Be Smart!

or

What they don’t teach you about software at school

Ivar Jacobson

with

Ian Spence, Pan Wei Ng and Kurt Bittner

ivar@ivarjacobson.com

www.ivarblog.com
Your goal is the same as always!

Good Software, Quickly and at Low Cost!
What it takes
What it takes

Good Software
What it takes

Good Software

Useful  Extensible  Reliable
Quickly

Good Software

Useful  Extensible  Reliable
What it takes

Quickly
Competent & Motivated People

Good Software
Useful Extensible Reliable
What it takes

Quickly
Competent & Motivated People

Low Cost

Good Software
Useful Extensible Reliable
What it takes

- Quickly
  Competent & Motivated People

- Low Cost
  Large Scale Reuse of Components

- Good Software
  Useful
  Extensible
  Reliable

© 2008 Ivar Jacobson International
What it takes

Useful  Extensible  Reliable

What they don’t teach you about software at school 😊
One major obstacle…we are a fashion industry
One major obstacle...we are a fashion industry

Software Development is driven by fashions and fads

- Fifteen years ago it was all about OO
- Ten years ago it was about components, UML, Unified Process
- Five years ago it was about RUP and CMMI
- Two years ago it was about XP
- Today it is about Scrum

All good, but none is all you need
Software Development is driven by fashions and fads

- Fifteen years ago it was all about OO
- Ten years ago it was about components, UML, Unified Process
- Five years ago it was about RUP and CMMI
- Two years ago it was about XP
- Today it is about Scrum

All good, but none is all you need

The software industry keeps looking for a silver bullet
Software Development is driven by fashions and fads

- Fifteen years ago it was all about OO
- Ten years ago it was about components, UML, Unified Process
- Five years ago it was about RUP and CMMI
- Two years ago it was about XP
- Today it is about Scrum

All good, but none is all you need

The software industry keeps looking for a silver bullet

This is unsmart!
One major obstacle…we are a fashion industry

Yes, Software Development is driven by fashions and fads

- Fifteen years ago it was all about OO
- Ten years ago it was about components, UML, Unified Process
- Five years ago it was about RUP and CMMI
- Two years ago it was about XP
- Today it is about Scrum

All good, but none is all you need

There is no silver bullet!

Software industry keeps looking for silver bullets

This is unsmart!
One major obstacle… we are a fashion industry

Yes, Software Development is driven by fashions and fads

- Fifteen years ago it was all about OO
- Ten years ago it was about components, UML, Unified Process
- Five years ago it was about RUP and CMMI
- Two years ago it was about XP
- Today it is about Scrum

All good, but none is all you need

The software industry keeps looking for silver bullets

This is unsmart!

We have to work smarter

© 2008 Ivar Jacobson International
1. What does Smart mean?
2. Smart Cases – Recognize it when you see it
3. How do you become Smart
4. What does Smart really mean?
What does *Smart* mean?

Things should be done as simple as possible – but no simpler

© 2008 Ivar Jacobson International
What does *Smart* mean?

Things should be done as simple as possible – but no simpler

- *Albert Einstein*
What does *Smart* mean?

Things should be done as simple as possible – but no simpler.

- *Albert Einstein*

\[ E = mc^2 \]
Smart and Intelligent?

Mr Smart
Smart and Intelligent?

• Being Smart is not the same thing as being intelligent
  – You can be intelligent without being smart, and
  – You can be very smart without being very intelligent
Smart and Agile?

Mr Smart
Smart and Agile?

- Being Smart is an *evolution* of Being Agile
  - Agile means being flexible and adaptable.
  - Agile provide simple/lightweight starting points
  - But being smart is knowing when to go beyond agile
    - Knowing when to follow the rules and when to break them
    - Knowing when to be consistent and when to change
    - Knowing when to grow and when to shrink
Smart and Agile?

• Being Smart is an *evolution* of Being Agile
  – Agile means being flexible and adaptable.
  – Agile provide simple/lightweight starting points
  – But being smart is knowing when to go beyond agile
    • Knowing when to follow the rules and when to break them
    • Knowing when to be consistent and when to change
    • Knowing when to grow and when to shrink

Smart = Agile ++
Agenda

1. What does Smart mean?
2. Smart Cases – Recognize it when you see it
   1. People
   2. Teams
   3. Projects
   4. Requirements
   5. Architecture
   6. Modeling
   7. Test
   8. Documentation
   9. Process
   10. Knowledge
   11. Outsourcing
   12. Tools
3. How do you become Smart
4. What does Smart really mean?

