IBM Rational Software Development Conference 2008

WHERE TEAMS ARE R-HEROES

IBM

Rational. software

Back to Basics:
Getting Good Software Quickly and at Low Cost

Ivar Jacobson
Chairman, Ivar Jacobson International
ivar@ivarjacobson.com

IVAR JACOBSON INTERNATIONAL

Agenda

1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

Our goal

Good Software, Quickly and at Low Cost!

What it takes

Quickly

Competent & Motivated People

Low Cost

Large Scale Reuse of Components

Good Software

Useful Extensible Reliable

What it takes

It is as easy

as that!

☺

What it takes

Quickly

You just need a good process

☺

Useful Extensible Reliable

Problem with Process (Methodology, Method…)

- Every process tries to be complete
  - As a consequence every successful process will grow until it dies under its own weight
- Every branded process is just a soup of ideas ”borrowed” from other processes
  - With some new idea(s)
- Every process usually becomes just shelf-ware
  - Law of Nature: People don’t read process descriptions
- The process is out of sync with what the team does…
  - …and the project – process gap get wider and wider
- The project has to adopt an entire process
  - No-one uses an entire process or limits themselves to practices from one process

Problem with Process (Methodology, Method…)

-
-
-
-

It’s no wonder

no-one likes process ☹

from one process

Problem with Process (Methodology, Method…)

There are practices to help you

☺

Agenda

1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

Enough Process – Let's Do Practices

In the future, … n ever pr… t but invis… proce…

Process … sec…

We n… new paradigm

…en

…of process

From the successes in modern software development

The … e Eng… g C…

…ess Mat… Camp

Agile Methods Camp

Examples: Unified …ocess CMMI, Spice XP, Scrum

A new paradigm

Enough Process – Let's Do Practices

In the future, an ever present but invisible process

Process becomes second nature

The team's way-of-working is just a composition of Practices

We need a new paradigm

Practice is a First Class Citizen

the unit of adoption, planning and execution of process

From the successes in modern software development

	The Software Engineering Camp	Process Maturity Camp	Agile Methods Camp
Examples:	Unified Process	CMMI, Spice	XP, Scrum

History of Practices

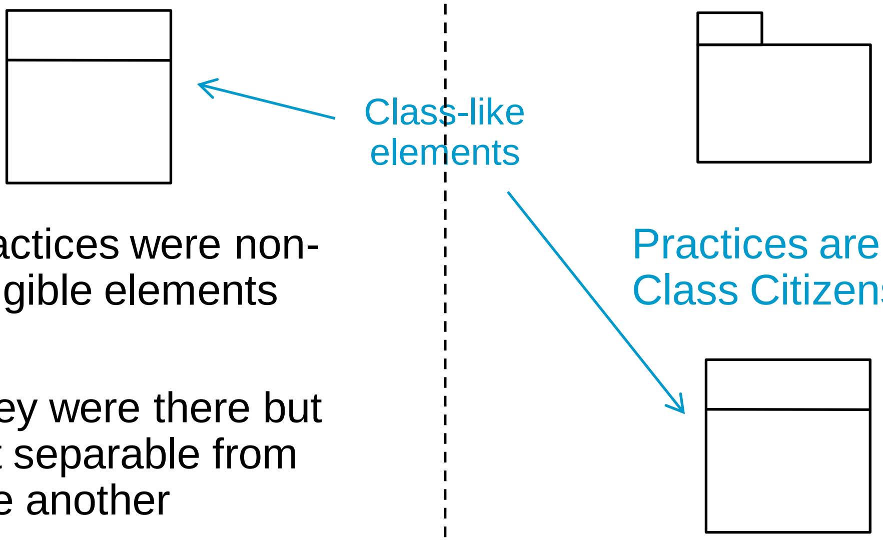
1950s -	Software developers have always talked about ‘best’ practices
Late 1990s	Processes presented as collections of best practices, but practices were not separable from one another
2003 – Aug	Practices as Aspects or First Class Citizens presented by Ivar at XP conference in New Orleans
2004 – June	• Practices formalized as ‘use cases for processes’ • Practices popularized and made more practical through usage of cards, game boards, etc. • Problem with how to produce loosely coupled practices frameworks solved.
2007	Practice composition and execution in EssWork
2008	Practices adopted as first class citizens by IBM Rational.

This is smart!

The Paradigm Shift

- We have always had practices in a loose meaning

	Before	**Now**
Process	Process is First Class Citizen	Process is just a package of practices
Practice	Practices were non-tangible elements They were there but not separable from one another	Practices are First Class Citizens



- After the paradigm shift you can do all kinds of operations on practices
  - Separate them, compose them, teach them, execute them

Agenda

1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

We need a shared definition of “practice”

Pragmatics

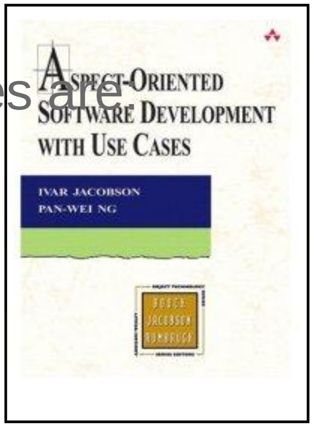
- A practice provides a way to **systematically** address a particular aspect of a process.
- There are three kinds of practices (at the least):
  - Peer practices
    - A practice has a clear beginning and an end

    allowing it to be separately applied, examples are:
    - Iterative development
    - Use case driven development
    - Project management à la Scrum
  - Extension practices
    - Use cases for SOA
  - Cross-cutting practices
    - Team practice incl workshops, war room,

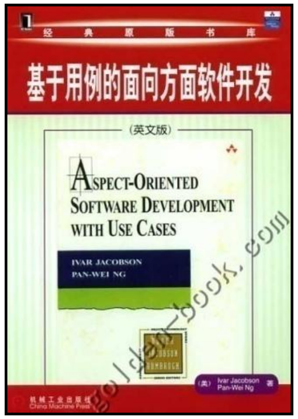
      pair programming, etc.

More precisely

- A use-case module in our AOSD book


Aspect-Oriented Software Development with Use Cases

Ivar Jacobson
Pan-Wei Ng


基于用例的面向方面软件开发

(英文版)

Aspect-Oriented Software Development with Use Cases

Ivar Jacobson
Pan-Wei Ng

ユースケースによるアスペクト指向ソフトウェア開発

There are 100’s of so-called practices…

Business Modeling

Test-Driven Development

Scrum

Product-Line Engineering

Risk-Driven Iterative Development

Systems Engineering

Aspect Orientation

Robustness Analysis

Retro-spectives

Business Process Re-Engineering

Use-Case Driven Development

Pair Programming

PSP

User Stories

SOA

Prince2

Use-Case Modeling

Program Management

…but are really all the same kind of thing?

