procedures, vertical layers, interface structures, coordination bodies, scorecards and decision approvals has increased dramatically: between 50% and 350% depending on the company"* – Yves Morieux and Peter Tollman in 'Six Simple Rules: How to Manage Complexity without Getting Complicated'**

As Tollman and Morieux found, our response to increased market complexity has been an increase in operational "complicatedness": layers of process and governance which don't actually work very well in the IT environment.

The methodologies that PM's look at have also become more complicated. For example, "brand-name" traditional methodologies such as APM or Prince 2 expand over time to deal with more problematic scenarios. For example, APM now has 47 competency areas with a mass of detail. And PMI have continued to expand and extend the PMBOK and related services, and have added "Soft Skills" and specifications for Agile project delivery processes

And at least a component of the additional organizational complexity has been in the portfolio management domain, with these organizations investing in more internal structures to support wider and deeper use of these methodologies

So we have more complex environments butting up against more complex methodologies, which produce more complexity in

implementation.

But does this expanded complexity in our methodologies, these layers and layers of process and overhead, help us address complex solutions? The answer partly lies in the nature of "wicked problems"

Making problems "Wicked"

The term "wicked problem" has been used for more than 30 years in social planning, economics and government problem solving. A "wicked problem" is a problem that isn't tractable using standardized linear approaches. As Wikipedia describes it, a "wicked problem" is:

> ***"difficult or impossible to solve because of incomplete, contradictory, and changing requirements that are often difficult to recognize. Moreover, because of complex interdependencies, the effort to solve one aspect of a wicked problem may reveal or create other problems" - Wikipedia***

This term originated to describe societally complex problems such as global climate change, the AIDS epidemic and similar intractable problems.

But for me this definition could easily describe most of the projects I have worked in for the past 20 years. Do you think that problems have intrinsically become more complex? Or, have organizations created or promoted the characteristics that lead to normal problems exhibiting the characteristics of complexity in projects, and therefore becoming more difficult to solve, more full of "wickedness"?

Most of the "branded" methodologies have their roots in practices that predate this rapid rise in complexity. They have roots in Taylor scientific management practices which emphasize process, structure, rules and controls. So at least in part applying linear and Taylorian solutions to non-linear problems can create solution issues.

The question is: why do we appear to be trying to solve

“wicked” problems? For example, if I’m just building a website or a content publishing system or an app. How can this be so complicated?

The answer is that the problems are not ‘wicked’ in and of themselves. Instead we make them ‘wicked’ by how we go about framing or/or delivering them.

Projects fail because we make them Fail

Anyone who has been involved in IT projects will have been through the ritual of a Post Implementation Review (PIR): the process or meeting that captures what happened in a project, and tries to capture reasons behind success or failure.

Anyone who has been involved in more than 1 or 2 PIR's will have witnessed their failings: the pained and complicated way in which problems are phrased to avoid pointing the finger at anyone or any group in particular; the way in which problems are described as normal or inevitable or unavoidable. Whether it is to protect a colleague's reputation, or to avoid criticizing more senior management, or to avoid guilt, or even to avoid reliving the errors or problems themselves, these statements lessen the impact of these "learnings", and this most often, except in the most disciplined of shops, reduces the PIR results to impotence.

Anyone who has been involved with IT projects will have most likely witnessed very few times that PIR's from previous projects are reviewed at the start of a new project. As well as the painful nature of the process, the assumption that the result of the PIR will not actually be used by anyone is a contributor to many project teams trying to skate on having to conduct the process at all.

The problem with this, more than anything, is that we don't call out what lawyers call "reckless disregard" of known contributors to failure when setting up new projects: people are assigned to teams more because they are available than their specific skills, risks are disregarded or even actively suppressed, development tools and components are selected because they are "new and cool" rather than because they are known and reliable. We set schedules based on arbitrary events or external milestones and we assume we know the problems rather than go through the painful and messy process of talking to users or customers and getting to the bottom of their

needs and problems.

And so it goes. We should not lose sight of the fact that projects don't fail just because "shit happens", except in the most extreme circumstances.

Projects fail because we make them fail. The answer to Cobbs Paradox is not the methodologies that are available but the actions of the people who run the projects.

Until we start thinking in these terms, we're likely to continue to make them fail, because we downplay the well-established causes of failure, and ignore the most obvious and well known contributors of success.

