Methodology Patterns

A Different Approach to Create a Methodology for Your Project

Agile India 2014, Bengaluru

Giovanni Asproni
email: gasproni@asprotunity.com
twitter: @gasproni
linkedin: http://www.linkedin.com/in/gasproni

The 15 Seconds Version

- Projects and teams differ, so one methodology won't fit all of them — Alistair Cockburn

- We are solving the wrong problem

http://alistair.cockburn.us/Just-in-time+methodology+construction/v/slim

The Two Minute Version

- There will never be an universal methodology fitting all projects
- Haphazard mix and match of practices is not a solution
- Using patterns and pattern languages to describe and connect practices can help in
  - Finding what works and what doesn't in a given context
  - Creating an appropriate methodology for each project

Agile Is Boring

- This talk is not about becoming “more Agile”
- It is about
- Being more effective
- Being more efficient
- Having fun

“None of the ideas presented here are new; they are just forgotten from time to time.”

Alan J. Perlis, 1966 Turing Award Lecture.

http://amturing.acm.org/award_winners/perlis_0132439.cfm

EPISODES:
A Pattern Language of Competitive Development
Part I

Ward Cunningham, ward@c2.com
Cunningham & Cunningham, Inc.

Submitted to the Second International Conference on
Pattern Languages of Programs
Monticello, Illinois, 6-8 September, 1995

Review Draft of August 6, 1995 (Converted to HTML by the Microsoft Word 6.0 Internet Assistant)

This pattern language describes a form of software development appropriate for an entrepreneurial organization. We assume the entrepreneur to work in a small team of bright and highly motivated people. We also assume time to market is highly valued as it often is where market windows close quickly and development dollars are in short supply. But, unlike some entrepreneurs, we also place high value in being able to get a second version out the door in a timely way; and a third version; and an Nth version; many years down the road. That is, we expect to be successful and have every intention of exploiting that success by continuing development for as long as our customer has desires.

These patterns describe how to develop software. They could be fairly described as *process* patterns though they don't actually describe a process the way a methodology document might. Nor do they describe *designs* or *organizations* a other patterns have. Being patterns, they do describe *things*, things that solve problems that occur in the process. The things can be physical like a document or meeting. Or they can be mental like a commitment or state of mind.

We are particularly interested in the sequence of mental states that lead to important decisions. We call the sequence an *episode*. An episode builds toward a climax where the decision is made. Before the decision, we find facts, share opinions, build concentration and generally prepare for an event that cannot be *known* in advance. After the climax, the decision is known, but the episode continues. In the tail of an episode we act on our decision, promulgate it, follow it through to its consequences. We also leave a trace of the episode behind in its products. It is from this trace that we must often pick up the pieces of thought in some future episode.

We won't be so naive as to suggest that the thoughts leading to a decision be written down. These thoughts are too complex and decisions too numerous for this to be practical. What we do suggest is that hints and pointers be placed in strategic locations so that preparation for subsequent episodes might go more smoothly. Of course they won't. That's because each episode to touch a given area does so with more expectation. We only hope to rise to the occasion. We will know that we have done so if our episodes remain well shaped: not too heavy on the front or the back, and not always getting longer.

There is an old saying that laments, *there is never time to do it right but always time to do it over*. We take this to be a fact of competitive life. We find ourselves unable under competitive

http://c2.com/ppr/episodes.html
Also in: “Pattern Languages of Program Design 2”

Main Page

Methodology Patterns

[edit]

Methodology Patterns are patterns of common Principles, Parameters, and Practices found in Software Development Methodologies.

These patterns are based upon the position that Methodology Design is not only an appropriate approach to building software, but that it's the preferable approach. It is typically better to design your own software development methodology than to "pick one off the shelf." However, that does not mean that existing methodologies (i.e. RUP, Agile, Waterfall, etc.) don't provide value. For a full discussion on this, see Methodology Fundamentals.

The goal of this website is three-fold:

- Advocate and educate on the ideas and concepts behind Methodology Design
- Define a library of Methodology Practices
- Define a library of Methodology Patterns

Opinions, help, and feedback are welcome. At this time, user accounts are locked down. However, if you'd like to submit ideas, feedback, opinions, or otherwise get involved, please email to dshellman at methodologypatterns dot org.

The host and creator of this website is Dan Shellman (User page: User:Dan.shellman).

Recent Additions/Changes

[edit]

- Updated Methodology Fundamentals *(last updated 1 Nov 2007)*
- Added Elaborative Iterations *(added 4 August 2007)*
- Added Variable-Length Iterations *(added 4 August 2007)*
- Updated FAQ *(updated 2 August 2007)*
- Added FAQ *(added 2 June 2007)*
- Added Iteration *(added 26 May 2007)*
- Added Lifecycle Time-Boxed Iterations *(added 26 May 2007)*
- Added Phased Pattern *(added 26 May 2007)*
- Description of Category:Structural Patterns updated *(last updated 26 May 2007)*
- Work in Progress: Why Design a Methodology *(last updated 3 May 2007)*

navigation

- Main Page
- Community portal
- Current events
- Recent changes
- Random page
- Help
- Donations

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

http://www.methodologypatterns.org/wiki/index.php/Main_Page

Methodology Patterns

concept, classification and processes for their application

Ing. Alena Buchalcevová, Ph.D.
Department of Information Technology
University of Economics Prague
Czech Republic
buchalc@vse.cz

CITSA 2004

http://mefis.panrepa.org/CITSA2004PresentationA.pdf

Just-in-time methodology construction

Abstract

Projects and teams differ, so one methodology won't fit all of them. People vary and tend to inconsistency, but they are good at communicating, looking around, and often take initiative to do whatever is needed. Is it possible to build methodologies that are habitable and effective, and accommodate these two statements?

In this paper, I show one can dynamically construct a flexible methodology that draws upon the native strengths of people, giving consideration to their native weaknesses. The mechanisms involved help with communication of both technical issues and project dependency tracking. Team morale and cooperation receive first-order attention. Work products are developed just to the point that intelligent colleagues can find their way forward. Each project's process is tailored over the course of the project to suit that project and team need.

Keywords*

just-in-time methodology, dynamic methodology, flexible process, lightweight methods, software engineering practices, human factors.

Synopsis of the argument

This paper builds on "A methodology per project" [Cockburn99a] and "Characterizing people as first-order, non-linear components of software development" [Cockburn99b] to discuss creating a lightweight, people-centric methodology tuned just for a particular project team, and to do so during the course of the project.