There are 100's of so-called practices…

✗ Business Modeling	✓ Test-Driven Development	✓ Scrum	Product-Line Engineering	✓ Risk-Driven Iterative Development	✓ Systems Engineering
✗ Aspect Orientation	? Robustness Analysis	? Retro-spectives	Business Process Re-Engineering	✓ Use-Case Driven Development	✗ Pair Programming
✓ PSP	✓ User Stories	SOA	Prince2	✗ Use-Case Modeling	Program Management

…but are really all the same kind of thing?

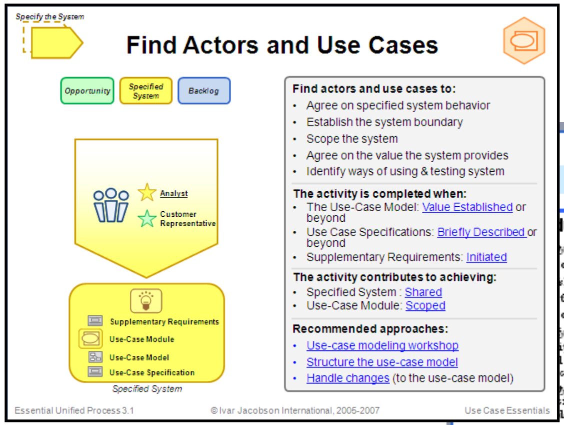
Focus on the Essentials

What is Essential?

- It is the key things to do and the key things to produce
- It is about what is important about these things
- It is less than a few percent of what experts know about these things
  - Law of nature: People don’t read process books
- It is the placeholders for conversations
  - Law of nature: People figure out the rest themselves
  - Training helps
- It is the base for extensions

Starting with the essentials makes the practice easy to learn and adopt.

How much do you need in your hands?



Specify the System

Find Actors and Use Cases

Opportunity
Specified System
Backlog
Analyst
Customer Representative
Supplementary Requirements
Use-Case Module
Use-Case Model
Use-Case Specification
Specified System

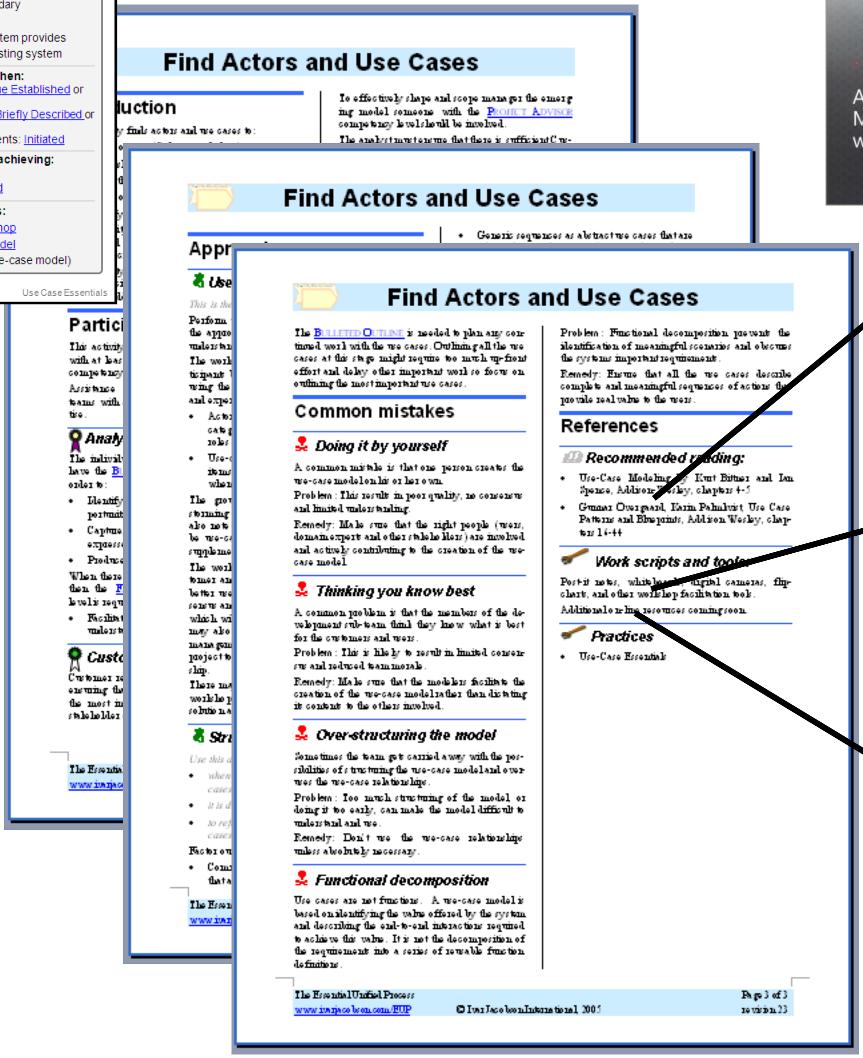
Find actors and use cases to:
- Agree on specified system behavior
- Establish the system boundary
- Scope the system
- Agree on the value the system provides
- Identify ways of using & testing system

The activity is completed when:
- The Use-Case Model: Value Established or beyond
- Use Case Specifications: Briefly Described or beyond
- Supplementary Requirements: Initiated

The activity contributes to achieving:
- Specified System : Shared
- Use-Case Module: Scoped

Recommended approaches:
- Use-case modeling workshop
- Structure the use-case model
- Handle changes (to the use-case model)

Essential Unified Process 3.1 ©Ivar Jacobson International, 2005-2007 Use Case Essentials



Find Actors and Use Cases

The BULLETED OUTLINE is needed to plan any continued work with the use cases. Outlining all the use cases at this stage might require too much up-front effort and delay other important work so focus on outlining the most important use cases.

Common mistakes

Doing it by yourself

A common mistake is that one person creates the use-case model on his or her own.

Problem : This result in poor quality, no consensus and limited understanding.

Remedy: Make sure that the right people (users, domain expert and other stakeholders) are involved and actively contributing to the creation of the use-case model.

Thinking you know best

A common problem is that the members of the development sub-team think they know what is best for the customers and users.

Problem : This is likely to result in limited consensus and reduced team morale.

Remedy: Make sure that the modelers facilitate the creation of the use-case model rather than dictating its contents to the others involved.

Over-structuring the model

Sometimes the team get carried away with the possibilities of structuring the use-case model and overuses the use-case relationships.

Problem : Too much structuring of the model or doing it too early, can make the model difficult to understand and use.

Remedy: Don't use the use-case relationships unless absolutely necessary.

Functional decomposition

Use cases are not functions. A use-case model is based on identifying the value offered by the system and describing the end-to-end interactions required to achieve this value. It is not the decomposition of the requirements into a series of reusable function definitions.