It is actions that we actively take within the project team, or within the broader group of stakeholders that make these problems wicked, and therefore less tractable to the tools and skills that we apply.

Why have organizations not resolved "Cobbs Paradox"?

As we said in the previous section, projects fail due to actions by individuals: deliberate, specific and accountable actions that result in project failure or impairment. The answer to “Cobbs Paradox” is us. But there would be no paradox if we had responded to this issue. Why?

Firstly, it seems that organizations do not recognize the core problem, and so they haven’t taken the first step in trying to resolve it

It’s not uncommon for project teams or even senior managers to want to represent their systems development shops as great successes, and to underplay “teething problems” that arise from development or deployment of their initiatives.

As a side-thought, I must have interviewed hundreds of candidates for roles as Project Managers in the various organizations I’ve worked for, and not once has anyone claimed other than to have delivered every project “on time, on budget and according to the specifications”. Of course they include the revised baseline that occurs after change requests, including the implied revision that occurs when the project is launched.

As a salesperson I've very rarely sat in front of a CIO or senior executive who will acknowledge much in the way of problems. Perhaps that's just me.

So, we should also be aware that the picture may not be so "rosy" as is publicized. It is so rare for managers and executives to say much other than "the project was a great success". In current business environment, no-one wants to admit failure unless it is as obvious as a plane-crash.

So the real picture behind the survey results might actually be much worse. How do we bring back the objectivity and honesty of our assessments?

Methodology Structure, Selection and Use

If you look at selection criteria commonly published for methodology use, a common driver for selecting Agile methodologies is uncertain requirements. The question is: are the requirements intrinsically uncertain? This would mean that we are unable to capture the information and structure it to come to a common set of requirements. Are the requirements intrinsically uncertain so that we get contradictory requirements from different groups or rapid changes in requirements?

Or are the requirements uncertain because the analysts are approaching the elicitation incorrectly, or the stakeholders cannot articulate their needs

The issues come primarily from how the teams apply the methodologies, tools or techniques. Are the selected methodologies, tools or techniques correct and appropriate for the project's technology and business domains? Do they meet timeline and budget targets.

Projects have problems because the methodologies are focused on the wrong thing: prescribed processes and controls.

We can make projects wicked by slavishly following a methodology to the letter. And we certainly can make projects "wicked" by being poorly trained or with a poor understanding of the methods that we apply.

Discipline

The last section touched on uncertain or unstable requirements and how that feeds into the methodology selection. My experience is that the requirements are not particularly uncertain in and of themselves: it is the participants who make them uncertain.

Often they are uncertain because the goal is wrong, so it is hard to develop good requirements

Often the participants choose not to want to define requirements in any persistent manner: it's too hard or they've been burned before, or perhaps they are just concerned. No small number of times these have been vague because the product managers intended product does not agree with some senior executive's view of the world.

We make projects "wicked" by being ill-disciplined and gaming processes or methodologies.

Sometimes it is just that the product owner does not have the time (in the sense of bandwidth) to investigate and analyze deeply. Sometimes they do not have the skills.

It is very common for product owners or other influential managers or executives to simply believe that they do not need to define requirements because they know what is best for the market, which is usually code-word for they know what is best for the organization.

Lean development certainly has a place, but between agile, lean and similar movements, there has emerged that nothing can be defined up front.

- We need to understand why our requirements appear "incomplete, contradictory, and changing", or why the interdependencies are complex.
- Is it because we are unable to capture and reconcile requirements?
- Is it that we create complex interdependencies through inappropriate design or implementation reasons?

- Do the requirements change because we take so long to get the project complete?
- Do the requirements change because the idea is bad?

If you are trying to improve the effectiveness of project delivery, the solution is not more tools and processes. It is also not more emphasis on execution effectiveness at the tool or process level.

One other way in which we make our projects "wicked" is the effective reduction in the value and power of the role of Project Manager.

Undermining the Role of Project Manager

Along with all the other problems of software projects, many organizations seem also to have emasculated or at least undermined the role and status of the "Project Manager".

I keep asking myself the question "why is Project Management becoming less and less relevant to its primary consumers?"

Have you noticed the reduction in value with which the project management practice (and its practitioners) is viewed?

This has now got to the point where many modern shops have almost completely removed the role from their organization structure.

Will we be able to devise ways so that the role can regain that relevance and become stronger voice in the end-to-end process of software development, particularly in complex integrations?