The heart of the argument is this:

- Every project and team is different, but any one organization exhibits common traits, strengths and weaknesses. Through interviews, those can be identified, and used to seed the methodology.
- Principles of methodology construction also seed the methodology: People factors can easily dominate process matters on a project [Cockburn99b], [Weinberg]. Light methodologies based on interpersonal communication are strong ones [Cockburn99a]. The optimal "lightness" varies by project, depending on criteria such as staff size and proximity, and the system's damage potential [Cockburn99a].
- The power of a light methodology centers around using informal, face-to-face communication channels (rather than written documents) to bind the large amount of information flowing within the project. The team draws on the ability of people to "look around, ask, and discuss" while tracking project dependencies, requirements and design issues. This makes the methodology lighter and more economic, but also more "habitable" — pleasant to live in. Experienced team leads frequently use these sorts of mechanisms to run a project to successful completion.

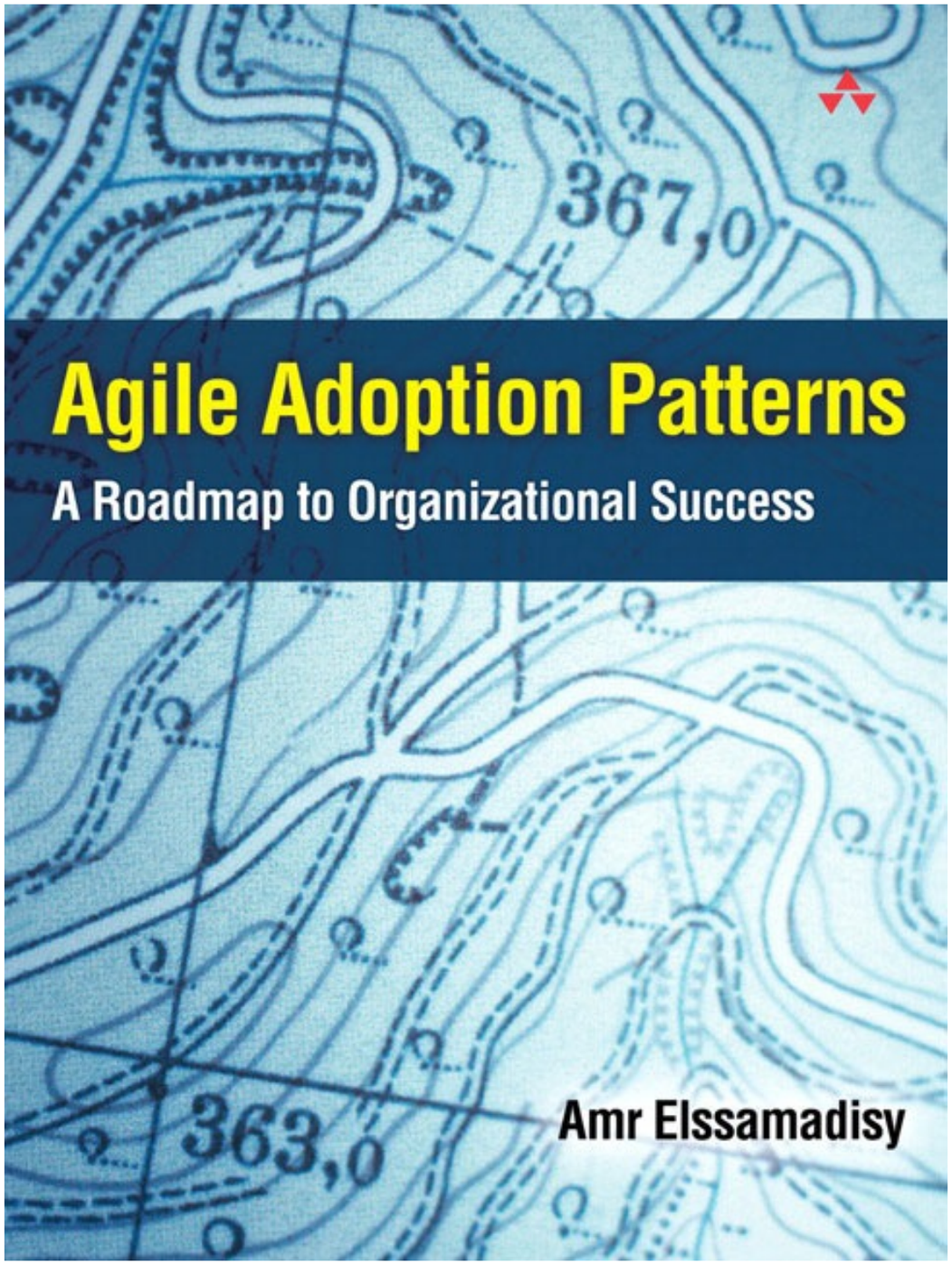
Based on the above, it is even possible to construct a "family" of methodologies that adapt to the local team's size and culture. Such a family is outlined at the end of the paper.

Methodology per project and people's characteristics

The first paper that this one builds upon is "A methodology per project" [Cockburn99a]. It ends with the summary:

A methodology contains ten basic elements: roles, skills, activities, techniques, tools, teams, deliverables, standards, quality measures and project values. The main result of the paper is that there are necessarily multiple methodologies. Different methodologies are needed depending on the "project size" (number of people being coordinated) the "criticality" of the systems being created, and the "priorities" of the project. For any point in the size/criticality space, the designers of the methodology select a "scope" of concerns to address (which project roles, activities, deliverables, and standards to cover) and "optimize" some quality of the project, working from their personal "experiences", including their wishes, fears and base philosophy. Comparison of methodologies should include these dimensions, and their relationship to the needs of the project or organization.

Four principles in methodology design were elaborated, which can be summarized in short form as:


Agile Adoption Patterns
A Roadmap to Organizational Success
Amr Elssamadisy

Organizational Patterns of Agile Software Development

James O. Coplien · Neil B. Harrison

The Problems

The mainstream debate focus is around finding “The Methodology”

The Never-ending Software Methodology Debate

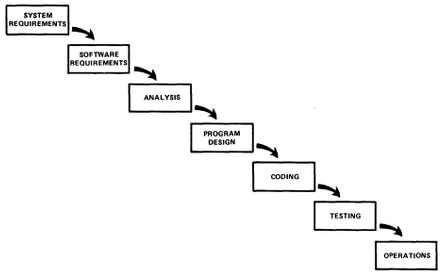
1. Start: Waterfall is the mainstream methodology

2. The mainstream methodology shows some limitations

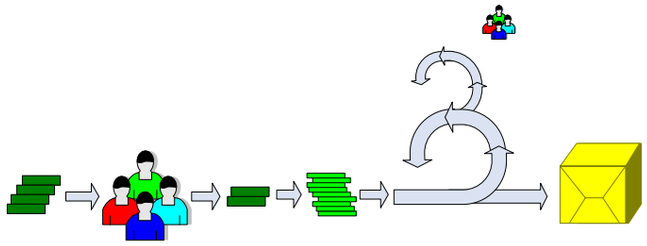
3. Somebody proposes a new, or lesser-known, or almost forgotten methodology and shows how it addresses the limitations at step 2 (and how it's better than Waterfall, of course)

4. After a long argument, heavily based on personal opinions, and often light on facts, the methodology at step 3 becomes the mainstream one