Problem : Functional decomposition prevents the identification of meaningful scenarios and obscures the systems important requirements.

Remedy: Ensure that all the use cases describe complete and meaningful sequences of actions that provide real value to the users.

References

Recommended reading:

- Use-Case Modeling by Kurt Bittner and Ian Spence, Addison-Wesley, chapters 4-5
- Gunnar Overgaard, Karin Palmkvist, Use Case Patterns and Blueprints, Addison Wesley, chapters 14-44

Work scripts and tools:

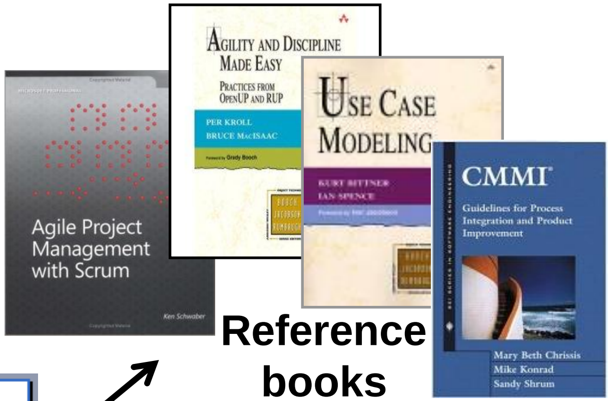
Post-it notes, whiteboards, digital cameras, flip-charts, and other workshop facilitation tools.

Additional on-line resources coming soon.

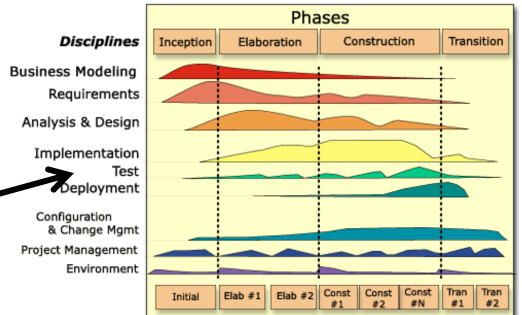
Practices

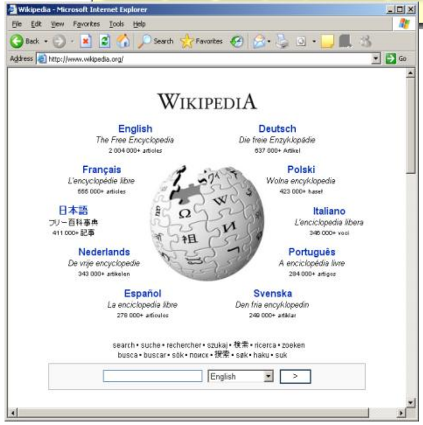
- Use-Case Essentials

The Essential Unified Process
www.ivarjacobson.com/EUP ©Ivar Jacobson International 2005 Page 3 of 3 revision 23

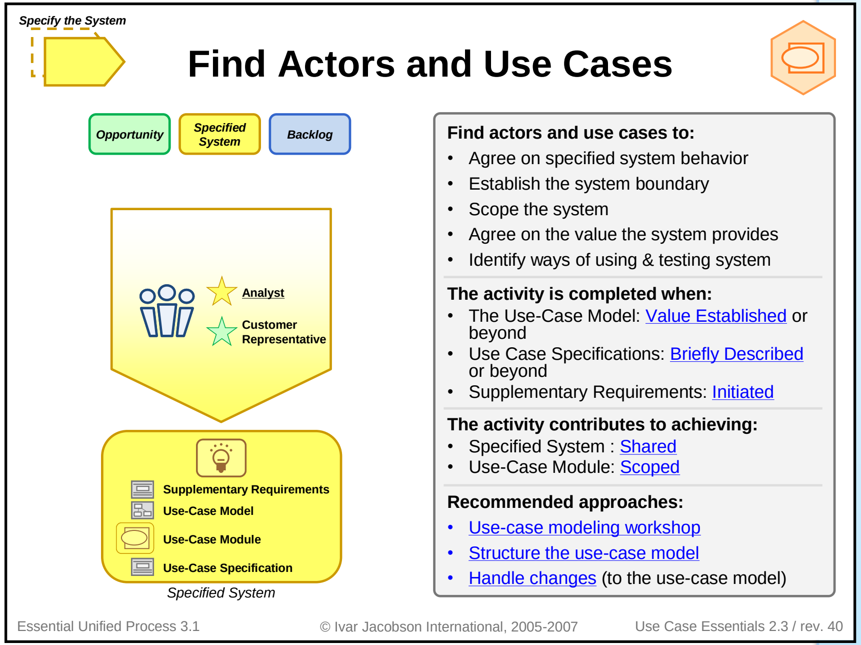

Agile Project Management with Scrum
Ken Schwaber
Agility and Discipline Made Easy
Practices from OpenUP and RUP
Per Kroll
Bruce MacIsaac
Use Case Modeling
Kurt Bittner
Ian Spence
CMMI
Guidelines for Process Integration and Product Improvement
Mary Beth Chrissis
Mike Konrad
Sandy Shrum

Reference books





Why Cards?


Specify the System

Find Actors and Use Cases

Opportunity
Specified System
Backlog
Analyst
Customer Representative
Supplementary Requirements
Use-Case Model
Use-Case Module
Use-Case Specification
Specified System

Find actors and use cases to:
- Agree on specified system behavior
- Establish the system boundary
- Scope the system
- Agree on the value the system provides
- Identify ways of using & testing system

The activity is completed when:
- The Use-Case Model: Value Established or beyond
- Use Case Specifications: Briefly Described or beyond
- Supplementary Requirements: Initiated

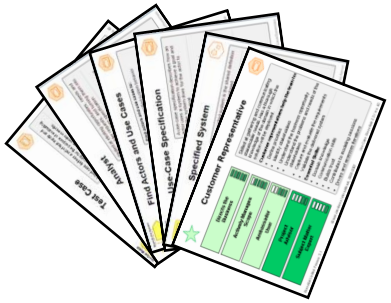
The activity contributes to achieving:
- Specified System : Shared
- Use-Case Module: Scoped

Recommended approaches:
- Use-case modeling workshop
- Structure the use-case model
- Handle changes (to the use-case model)

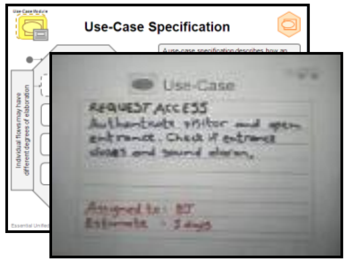
Essential Unified Process 3.1 © Ivar Jacobson International, 2005-2007 Use Case Essentials 2.3 / rev. 40