What they don’t teach you about software at school 😊

www.ivarblog.com
Not smart with People

Some companies view process and tools as more important than people
Some companies view process and tools as more important than people

This is unsmart!
Not smart with People

Some companies view process and tools as more important than people

This is unsmart!

A fool with a tool is still a fool but a dangerous fool
Case study: Ericsson AXE – the largest commercial success story ever in Sweden

– We had no tools and no defined process
– Despite this, we developed components, use cases, and a modeling language now part of UML
– This could only have been done with people – good people
Case study: Ericsson AXE – the largest commercial success story ever in Sweden

– We had no tools and no defined process
– Despite this, we developed components, use cases, and a modeling language now part of UML
– This could only have been done with people – good people

Software is developed by people, not by process and tools.
Case study: Ericsson AXE – the largest commercial success story ever in Sweden

– We had no tools and no defined process
– Despite this, we developed components, use cases, and a modeling language now part of UML
– This could only have been done with people – good people

Software is developed by people, not by process and tools.
Not smart with Teams

- Many software projects involve 20+ people
- Often organized into stove-pipe groups:
  - Requirements, Analysis, Design, Coding, Testing, etc.
Not smart with Teams

- Many software projects involve 20+ people
- Often organized into stove-pipe groups:
  - Requirements, Analysis, Design, Coding, Testing, etc.

This is unsmart!
Smart with Teams

- Teams are cross-functional
  Including analysts, developers, testers etc…
- Ideal size of the team is less than 10 people
Smart with Teams

• Teams are cross-functional
  Including analysts, developers, testers etc…

• Ideal size of the team is less than 10 people

A software team is like a sport team with all needed competencies to win.
Smart with Teams

- Teams are cross-functional
  Including analysts, developers, testers etc…
- Ideal size of the team is less than 10 people

A software team is like a sport team with all needed competencies to win.
Not smart with Projects
• Most companies still follow the waterfall approach
Most companies still follow the waterfall approach.
• Most companies still follow the waterfall approach
Not smart with Projects

• Most companies still follow the waterfall approach
Most companies still follow the waterfall approach.
• Most companies still follow the waterfall approach
Most companies still follow the waterfall approach
Most companies still follow the waterfall approach

This is unsmart!
• Build a skinny system to demonstrate that you have eliminated all critical risks
• Add more capabilities on top of that skinny system
• Build a skinny system to demonstrate that you have eliminated all critical risks
• Add more capabilities on top of that skinny system
• Build a skinny system to demonstrate that you have eliminated all critical risks
• Add more capabilities on top of that skinny system
Smart with Projects

- Build a skinny system to demonstrate that you have eliminated all critical risks
- Add more capabilities on top of that skinny system

This is smart!

Skinny System

Think big, build in many steps
Not smart with Requirements
Not smart with Requirements

• Many managers (and customers) believe you can detail all the requirements upfront...
• ...and based on these can accurately predict the cost of the solution
Not smart with Requirements

• Many managers (and customers) believe you can detail all the requirements upfront...
• ...and based on these can accurately predict the cost of the solution

Thou shalt work with fixed requirements for fixed prices
Not smart with Requirements

• Many managers (and customers) believe you can detail all the requirements upfront...
• ...and based on these can accurately predict the cost of the solution

Thou shalt work with fixed requirements for fixed prices

This is unsmart!
Not smart with Requirements

• Many managers (and customers) believe you can detail all the requirements upfront...
• ...and based on these can accurately predict the cost of the solution

A constant in software development is that requirements **always** change

© 2008 Ivar Jacobson International
Smart with Requirements

• Base early decisions on lightweight requirements and detail as and when it is needed
  – Use case outlines, feature lists or user stories

• Remember requirements are negotiable and priorities will change

I understand your needs, let’s work together to make sure we develop the right system for the right price.
Smart with Requirements

- Base early decisions on lightweight requirements and detail as and when it is needed
  - Use case outlines, feature lists or user stories
- Remember requirements are negotiable and priorities will change

I understand your needs, let’s work together to make sure we develop the right system for the right price.