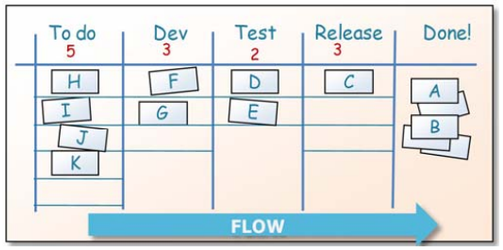
5. Go to step 2



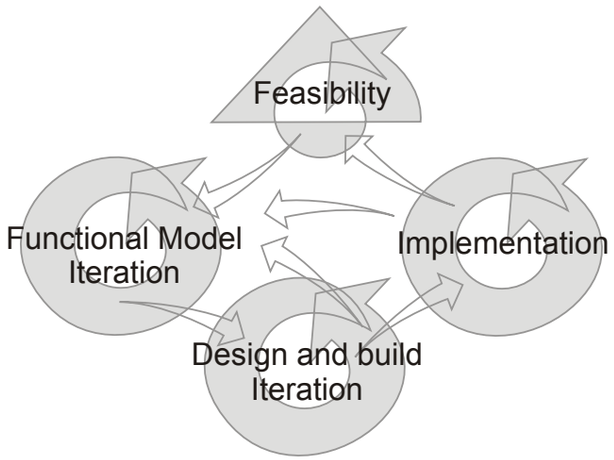
Waterfall



Scrum

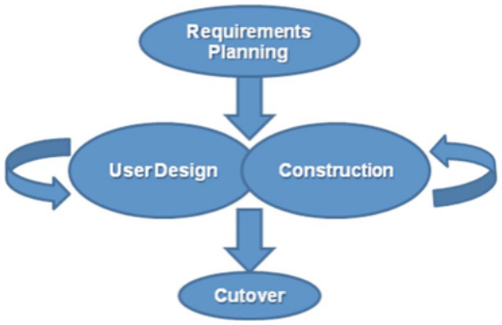


Kanban

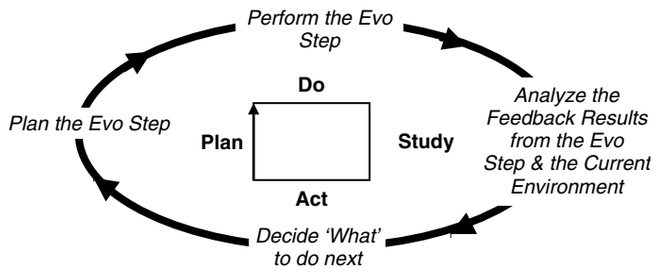


DSDM

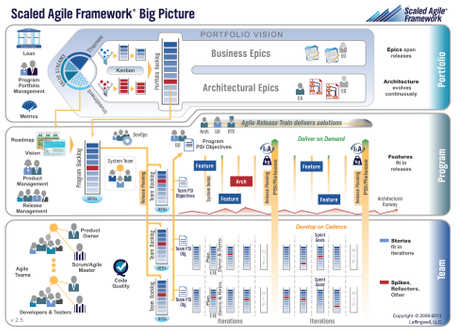
FDD



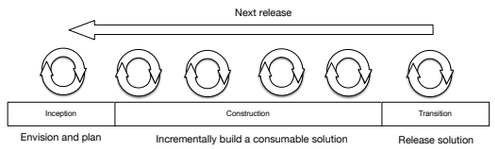
RAD



Evo



SAFe



DAD

eXtreme Programming

The focus is too much on the methods and not enough on the products and the contexts

Comparing the effectiveness of two off-the-shelf methodologies in a given context is impractical

Implementation details matter

a lot

Methodology

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.

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

Methodology (Another Definition)

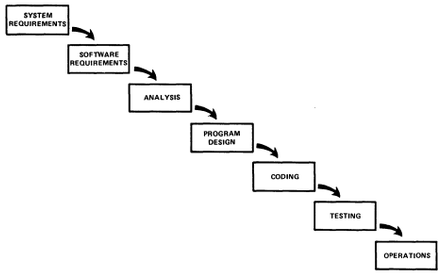
“Your ”methodology“ is everything you regularly do to get your software out. It includes who you hire, what you hire them for, how they work together, what they produce, and how they share. It is the combined job descriptions, procedures, and conventions of everyone on your team. It is the product of your particular ecosystem and is therefore a unique construction of your organization.”

Alistair Cockburn

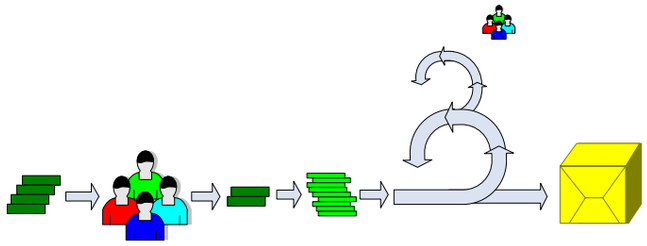
What Is A Methodology For?

- Introducing new people to the process
- Substituting people
- Delineating responsibilities
- Impressing the sponsors
- Demonstrating visible progress
- Curriculum for education

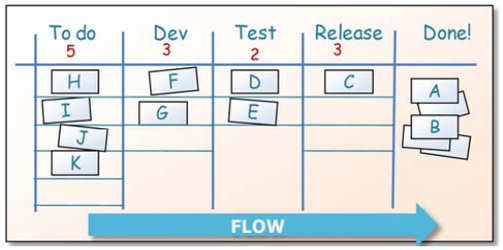
From: “Agile Software Development”, Alistair Cockburn