- Cards are tactile
- Cards are simple and visual
- Cards use conversational and personalized style
- Cards are not prescriptive so they get the learner to think more deeply
- Cards get…and keep…the readers attention
- Cards promote agility
  - They can be written on to make minor adjustments to the practice on the fly

- A practice is a set of cards



- A team works on a set of instance cards


Use-Case Specification
Use-Case
REQUEST ACCESS
Authenticate visitor and open entrance. Check if entrance closes and sound alarm.
Assigned to: BT
Estimate: 3 days

A Good Practice is good for the team

- Gives a result of observable value to the customer of the team
  - It is a building block for the team – not necessarily for the process engineers.
    - Not too big – not too small
  - Thus, it includes its own verification
  - Solves a problem rather than presents a technique (for that we have patterns)
  - Provides practical advice
- Starts from the essentials
  - Can be easily adapted and extended to meet your needs
  - Complements the industry body of knowledge

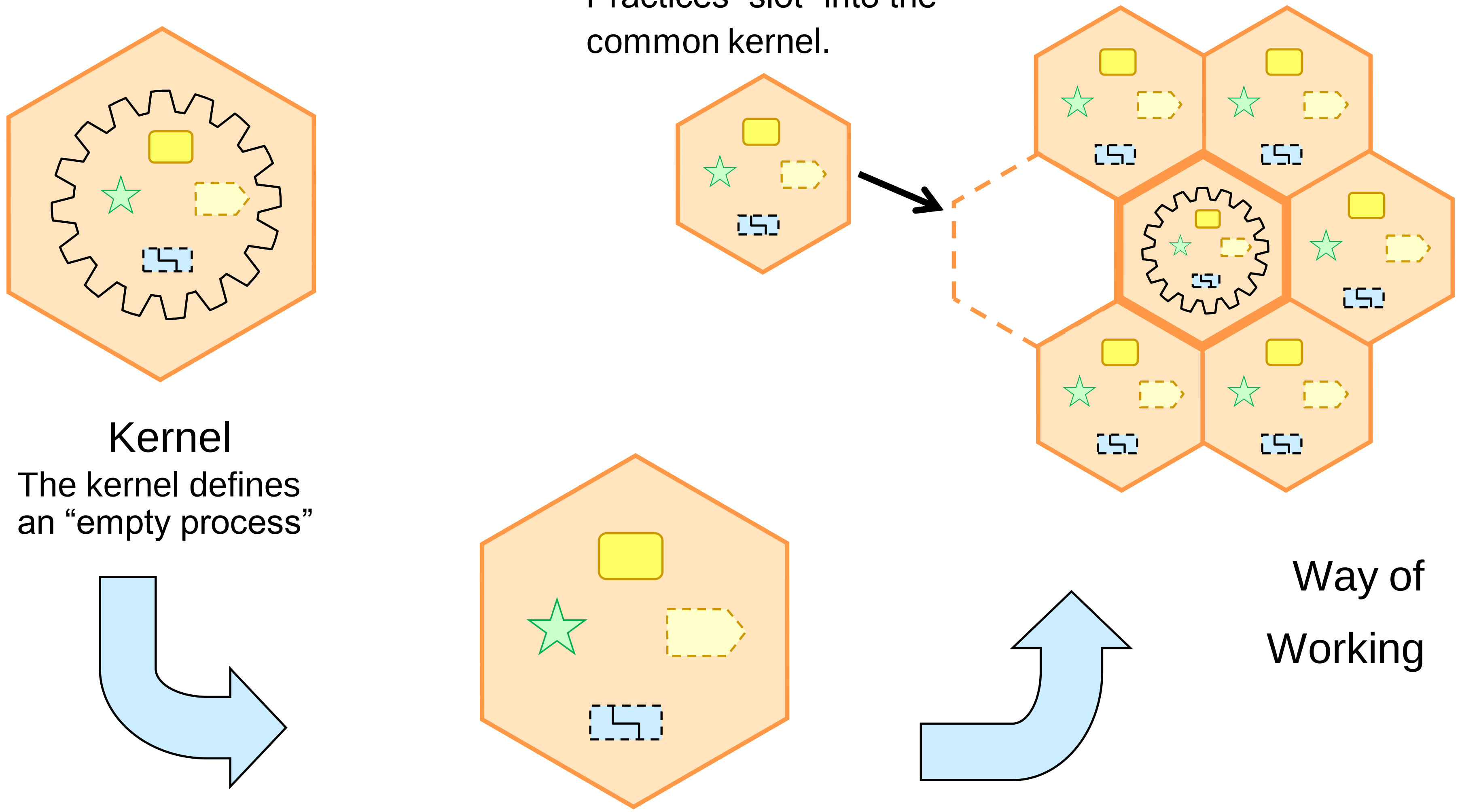
A Good Practice combines well with other practices

- Practices are separate but not independent – like use cases
- A Practices has a particular position in a practice architecture – The Kernel is such an architecture baseline

Agenda

1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

You need a kernel



Practices “slot” into the common kernel.

Kernel

The kernel defines an “empty process”

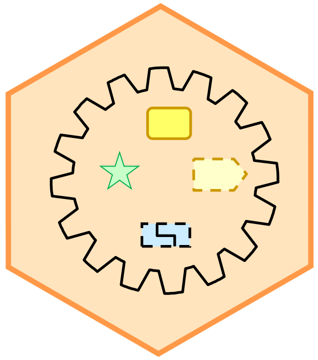
Practice

Each practice contains practice-specifics to add to the kernel.