Design your project for requirement changes
• Base early decisions on lightweight requirements and detail as and when it is needed
  – Use case outlines, feature lists or user stories

• Remember requirements are negotiable and priorities will change

I understand your needs, let’s work together to make sure we develop the right system for the right price.

Design your project for requirement changes
Not smart with Architecture
Two extremes:
Not smart with Architecture

Two extremes:

Mr Supposedly Agile

No architecture
Just Code
Refactor later
Not smart with Architecture

Two extremes:

- No architecture
  Just Code
  Refactor later

Mr Supposedly Agile

Mr Enterprise
Architect on Ivory Tower

I’ll design everything up front
Not smart with Architecture

Two extremes:

Mr Supposedly Agile

No architecture
Just Code
Refactor later

This is unsmart!

Mr Enterprise Architect on Ivory Tower

I’ll design everything up front
Two extremes:

Mr Supposedly Agile

- No architecture
- Just Code
- Refactor later

This is unsmart!

Mr Enterprise Architect on Ivory Tower

I’ll design everything up front

The single most important determinant of a software system’s quality is the quality of its architecture
Skinny System
Full Fledged System
Architectural Blue Print
Focus on the skinny system
• Focus on the skinny system
• But an architecture without executable code is a hallucination
Smart with Architecture

- Focus on the skinny system
- But an architecture without executable code is a hallucination
- Refactor over releases, but large refactoring
• Focus on the skinny system
• But an architecture without executable code is a hallucination
• Refactor over releases, but large refactoring is very costly
Smart with Architecture

- Focus on the skinny system
- But an architecture without executable code is a hallucination
- Refactor over releases, but large refactoring is very costly

Start to build a skinny system, add muscles in later steps
• Focus on the skinny system
• But an architecture without executable code is a hallucination
• Refactor over releases, but large refactoring is very costly

Start to build a skinny system, add muscles in later steps

© 2008 Ivar Jacobson International
Not smart with Test
We have two classes of people: Developers and Testers
We have two classes of people: Developers and Testers

– Developers are the creators…it is OK to create bugs as well
We have two classes of people: Developers and Testers

- Developers are the creators…it is OK to create bugs as well
- Testers are the cleaners in the software world
We have two classes of people: Developers and Testers

- Developers are the creators…it is OK to create bugs as well
- Testers are the cleaners in the software world

Testing is done as an after thought – too late and too expensive
We have two classes of people: Developers and Testers

- Developers are the creators… it is OK to create bugs as well
- Testers are the cleaners in the software world

Testing is done as an after thought – too late and too expensive

This is unsmart!
We are all testers!
Whatever you do you are not done until you have verified that you did what you wanted to do.
We are all testers!

Whatever you do you are not done until you have verified that you did what you wanted to do.
Not smart with Documentation
Not smart with Documentation

• There has been an over-emphasis on teams producing documentation
• There has been an over-emphasis on teams producing documentation

Thou shalt follow the document template I give you to document every part of the system.
• There has been an over-emphasis on teams producing documentation

Thou shalt follow the document template I give you to document every part of the system.

This is unsmart!
Myth: The idea that you document software so people later can read what you did.
Myth: The idea that you document software so people later can read what you did.
Myth: The idea that you document software so people later can read what you did.

– Law of nature: People don’t read documents
Myth: The idea that you document software so people later can read what you did.

- Law of nature: People don’t read documents

Focus on the essentials - the placeholders for conversations – people figure out the rest themselves
Myth: The idea that you document software so people later can read what you did.

- Law of nature: People don’t read documents

Focus on the essentials - the placeholders for conversations – people figure out the rest themselves
Not smart with Process/Methodology/Approach

- Agile
- Unified Process
- CMMI

Existing Way of Working

Project Lead
Not smart with Process/Methodology/Approach

Agile

Unified Process

CMMI

Existing Way of Working

Project Lead

© 2008 Ivar Jacobson International
Not smart with Process/Methodology/Approach

Agile

Unified Process

CMMI

Project Lead
Not smart with Process/Methodology/Approach
Not smart with Process/Methodology/Approach

I want a new way of working

Agile

Unified Process

CMMI
Not smart with Process/Methodology/Approach

I want a new way of working

New Way of Working

Agile

Unified Process

CMMI
Not smart with Process/Methodology/Approach

I want to have some of CMMI

I like to have some of unified process

Let me start with Agile, maybe

© 2008 Ivar Jacobson International
Not smart with Process/Methodology/Approach

I want to have some of CMMI.
I like to have some of unified process.
Let me start with Agile, maybe.