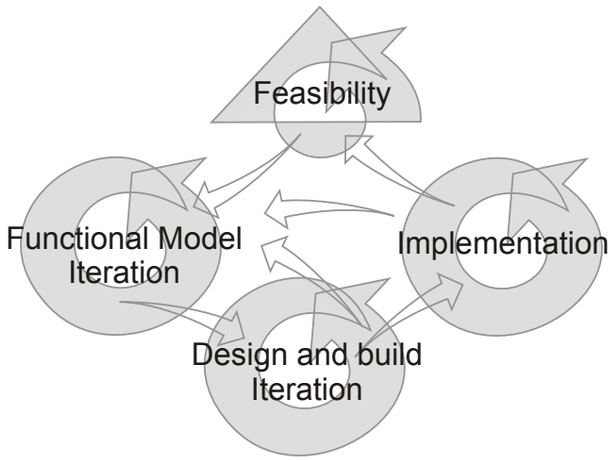
Waterfall



Scrum

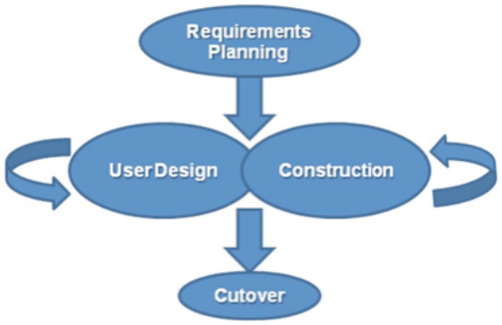


Kanban

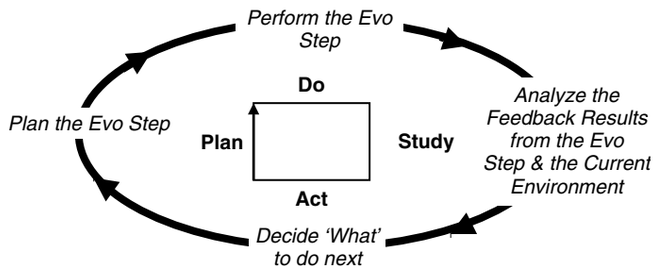


DSDM

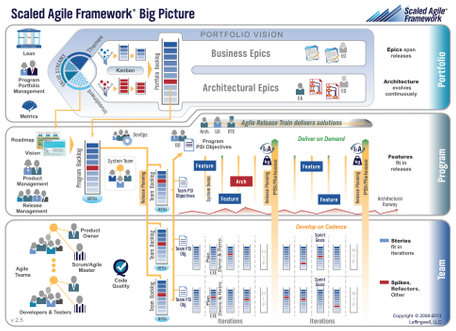
FDD



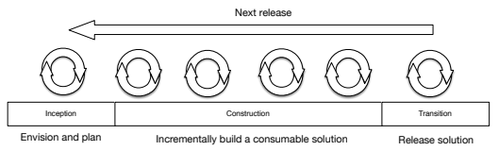
RAD



Evo



SAFe



DAD

eXtreme Programming

Context

Company culture

Team culture

Skills

Customer culture

Product

Preferences

Budget

Time

Methodology and context are strongly connected

The Problems Revisited

- The mainstream debate focus is around finding “The Methodology”
  - The focus is too much on the methods and not enough on the products and the contexts
  - Measuring the effectiveness of an off-the-shelf methodology in a given context is impractical
  - The implementation details matter **a lot**

A Revolutionary Idea

One methodology per project

…Not Really Revolutionary…

The idea *[of one methodology per project]* has been re-derived a number of times. Capers Jones in his "Benchmarking" book made a nice argument for needing 65,000 methodologies (back then, even). **When I was doing my Dr. thesis, I found I couldn't use the methodology-per-project as a new result, since there was ample previous research concluding the same**

…

Working out that different projects need different processes/ methodologies is the easy part. **Recognizing that EVERY project needs a different methodology is a hard pill to swallow, but a number of people have seen fit to swallow it.** Those are not yet the hard or even critical parts…

The ultimate question is, **What To Do About It? HOW does one create a new methodology on every project without hosing the project budget?**

The thing the doctoral defense committee found most novel unusual surprising about my thesis was **the idea of tailoring and tuning a methodology in stages / during/ a project via the reflection workshops.**

http://alistair.cockburn.us/Just-in-time+methodology+construction/v/slim

Like All Frameworks…

Methodology frameworks are **customizable up to a point**, e.g., the Scrum Guide says:

> **"Scrum's roles, artifacts, events, and rules are immutable** and although implementing only parts of Scrum is possible, the result is not Scrum. **Scrum exists only in its entirety** and functions well as a container for other techniques, methodologies, and practices."

Methodologies Can Share Something

Different methodologies mostly
mix and match the same tools
and practices differently

And many teams do that haphazardly

Measuring the effectiveness of a practice in a given context is doable

Some Studies

- "A Comparison of Pair Programming to Inspections for Software Defect Reduction" - http://cs.adelaide.edu.au/users/ljiang/research/ResearchInComputerScienceEducation/7276046.pdf

- Test-Driven Development as a Defect-Reduction Practice - http://collaboration.csc.ncsu.edu/laurie/Papers/williamsltestDrivenDevelopment.pdf

- A study to investigate the impact of requirements instability on software defects - http://dl.acm.org/citation.cfm?id=986727

- Etc.

Which Practices?

- Practices can be
  - Technical
  - Management
  - Leadership
  - …And any other relevant classification
  - Some practices belong to several categories

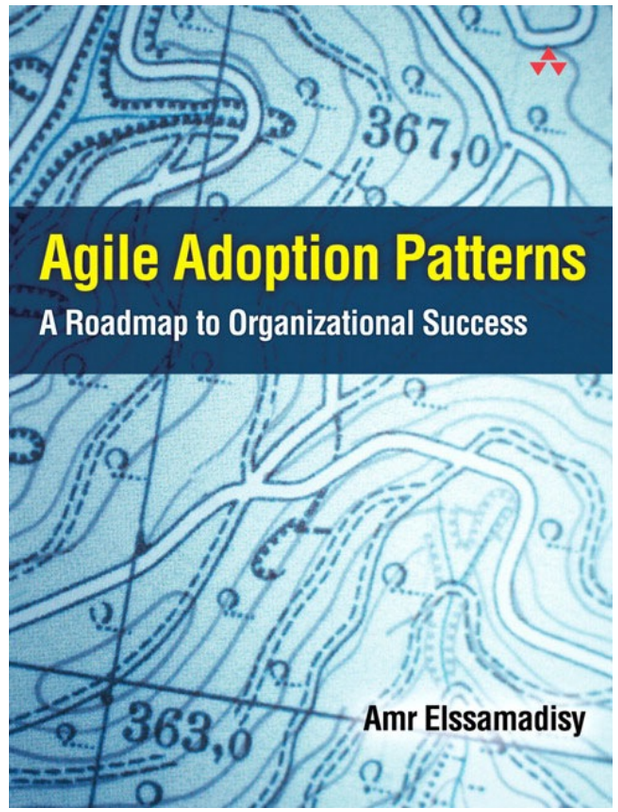
Patterns and pattern languages provide guidance

Parts Of A Pattern

Pattern Name and Classification: A descriptive and unique name that helps in identifying and referring to the pattern.
Intent: A description of the goal behind the pattern and the reason for using it.
Also Known As: Other names for the pattern.
Motivation (Forces): A scenario consisting of a problem and a context in which this pattern can be used.
Applicability: Situations in which this pattern is usable; the context for the pattern.
Consequences: A description of the results, side effects, and trade offs caused by using the pattern.
Known Uses: Examples of real usages of the pattern.
Related Patterns: Other patterns that have some relationship with the pattern; discussion of the differences between the pattern and similar patterns.