Way of Working

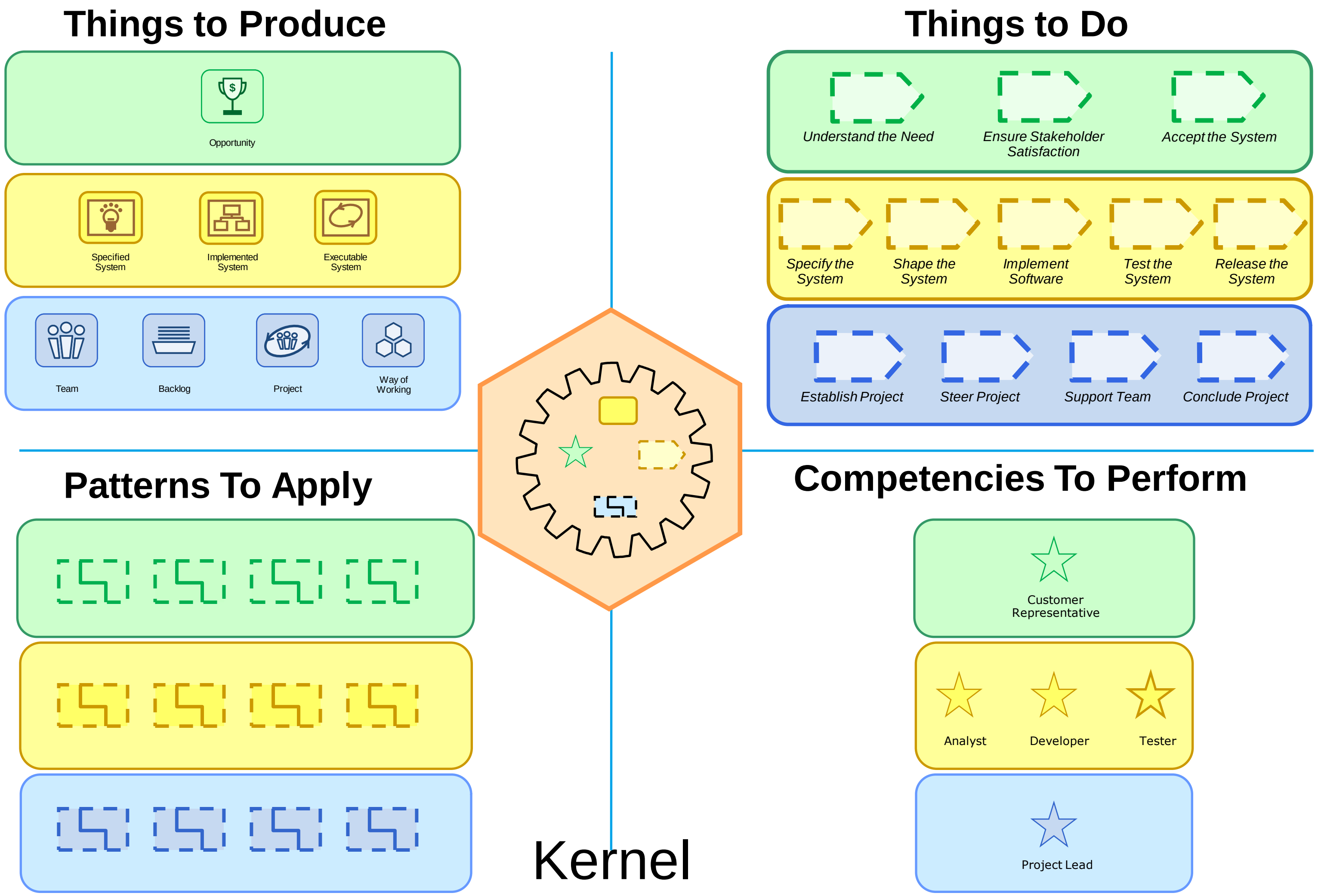
Start Understand the Kernel

- The Kernel is very small, extracted from a large number of teams way-of working
  - It contains empty slots for things that every process have
  - Slots for
    - Competencies, such as analyst, developer, tester
    - Things to work with , such as backlog, implementation, executable system
    - Things to do, such as implement the system, test the system
- The Kernel is practice agnostic