These key stakeholders and service consumers are the sponsors of projects, the teams of people who are engaged to deliver the project and the teams of people who are going to be impacted by the project, both during and after have.

In many organizations, these stakeholders have been sold the story that the role is irrelevant and in fact damaging, due primarily to the perception that project managers and their methodologies are slowing things down, process focused over outcomes, and raising difficulties unnecessarily.

So many of Project Management's customers are no longer buying what project managers have to offer, that is, if they ever did.

Project management in many ways has itself to blame. The practice is of course, not an end in itself, despite the high visibility and ceremony upon which it seems to rely. Project Management is only one of the means by which the sponsor or ultimate customer gets what they want.

PMO organizations in many ways have promoted and advanced those that adopt their processes and methodologies the best, and those who have studied and obtained their certifications. In some ways these processes have bubbled up those who are the very worst people to be prosecuting complex, dynamic and fast-paced project deliveries. There is so much pointless "makework" in projects these days that people often feel that turning up and turning the handle on an engine called "project management" is an acceptable performance of one's job.

Sponsors and other "consumers" of Project Management services are looking to new ways that promise, often falsely, quicker and faster ways of doing the same thing.

Software development teams and IT departments look at what they perceive as Project Management interference, and seek ways of removing that "interference".

There is often conflict between managers and executives as to whether these project management roles should exist, and claims that these functions can be rolled into line managers within development shops, without the need for these painful complainers and naggers.

And yet as I write this, I am watching a complex program stumble from one problem to another at the point of going live. It is 3 months late, and cannot launch due to performance and stability issues. The explanations of why this is the case are dumbfounding: reasons given that are fundamental selection criteria for the platform concerned, way back 9 months ago when there was a competitive selection. The most basic of non-functional requirements and vendor / product validations has been failed. There is no performance model, and everyone mistakes the

need for calm with the minimizing of the embarrassment that these issues should be surfacing now.

Predictions on my part that this situation could still be in place in 6 weeks were seen as sensationalist troublemaking, and yet now we look like being at least 6 weeks due to technical problems (indeterminate really) plus additional delays due to the fact that the schedule is now bumping into fixed environmental issues, like football season and racing season events, key personnel holidays, and the Christmas / new year shutdowns.

Traditional hierarchical scheduling is virtually pointless in a bottom up environment driven by Kanban or scrum, particularly any sprint-based delivery model. But executives still seem to want these global schedules to show when the project will be complete. Executives and mid-managers are still adjusting these schedules to fit arbitrary end-dates, and then being (or feigning) anger or pained surprise when they don't work.

Project managers are reduced to naggers and hustlers who help people do their own jobs, deal with basic lack of accountability and bad teamwork, and doing basic menial work like organizing meetings, because it is the only way that those sorts of things will get done. Facilitating outcomes and basically "chewing people's food for them".

And the primary reason I think that I've stayed in this business for so long: the opportunity (some would say mandated requirement) to be objective but brutally honest, is no taken as being negative, critical and inappropriate.

So the Solution is another Methodology?

In order to achieve this book's goal, it's come down to codifying my views on Project Management, established and refined over my life as a practicing project manager.

My comments and views are not general: my experience has largely been in a very specific kind of environment, which is described in the next chapter. This environment is probably the most challenging environment to get anything delivered, but it seems also to be a growing one.

It has been suggested to me by various colleagues and friends to present this as a new branded methodology: yet another "silver bullet" solution to the problems facing project management. But that's not my style. This is a very personal exposition of my views and practices. I believe it is applicable by many others, and I believe strongly that it has been the basis of my successes as a Project manager over the years. It may not to be everyone's taste, especially those personally invested in other branded Project Management methodologies or practices.

In making this codification of Project Management principles general it makes it available for everyone, at least the underlying concept: I believe that people of all skill levels can apply this approach, or their version of this approach, to both simplify and make more effective they time that they spend facilitating and driving outcomes for their customers.

Conclusion

Whilst there are many failures and impairments, there are very obviously many successful software project outcomes. Over time, far more successful ones than failures. So how we go about things cannot be all wrong.

Someone (supposedly Einstein, but there's no definite attribution, and even he is quoted as having doubts that he said it) said that insanity is defined by repeating the same process many times, but expecting different results. Instead of continually performing the same processes, we need to consider different approaches to defining and executing projects.

At the end of the day, it's about groups of people making things. This has been a human practice for millennia. The situation is not new. There has to be a better way.

My view: stop doing what doesn't work.