Adapted from: http://en.wikipedia.org/wiki/Software_design_pattern

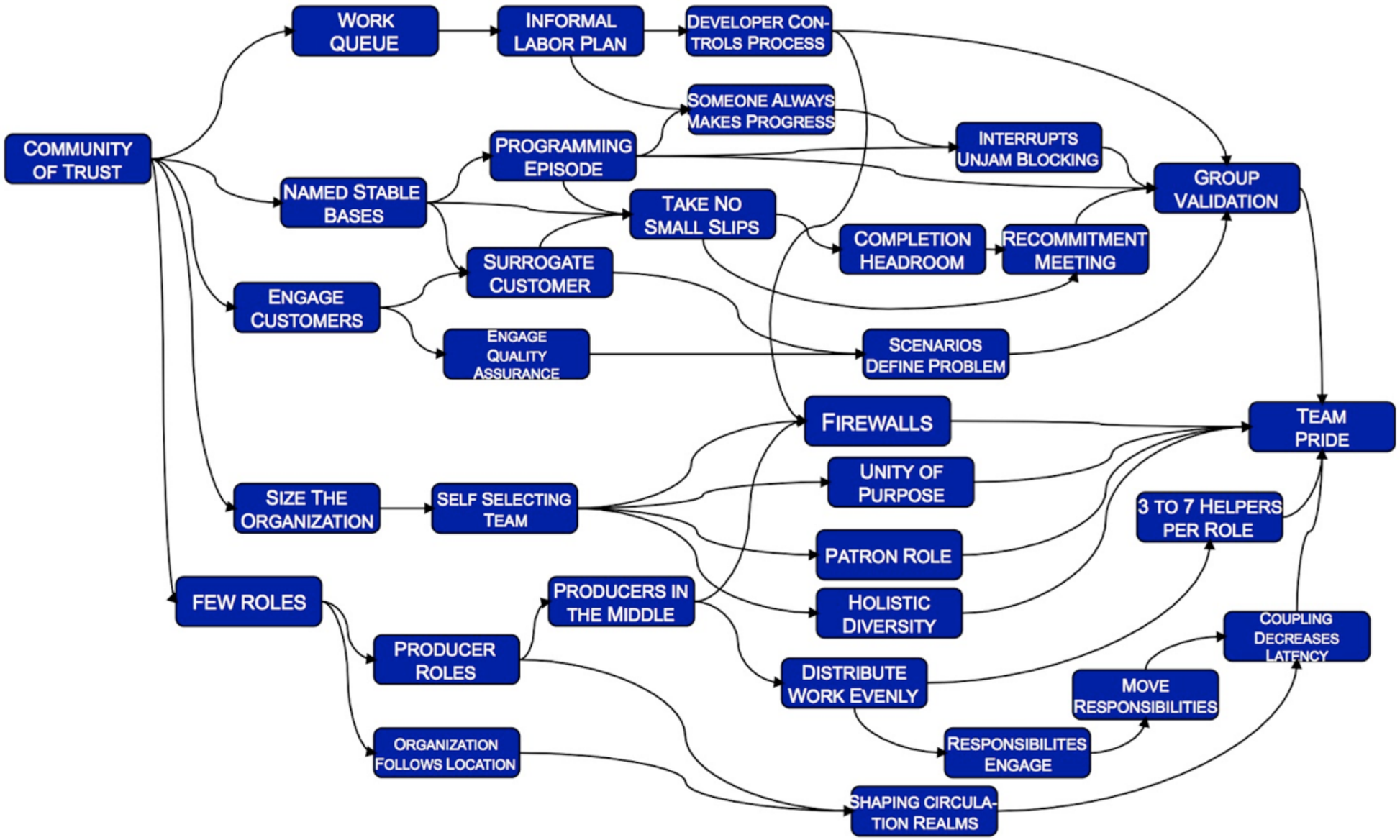
Example: Information Radiator



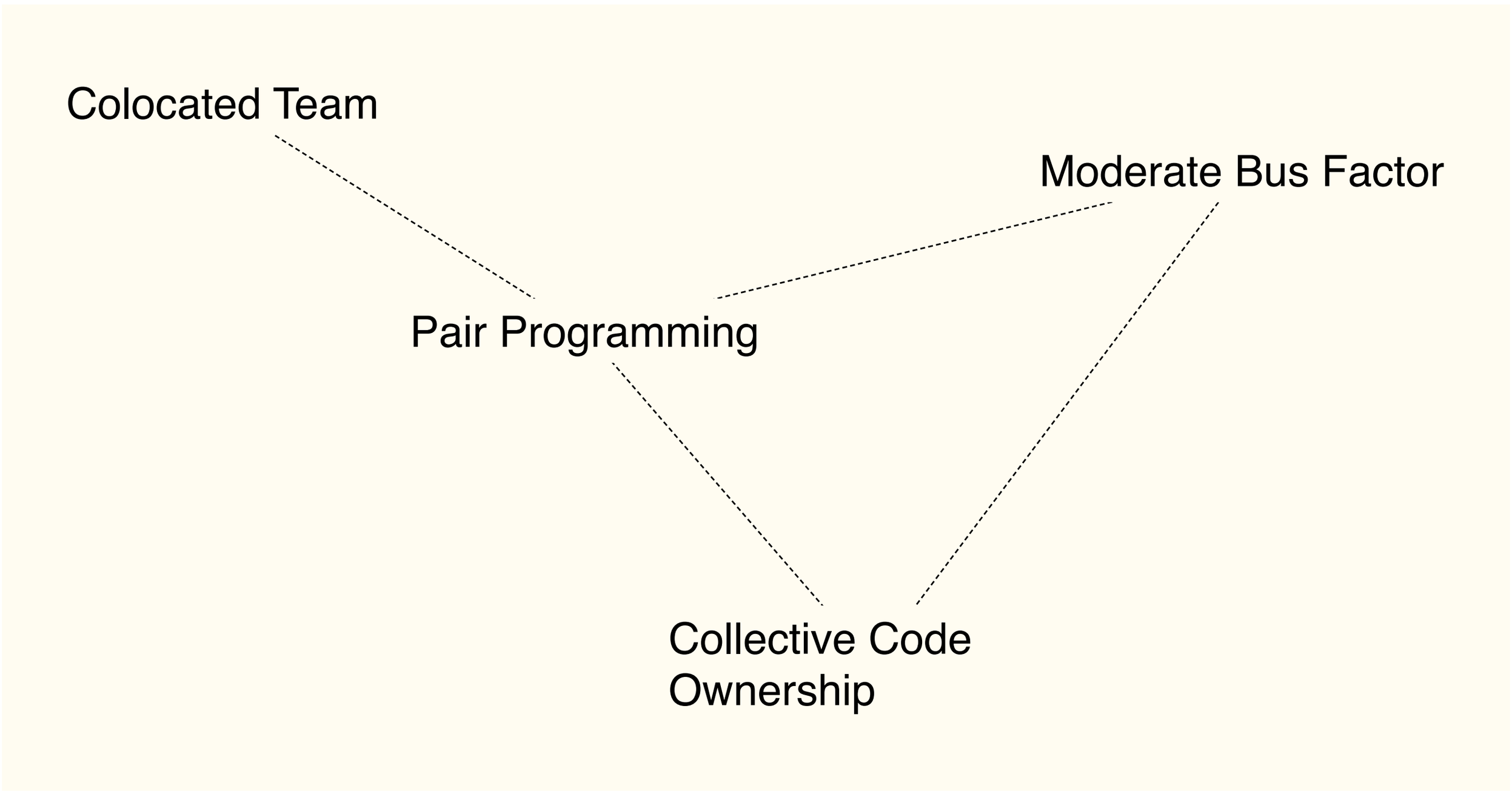
Before You Ask…

- Patterns are not best practices
- In fact, there are no best practices
  - Only useful or useless ones. Depending on the context