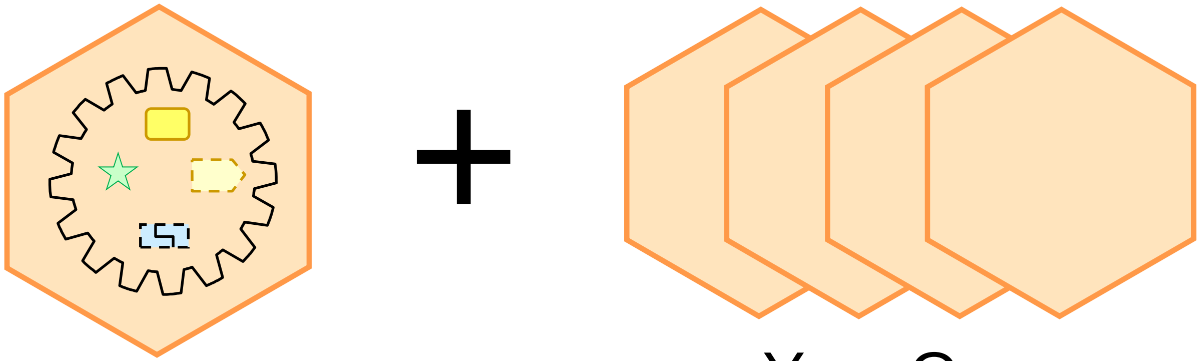

Kernel

Start with the Kernel

- The Kernel contains empty slots for things that every process have



Use the Kernel to Harvest Your Own Best Practices



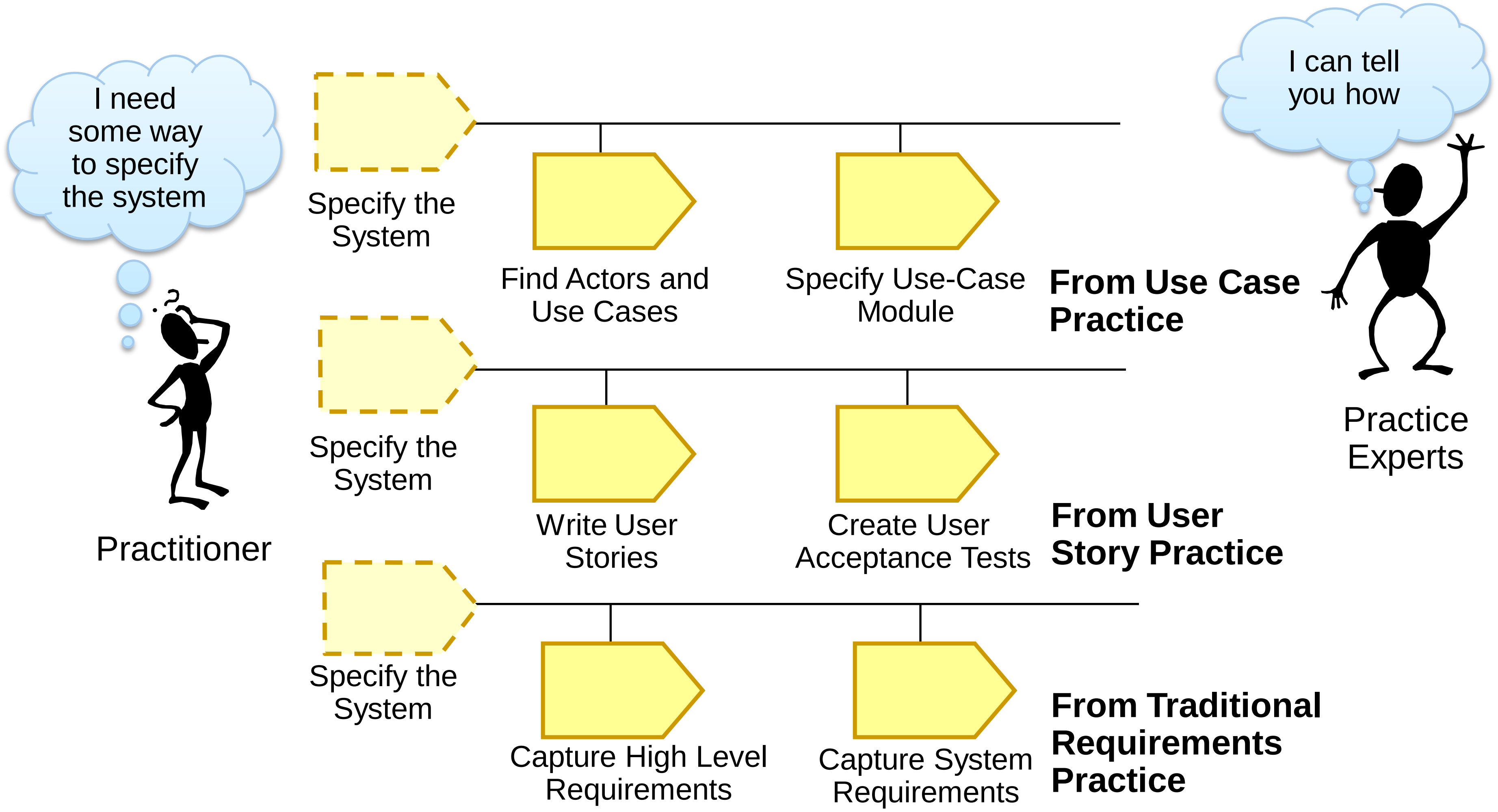
Kernel

Your Own

Best Practices

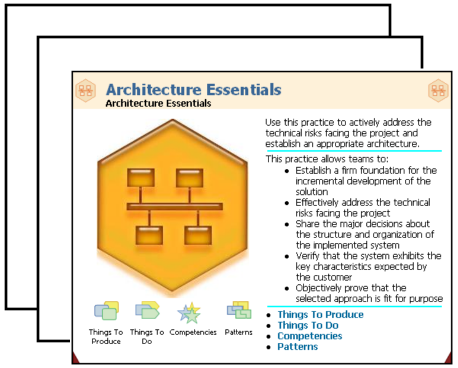
Add Your Practices on top of the Kernel

- Example for adding activities (from various practices) onto an activity space



Practice overlays many cards on the kernel

practices are aligned to the kernel (practice architecture)



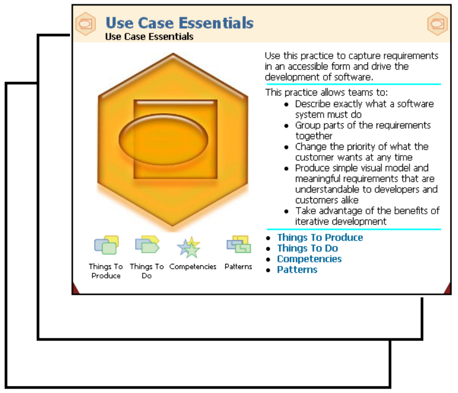
Architecture Essentials
Architecture Essentials

Use this practice to actively address the technical risks facing the project and establish an appropriate architecture.

This practice allows teams to:

- Establish a firm foundation for the incremental development of the solution
- Effectively address the technical risks facing the project
- Share the major decisions about the structure and organization of the implemented system
- Verify that the system exhibits the key characteristics expected by the customer
- Objectively prove that the selected approach is fit for purpose

- Things To Produce
- Things To Do
- Competencies
- Patterns



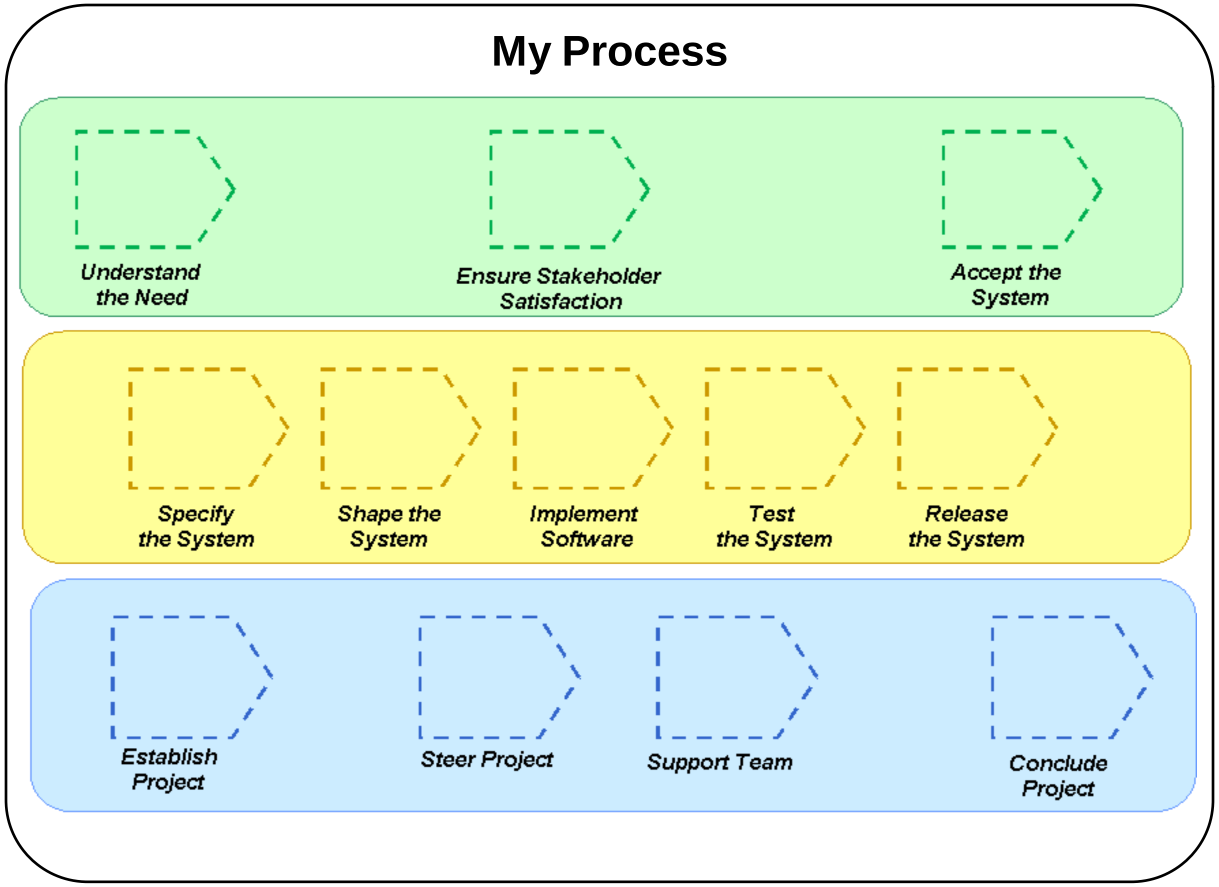
Use Case Essentials
Use Case Essentials

Use this practice to capture requirements in an accessible form and drive the development of software.