Project Lead

Agile

Unified Process

CMMI
Not smart with Process/Methodology/Approach

I want to have some of CMMI
I like to have some of unified process

Project Lead

Agile
Unified Process
CMMI
Not smart with Process/Methodology/Approach

- Agile
  - I like to have some of the unified process
- Unified Process
  - Does not work
- CMMI
  - Project Lead

© 2008 Ivar Jacobson International
I want to have some of CMMI
Not smart with Process/Methodology/Approach

I want to have some of CMMI

I cannot get it together

© 2008 Ivar Jacobson International
Not smart with Process/Methodology/Approach

I want to have some of CMMI

Agile

Unified Process

CMMI

Project Lead
I want to have some of CMMI

This is unsmart!
We have got enough of process

Agile
I want to have some of CMMI

Unified Process

CMMI

This is unsmart!

Project Lead
Existing Way of Working
Existing Way of Working
Don’t throw out your baby with the bathwater:
1. start from your existing way of working,
2. find your pain points, and
3. change one practice at the time.
Don’t throw away your baby with the bathwater:

1. start from your existing way of working,
2. find your pain points, and
3. change one practice at the time.
And the Law of Nature: People don’t read process books

New Way of Working

Don’t throw away your baby with the bathwater:
1. start from your existing way of working,
2. find your pain points, and
3. change one practice at the time.
And the Law of Nature: People don’t read process books. So focus on the essentials, people figure out the rest themselves.

New Way of Working

Don’t throw away your baby with the bathwater:
1. start from your existing way of working,
2. find your pain points, and
3. change one practice at the time.
And the Law of Nature: People don’t read process books  
So focus on the essentials, people figure out the rest themselves

New Way of Working

Don’t throw away your baby with the bathwater:  
1. start from your existing way of working,  
2. find your pain points, and  
3. change one practice at the time.
1. What does Smart mean?
2. Smart Cases – Recognize it when you see it
3. How do you become Smart
4. What does Smart really mean?
Summary: Key Elements to Smart Cases

- **People**: Software is developed by people, not by process and tools.
- **Teams**: A software team is like a sport team with all needed competencies to win.
- **Project**: Think big, build in many steps.
- **Requirements**: Design your project for requirement changes.
- **Architecture**: Start to build a skinny system, add muscles in later steps.
- **Modeling**: Don’t duplicate the effort by modeling everything
- **Testing**: Whatever you do you are not done until you have verified what you wanted to do.
- **Documentation**: Focus on the essentials - the placeholders for conversations – people figure out the rest themselves
- **Process**: Don’t throw away your baby with the bathwater:
  1. start from your existing way of working,
  2. find your pain points, and
  3. change one practice at the time.
How do you become Smart?

- You need knowledge in *good* (maybe best) practices
  - There are 100’s of practices, some of them are good

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspect Orientation</td>
<td>Robustness Analysis</td>
<td>Retro-spectives</td>
<td>Business Process Re-Engineering</td>
<td>Use-Case Driven Development</td>
<td>Pair Programming</td>
</tr>
<tr>
<td>PSP</td>
<td>User Stories</td>
<td>SOA</td>
<td>Prince2</td>
<td>Use-Case Modeling</td>
<td>Program Management</td>
</tr>
</tbody>
</table>
How do you become Smart?

• You need knowledge in *good* (maybe best) practices
  – There are 100’s of practices, some of them are good

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

• And you need experience in using them
How do you become Smart?

• You need knowledge in *good* (maybe best) practices
  – There are 100’s of practices, some of them are good

• And you need experience in using them

Training and mentoring are key
How do you become Smart?

• You need knowledge in *good* (maybe best) practices
  – There are 100’s of practices, some of them are good

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

• And you need experience in using them

... and you know where to get it, right?
Agenda

1. What does Smart mean?
2. Smart Cases – Recognize it when you see it
3. How do you become Smart
4. What does Smart really mean?
What does Smart really mean?

- If you didn’t get what smart is so far let me summarize it to you
Of course, eventually it comes back to you, but

We can all become smarter.
Thank You

Contact pernilla@ivarjacobson.com
Meet us at the booth