Pattern Languages



From: http://scrum.jeffsutherland.com/2008/02/scrum-and-organizational-patterns.html


Colocated Team
Moderate Bus Factor
Pair Programming
Collective Code
Ownership

Patterns Are Useful Because…

- Describe not only the positive effects, but also the contexts in which are known to work best, and the consequences of using them

- The contexts and the consequences are particularly important

  - They determine the chances of success or failure of a practice in a particular project

  - Are very rarely discussed by the proponents of a particular methodology framework

Pattern Languages Are Useful Because…

- Give an indication of which patterns work well or not so well with each other

- Knowing that a particular pattern belongs to several pattern languages (e.g., technical and managerial) helps in understanding better all the implications of using it

- Pre-packaged methodology frameworks tend to present practices as belonging to only one category

- Leading people to ignore several other implications

Some Important Consequences

- Free the mind from thinking in terms of Scrum, Kanban, Waterfall, etc.

- Start to think of what is really important for the project without fear of doing something bad because we are not following the methodology by the book

- There is no sacred book anymore

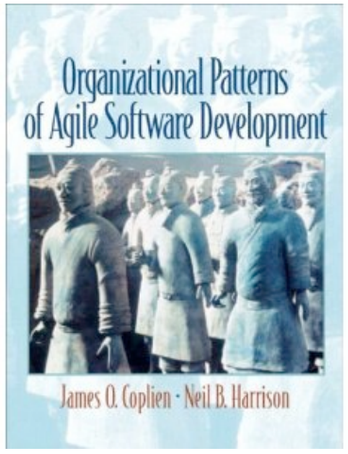
- No need to be ashamed anymore for doing Scrum but not quite

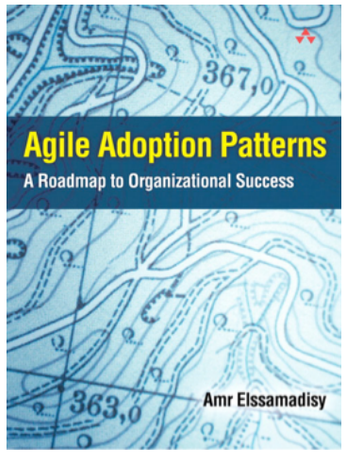
A Methodology Is A Living Thing

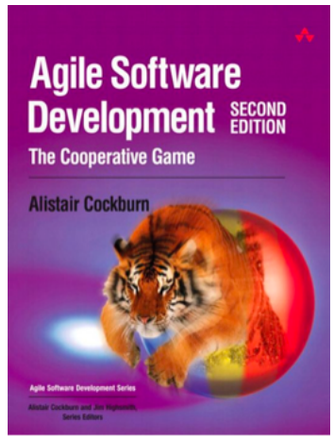
It has to change along with the surrounding context

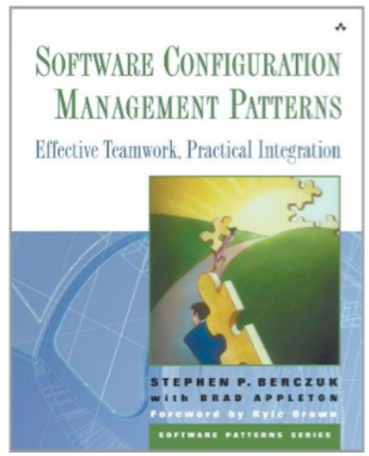
Patterns and pattern languages change over time

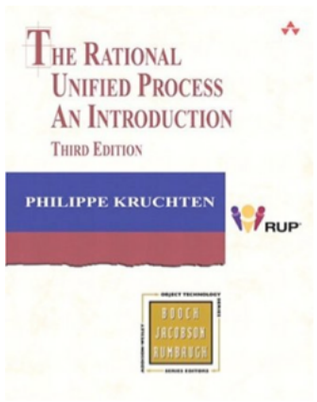
Finding the Patterns

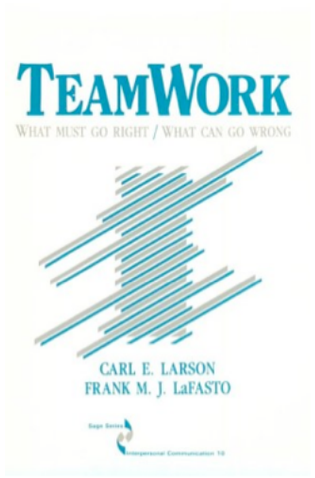


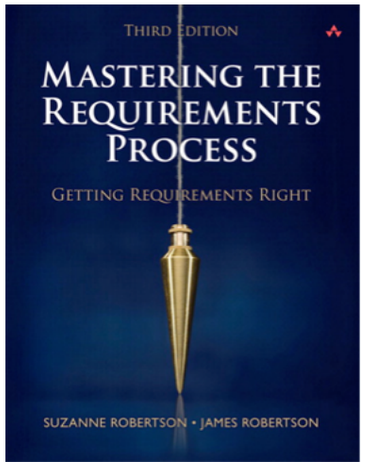












Using The Patterns

Key to the success of any performance improvement initiative is understanding what to improve and why; yet it's both astonishing and disappointing how many improvement initiatives are launched without anyone asking and answering these two most primal questions

From “Return On Process”, Michael West

Using The Patterns

- Set the main goals
- Understand the context
- Choose the patterns that best match goals and context
- Inspect and adapt

Keep In Mind

- The real context may be different from what you see
  - Defined vs Performed process
- The methodology has to work for the people using it

Don’t Over-constrain The System

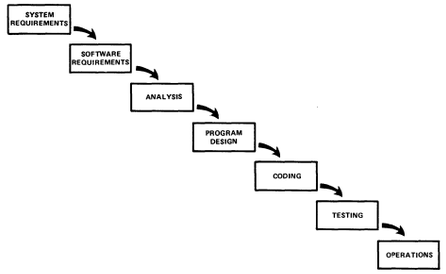
- Choose only the practices that match your main goals
- Leave secondary things alone
  - Let people have some initiative
  - You cannot predict everything