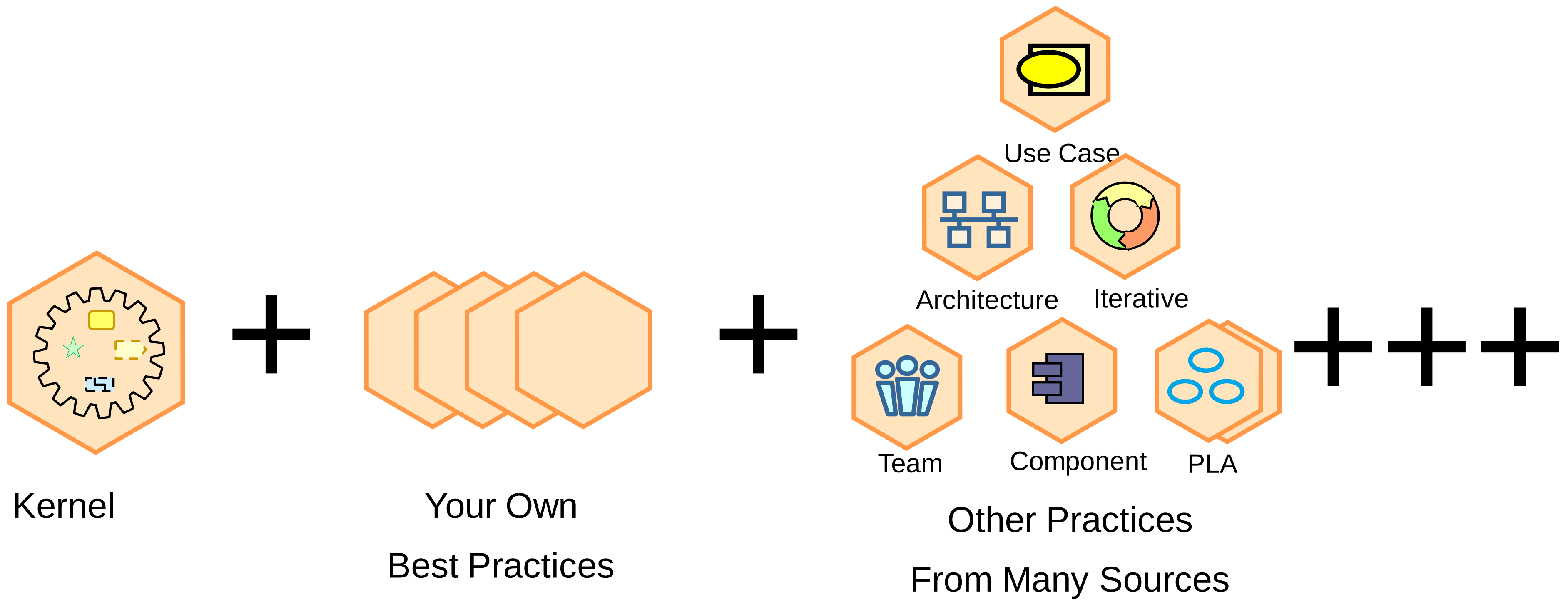
This practice allows teams to:

- Describe exactly what a software system must do
- Group parts of the requirements together
- Change the priority of what the customer wants at any time
- Produce simple visual model and meaningful requirements that are understandable to developers and customers alike
- Take advantage of the benefits of iterative development

- Things To Produce
- Things To Do
- Competencies
- Patterns



Improve Your Process by Adding Other Practices



Practices enable projects to run the way they need to

Project A

Use Case Architecture Component Iterative

Project B

Declarative Requirements Architecture Waterfall

Project C

User Story Architecture Iterative

Way of Working = A subset of the practices in the practice architecture

Practices enable projects to run the way they need to

Project A

Use Case Architecture Component Iterative

Project B

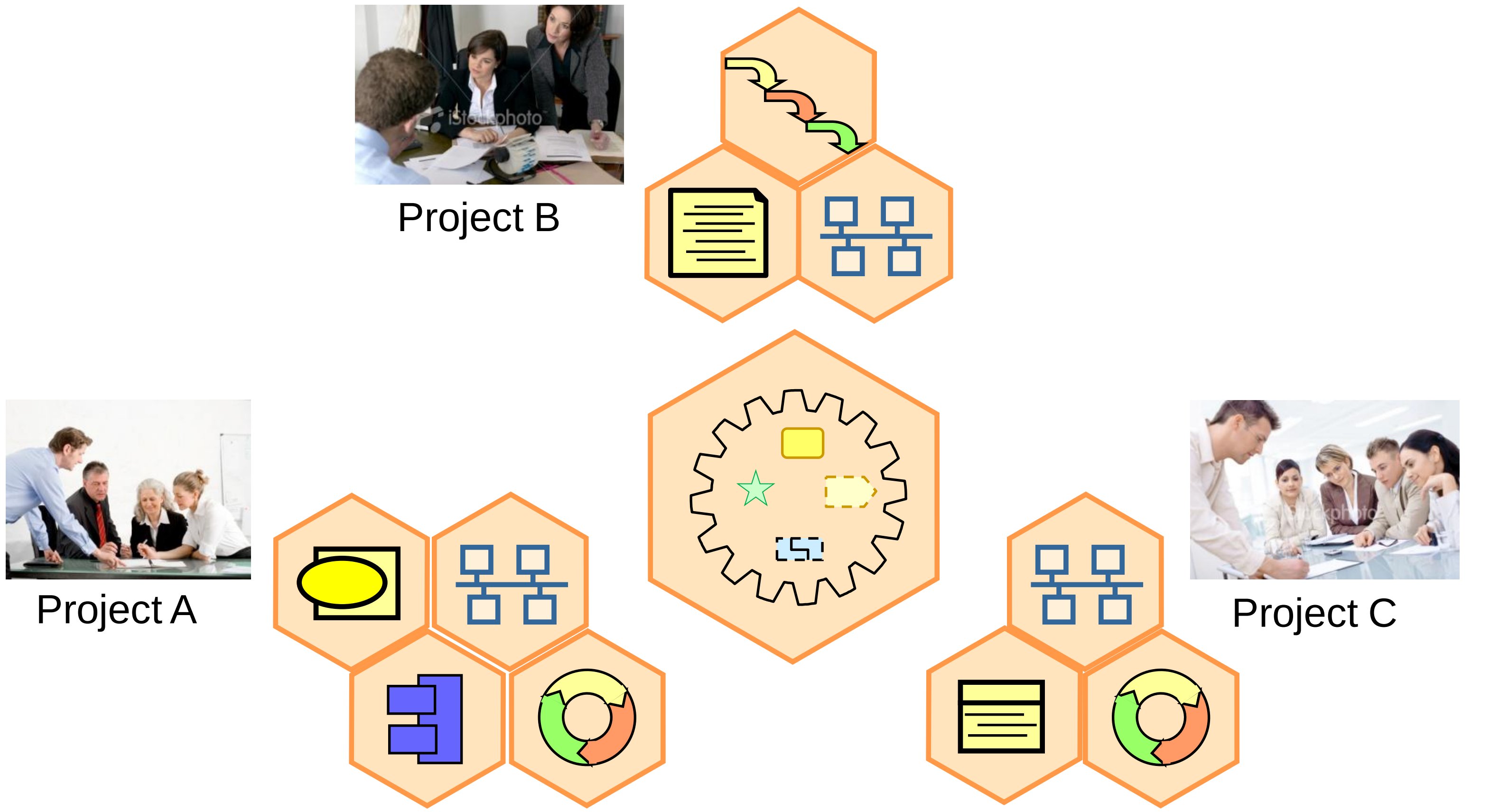
But how can we manage these projects if they all have different processes?

