Beyond Agile: Smart

Ivar Jacobson

Jaczone
ivar@jaczone.com

Ivar Jacobson Consulting in the
USA, UK, China, Korea &
Singapore
ivar@ivarjacobson.com
Today everyone is agile!

- Everyone wants to join the "Agile" club
- Method M is now presented as Agile M
- Me too! See my article:
  - A Resounding 'Yes!' to Agile Processes—But Also to the 'Next Big Thing' in Cutter IT Journal, January 2002.
Beyond agile: Smart!

- Our path to the future is very different compared to the most popular notions. It is smart!

Smart is Agile +++

I am not just agile. I am SMART

We are agile

XP, UP, etc.
Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
We agree that knowledge is needed

- Software development is very complex
- You need knowledge about
  - Programming languages and environments
  - Systemware and middleware – J2EE, .NET, MOM, DBMS
  - Packaged solutions, webservices, legacy systems
  - Business, requirements, analysis, design, coding, test
  - Design patterns, architecture, re-factoring
  - All kinds of tools (Rose, Eclipse, CC, ant, maven, bugzilla, etc.)
  - Etc.
- Where do you learn all this? 😊
You can get knowledge from books . . .
Are books the answer?

You will need 30 books or so

- Books written by different authors
- Books written with different terminology and approach
  - Overlapping, conflicting
- Books with 30 different focus
- Nobody can read 30 books
- No single book has it all
How to deal with knowledge? Two fundamentally different approaches...

- Both Agile Methods and Unified Process support agile principles, such as
  - iterative,
  - continuous integration & testing,
  - use only what you need,
  - customer focused,
  - use-case/scenario driven,…

- Both agree that success comes from knowledge and experience

- But, how you deal with knowledge is fundamentally different

- Extremes are Tacit (Implicit) Knowledge and Explicit Knowledge
What is tacit knowledge?

- It is ad hoc and implicit knowledge
  - Ad hoc means it is unstructured
  - Implicit means it is assumed

- Articles or books selected by individuals at their own choice
- Training or classes in areas of personal interest

- Experience from previous work
- Experience from other team members
What is explicit knowledge?

- It is knowledge engineered in a structured way
- It describes each micro-step to build software
  - precise decision rules to guide practitioners

- Consistent knowledge
- Comprehensive
- Non-overlapping, no conflicts

- Clearly formulated
- Easily communicated
- Actionable
Example of explicit knowledge?

Given feature X, how should I proceed?

- If X is distinct requirement
  - model as a use case
- If X is an add-on to
  - some existing requirement
  - model as a use case
- If X involves an entire
  - business process
  - model as use case package
- If existing system is legacy
  - system
  - model as existing use cases by just focusing on extension points

Create use case package
Create extension use cases

Given use case Y, how should I proceed?
Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
What does agility stands for?

- Manifesto for Agile Software Development
  - Individuals and interactions over process
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- That is, while there is value in the items on the right, we value the items on the left more

www.agilealliance.org
Of course, who doesn’t agree with that

Individuals and interactions over process
- Of course, people are more important than the steps laid out and described in a book about process, since books don’t produce software.

Working software over comprehensive documentation
- Yes, but the code needs to be understood and maintained after your initial development team has moved on.

Customer collaboration over contract negotiation
- Yes, requirements are difficult to specify at the beginning of a project; they evolve through iterations, but they need to be written down in some form for future reference.

Responding to change over following a plan
- Yes, software development is an ongoing change process, and the project should be flexible about accommodating changes. So, small plans are to be preferred.
What’s wrong with Agile Movement?

- Agile movement relies on tacit knowledge

Just follow these principles, you can figure out the details yourselves
Problem with Tacit Knowledge

- Everyone has different tacit knowledge

Reaching common understanding takes time and much debate!

The use of tacit knowledge makes process heavier ≈ NOT agile
Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
Ultimately we want to have good software
The crown jewels of UP

- UP has explicit rules for goodness
  - How to create a good component
  - How to create a good use case
  - How to make a good sequence diagram
  - What constitutes a good test case
  - There are 1000s of these rules
- To attack this problem was the Objectory vision in 1987

The crown jewels of UP is about goodness of design
It provides the rules and micro-steps
UP is Explicit Knowledge – it is a Process Definition

It has a Process Meta-Model (Who does what, when and how)

This is defined in OMG-SPEM
Agility through explicit knowledge

Now, that I know the steps explicitly, I can be much faster. I can even automate some of the steps. I can be very agile about this.

Mr X
Who has explicit knowledge

```
public class ReserveRoomHandler {
    public void makeReservation() {
        ... ...
    }
    ... ...
}
```
Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
Explicit knowledge vs Tacit Knowledge

Light Explicit Knowledge (Process Definition) \(\Rightarrow\) Need more Tacit Knowledge \(\Rightarrow\) Heavy Process

Rich Explicit Knowledge (Process Definition) \(\Rightarrow\) Need less Tacit Knowledge \(\Rightarrow\) Agile Process
Comparing Knowledge in UP and Agile Methods

Agile Methods
- Extreme Programming Project
- User Stories
- Architectural Spike
- Release Planning
- Iteration
- Test Scenarios
- Requirements
- New User Story
- Software Evolution
- Bugs
- Latest Version
- Acceptance Tests
- Customer Approval
- Small Releases