Work To Rule

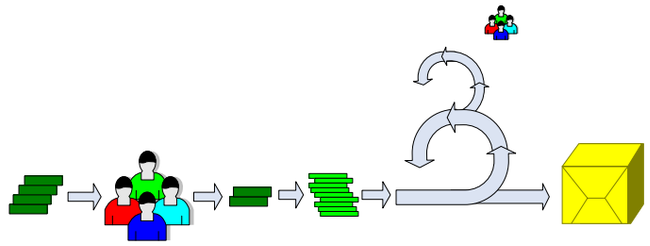
Work-to-rule is an industrial action in which employees do no more than the minimum required by the rules of their contract, and follow safety or other regulations precisely in order to cause a slowdown, rather than to serve their purposes.

http://en.wikipedia.org/wiki/Work-to-rule

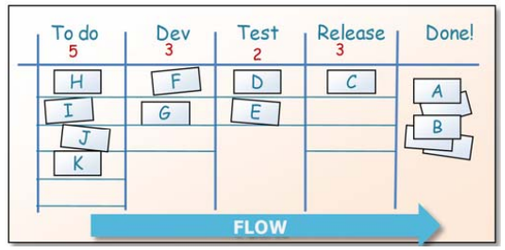
What About These?



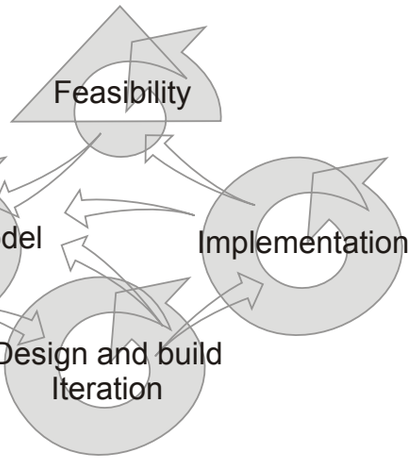
Waterfall



Scrum

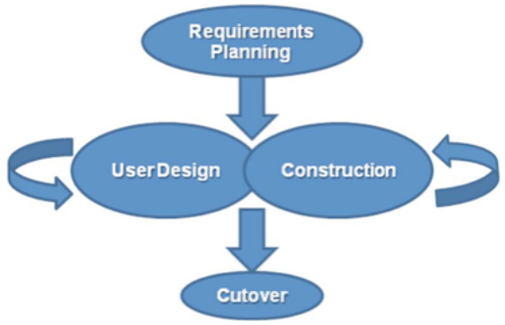


Kanban

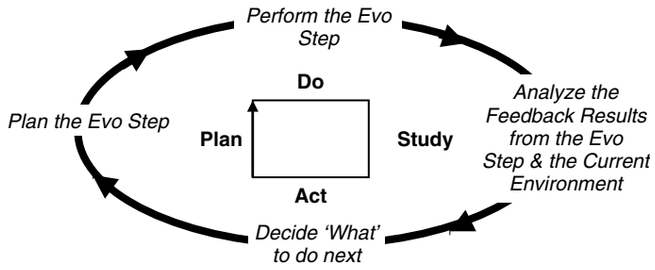


DSDM

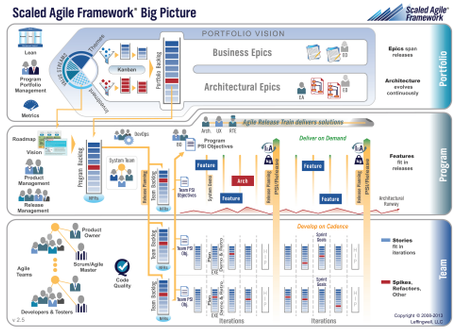
FDD



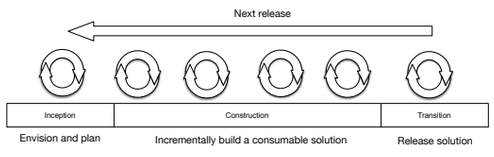
RAD



Evo



SAFe



DAD

eXtreme Programming

What About Scaling up?

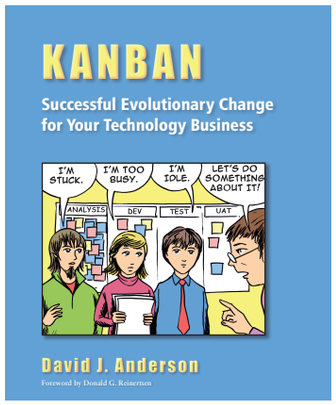
Final Rant

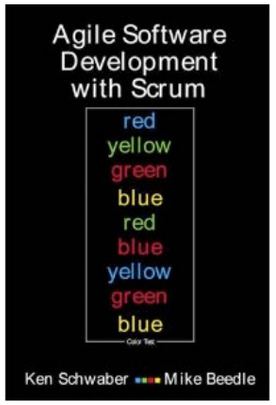
I think we need to fundamentally change the conversation about methodologies--which, in my opinion is too focused in the unachievable goal of finding the one that will work for every context.

Links

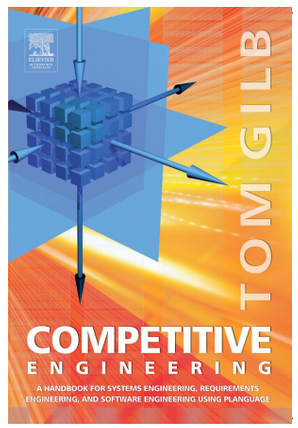
- http://alistair.cockburn.us/Just-in-time+methodology+construction/v/slim
- http://alistair.cockburn.us/Methodology+per+project/v/slim
- http://www.agilemanifesto.org
- http://www.agilealliance.org
- http://www.dsdm.org
- http://www.gilb.com/Project-Management
- http://limitedwipsociety.ning.com
- https://www.scrum.org/ScrumGuides.aspx
- http://www.scrumalliance.org
- http://asprotunity.com/papers.html

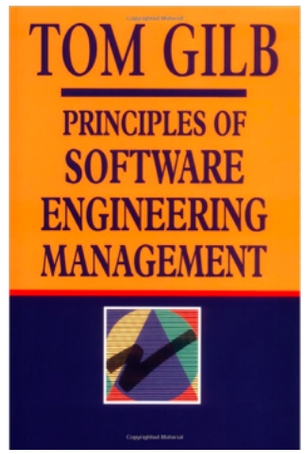
Some Books

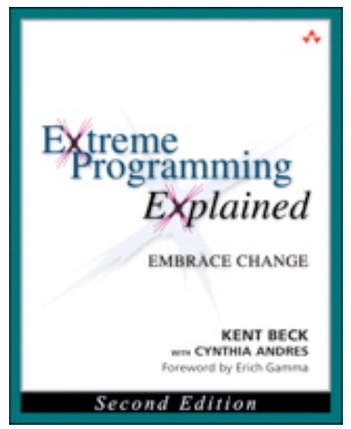

KANBAN
Successful Evolutionary Change for Your Technology Business
I'M STUCK.
I'M TOO BUSY.
I'M IDLE.
LET'S DO SOMETHING ABOUT IT!
ANALYSES
DEV
TEST
UAT
David J. Anderson
Foreword by Donald G. Reinertsen