Project C

User Story Architecture Iterative

Way of Working = A subset of the practices in the practice architecture

Practice Architecture is Important



The kernel ensures common understanding across teams in a minimal way

Agenda

1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

You need some basic technical practices

Component

Architecture

Iteration

Use Case

... or Scrum, User Stories, Test-Driven Design...

Good Software

Useful

Extensible

Reliable

You need some more advanced technical practices

Practices for Significant Reuse

Product Line Architecture

SOA

Enterprise Architecture

Basic Practices For Good Software

Architecture

Iterative

Use Case

Component

Low Cost

Large Scale Reuse of Components

Nothing is more important than competent and motivated people

Team Practice

“Creating the right working environment to enable the team to excel.”

Social engineering patterns

- Self-Directing Team
- Frequent Demonstration to Stakeholders
- Team Retrospective
- Everyone Contributes What They Can
- Common Ownership
- Keep the Team Small
- Self-Adapting Team
- Everyone is a tester
- Create alternative career paths
- Managing cross-cutting teams

Quickly

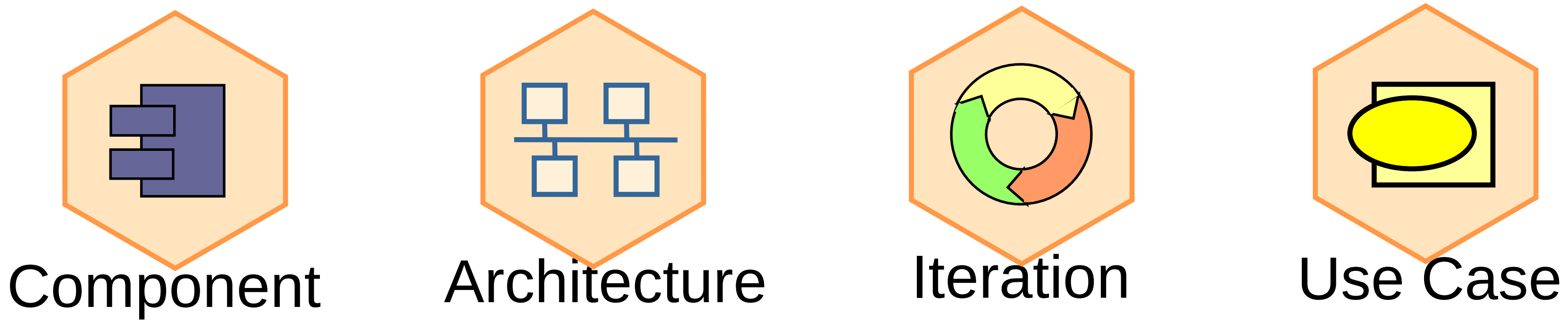
Competent & Motivated People

Agenda

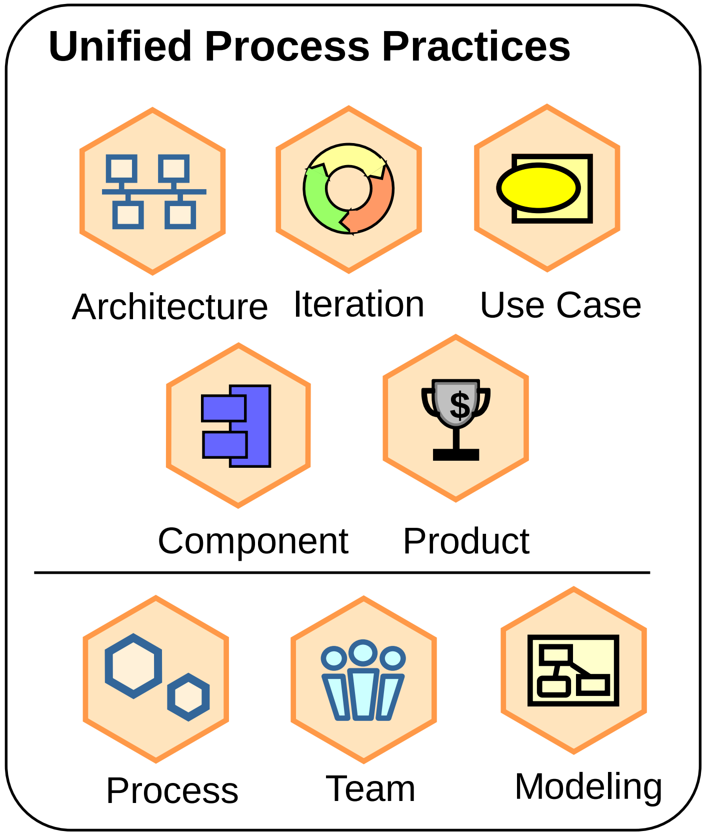
1. The Goal: Good Software Quickly and at Low Cost
2. Practices have become First-Class Citizens
3. Practices should focus on the Essentials
4. Using practices to build a process
5. Practices for Good Software, Quickly and at Low Cost
6. Wrap up

Why practices are different than processes

- You can learn practices individually
- You can apply practices separately
- You can adopt the practices you want, when you want, and at the pace that suits you
- You can mix-and-match practices from any source
- You only have to change the practices that need changing
- Different teams can adopt different practices according to their needs



How do you get started?



Select the most valuable practices and start using them.

"The way to get started is to quit talking and begin doing."

Walt Disney (Pioneer of animated cartoon films, 1901-1966)

Practices – Not Process help you to …

- Good Software
- Quickly
- Low Cost

There are practices and Practices.

Good Practices should

- focus on the Essentials
- start from a Kernel – a practice architecture
- be Smart
- be Executable

IBM Rational Software Development Conference 2008

WHERE TEAMS ARE R-HEROES

IBM

Rational. software