Unified Process
- Role: Designer
- Activities: Find Design Classes, Distribute Behavior
- Artifact: Use Case Realization

May result in

Heavy Process

Agile Process

http://www.extremeprogramming.org/map/project.html

However, there is a Problem with Explicit Knowledge

- For given problem

Find the knowledge

Control the use of knowledge

Learn the knowledge

Apply the knowledge
Using explicit knowledge

**Find the knowledge**

**Critique:**
Configuration and specialization of the PD is not agile

**Reality**
However, UP comes with many development cases from very light to very rich
You pick the one that fits your project Thus agile

**Learn the knowledge**

**Critique:**
Training and mentoring is not agile

**Reality**
But it is doable as opposed to not doable (as for Tacit Knowledge)

**Apply the knowledge**

**Critique:**
Applying UP is not agile,

**Reality**
However UP allows us to train our people
Only competent people can be smart Successful software is developed by smart people

**Control the use of knowledge**

**Critique:**
Controling UP is not agile

**Reality**
However UP provides a good model for understanding the project artifacts, relationships and status
Successful software is controlled by smart people.
Need explicit knowledge without being heavy-weight

- UP provides an explicitly structured knowledge base
- Knowledge base is rich
- However, it (specifically its process definition) is perceived as heavy-weight

We need to be smart!
Where is reliance on tacit and explicit knowledge heading?

Agile Methods rely primarily on Tacit Knowledge

Path to be smart

Unified Process rely primarily on Explicit Knowledge

Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
Smart process: under the hood

- Smart process gets the micro-steps and rules into process execution in a seamless way
- Can we get these rules into the process without needing humans to learn all of them?

Inject knowledge non-intrusively
Smart process executes process definition

I am not just agile. I am SMART

Smart Process Definition Executed

Here knowledge is injected by the process
Process is active
Practitioner works with the process

UP
Process Definition is static

Characteristics of a Smart Process

- Minimal training – learn as you go
- Make process invisible – yet very present
- Make it personal, make it light
  - Without sacrificing quality
- Give context-dependent, concrete advices
- Make people collaborate
- Let them focus on creative tasks

A process engine in the hands of each developer and the whole team.
Use Next Generation Technology: Intelligent Agents

- Characteristics of intelligent agent:
  - I am an intelligent agent
  - I have knowledge,
  - I am autonomous
  - I am goal-oriented

- Agents are proactive, adaptive and reactive
  - Think of intelligent agents as objects driven by rules
Pair Programming

- If XP talks about Pair Programming, then Intelligent Agents are:
  - Virtual Pair Programmers
  - Virtual Pair Analysts
  - Virtual Pair Designer
  - Virtual Pair Tester
  - Virtual Pair Project Managers
  - Etc.
Smart Process Needs Explicit Knowledge

- Only UP provides explicit knowledge – a rich process definition
Agenda

- You need knowledge
- Agile uses tacit knowledge
- UP makes knowledge explicit
- The truths about knowledge
  - Process and process definition
  - Comparing tacit and explicit knowledge
- Being smart with the Next Generation Process
- Being really smart
There is nothing as agile as competent and knowledgeable people
How does agents make you smart?

Find the knowledge
Find
With Active Process much of the tailoring will occur dynamically as you run the project

Learn the knowledge
Learn
With Active Process we get training as we work and we get virtual mentors

Apply the knowledge
Apply
With Active Process we can make people competent Only competent people can be smart

Control the use of knowledge
Control
With Active Process you have more control, since virtual team members will help you check your work

What more does smart stand for?

- Manifesto for Smart Software Development
  - Make well-known knowledge explicit over keeping it tacit
  - Active process over passive process
  - Let models be code over letting code be models
  - Team capability over key individuals

Process as friend over process as enemy

www.smartalliance.org

This website is taken by someone else 😕
Smart software development explained

- Make well-known knowledge explicit over keeping it tacit.
  - We should not need to spend time getting people to re-invent the well-known stuff.
  - Nor waste time explaining it
  - Well knowledge should be made explicit and easily accessible and learnt.
Smart software development explained

- Active process over passive process.
  - The practitioner no longer uses the process as something static which needs to be learnt, etc.,
  - Instead the process works together with the practitioner actively as peers.
Smart software development explained

- Let models be the code over letting the code be the models.
  - Since the detailed stuff are handled by the active process, practitioners can work effectively at higher level of abstraction.
  - The practitioners work with models.
  - The active process will translate the intent to the code.
  - This is more human centric than machine centric

`public class ReserveRoomHandler {  
  public void makeReservation() {  
    ...  
  }  
  ...  
}`
Smart software development explained

- Team capability over key individuals
  - Instead of letting knowledge be in the heads of some key individuals, let knowledge be shared by the team.
  - There is no over-stress or over burden on key people.
  - The team can share the workload.

Over reliance on key individuals

Total team capability
Next Generation Process

1\textsuperscript{st} Generation
\textit{tacit}

Ad Hoc Knowledge in textbooks
SA&SD, OOSE, Booch, OMT, XP

2\textsuperscript{nd} Generation
\textit{explicit}

Knowledge Structured Engineered PD
UP

3\textsuperscript{rd} Generation
\textit{smart}

Knowledge Executable Active Process
UP with Intelligent Agents (e.g. WayPointer)
Questions
Thank You

Ivar Jacobson