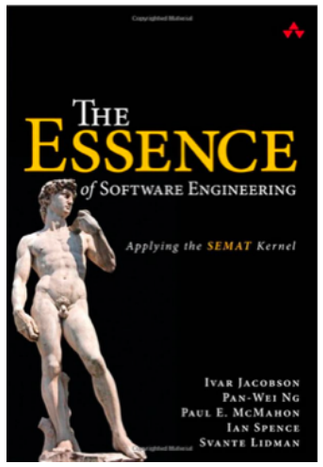

Agile Software Development with Scrum
red
yellow
green
blue
red
blue
yellow
green
blue
Ken Schwaber
Mike Beedle


FEARLESS CHANGE
PATTERNS FOR INTRODUCING NEW IDEAS
MARY LYNN MANNS, Ph.D.
From the Library of Giovanni Asproni

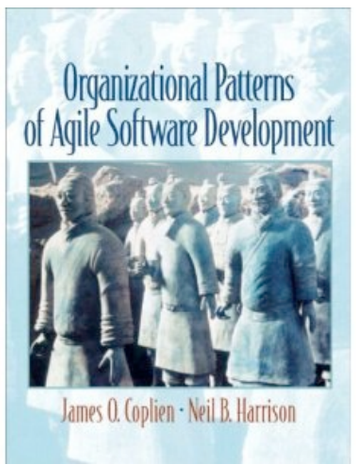

TOM GILB
COMPETITIVE ENGINEERING
A HANDBOOK FOR SYSTEMS ENGINEERING, REQUIREMENTS ENGINEERING, AND SOFTWARE ENGINEERING USING PLANGUAGE

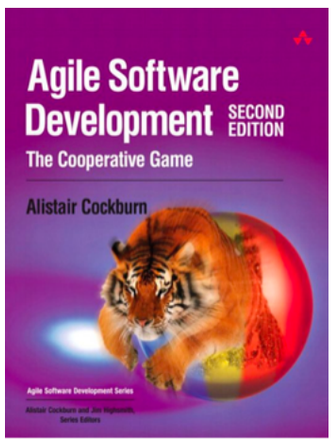

TOM GILB
PRINCIPLES OF SOFTWARE ENGINEERING MANAGEMENT

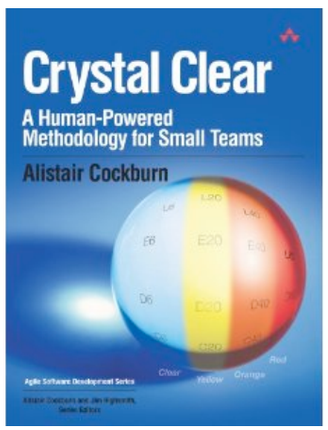

Extreme Programming Explained
EMBRACE CHANGE
KENT BECK
with CYNTHIA ANDRES
Foreword by Erich Gamma
Second Edition

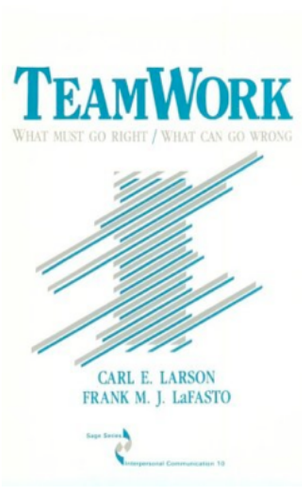

THE ESSENCE of SOFTWARE ENGINEERING
Applying the SEMAT Kernel
IVAR JACOBSON
PAN-WEI NG
PAUL E. McMAHON
IAN SPENCE
SVANTE LIDMAN

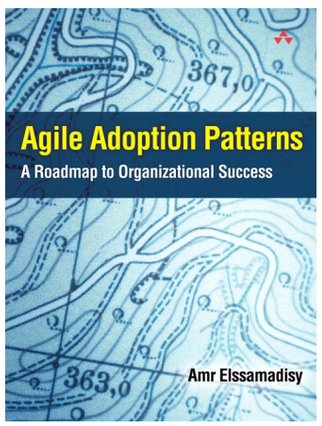

Disciplined Agile Delivery
A Practitioner's Guide to Agile Software Delivery in the Enterprise
Scott W. Ambler • Mark Lines
Foreword by Dave West

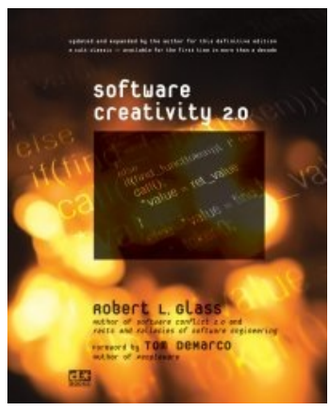

Organizational Patterns of Agile Software Development
James O. Coplien · Neil B. Harrison


Agile Software Development SECOND EDITION
The Cooperative Game
Alistair Cockburn
Agile Software Development Series

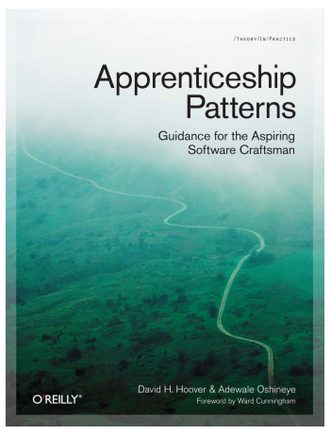

Crystal Clear
A Human-Powered Methodology for Small Teams
Alistair Cockburn
Agile Software Development Series


TEAMWORK
WHAT MUST GO RIGHT / WHAT CAN GO WRONG
CARL E. LARSON
FRANK M. J. LaFASTO


Agile Adoption Patterns
A Roadmap to Organizational Success
Amr Elssamadisy


software creativity 2.0
Robert L. Glass
Foreword by TOM DEMARCO


RETURN ON PROCE$S (ROP)
Getting Real Performance Results from Process Improvement
Michael West
CRC Press


Apprenticeship Patterns
Guidance for the Aspiring Software Craftsman
O'REILLY
David H. Hoover & Adewale Oshineye
Foreword by Ward Cunningham

Credits

- Images from
  - http://www.istockphoto.com
  - http://www.iconarchive.com/show/oxygen-icons-by-oxygen-icons.org/Categories-preferences-system-icon.html
  - https://commons.wikimedia.org/wiki/File:DSDM_Development_Process.svg
  - http://en.wikipedia.org/wiki/File:Question_mark.png
  - http://www.devzombie.com/pages/xp-extreme-programming
  - http://www.openscience.org/blog/?p=287
  - http://www.infoq.com/minibooks/kanban-scrum-minibook
  - http://en.wikipedia.org/wiki/File:Fdd_process_diagram.png