Death by dogma versus assembling agile

Sander Hoogendoorn
Principal Technology Officer & Global Agile Thoughtleader
Capgemini
Sander Hoogendoorn

Capgemini
Principal technology officer
Global agile thought leader
Chief architect Accelerated Delivery Platform

Other
Author books on UML, agile
Author +200 articles, columns
Speaker +100 international conferences
Microsoft Partner Advisory Council .NET
Editorial boards & Advisory boards
Capping IT Off Blog

Web
www.sanderhoogendoorn.com
sander.hoogendoorn@capgemini.com
@aahoogendoorn
9 ноября. Актовый зал ХНУРЕ

Приглашаем всех студентов на доклад и ответы на вопросы слушателей:

САНДОР ХУТЭНДОРН, НОРВЕГИЯ, www.norwegianarchitecture.com

Главный технологический директор Саржентен по стратегическому развитию. Оперативный директор по управлению разработкой Саржентен — Accelerated Delivery Platform (ADP).

Известный специалист в области методологии Agile, архитектуры, бизнес-процессов, XML, .NET, Java и средств разработки ПО. Преподает технологию разработки программного обеспечения в различных организациях, регулярно выступает на конференциях и является членом Microsoft's Advisory Board.

Автор ряда публикаций и книг по LML и методологии Agile.

Тема встречи: гибкие методологии разработки Agile

Вход свободный
12.00 – 14.00

Организатор: SIGMA | www.sigmaukraine.com
2203 Muteren Aanvullende Gevalsegegevens

BSN

Zoek

Zoek Parijsen

Aanvullende gevalsggegevens

Begindatum gegevens
Datum einde wachtijd 52 weken
Datum verlengde einde wachtijd WAG aanwezig
Datum einde wachtijd 104 weken
Datum verkorte wachtijd WIA
Datum einde vrijwel loondoorboet WIA
Datum loonsanctie WAG/WIA

Bericht van webpagina
Fout off
OK
Why waterfall won’t work
Figure 2. Implementation steps to develop a large computer program for delivery to a customer.
Waterfall?

STEP 3: DO IT TWICE
After documentation, the second most important criterion for success revolves around whether the product is totally original. If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned. Figure 7 illustrates how this might be carried out by means of a simulation. Note that it is simply the entire process done in miniature, to a time scale that is relatively small with respect to the overall effort. The nature of this effort can vary widely depending primarily on the overall time scale and the nature of the critical problem areas to be modeled. If the effort runs

Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.
Agile!
You would believe ...

because waterfall doesn’t work, right?
So the methodology doesn’t work

@aahoogendoorn They should never have used waterfall.

@aahoogendoorn Does the name of the project coincidently start with a C?
But how would you feel if ...

@aahoogendoorn

22 Sep

aahoogendoorn Sander Hoogendoorn

Heard about a Scrum project that spent several million Euros, and didn’t deliver anything. As said, agile nor Scrum are silver bullets. :)

22 Sep

aahoogendoorn Sander Hoogendoorn
@aahoogendoorn They didn’t apply Scrum right.

@aahoogendoorn This project likely did ScrumBut, not Scrum.

So it’s not the methodology, right?
SCRUM FAIL
1. **Shu** (守しゅ, “protect”, “obey”) — traditional wisdom — learning fundamentals, techniques, heuristics, proverbs
2. **Ha** (破は, “detach”, “digress”) — breaking with tradition — finding exceptions to traditional wisdom, reflecting on their truth, finding new ways, techniques, and proverbs
3. **Ri** (離り, “leave”, “separate”) — transcendence — there are no techniques or proverbs, all moves are natural
Jack states that we have over 300 resources who are trained as SCRUM masters. As it happens, I'm meeting him this afternoon. I'll ask him.
Lowering Our Fences
Scrumman
Dogma

From Wikipedia, the free encyclopedia

For other uses, see Dogma (disambiguation).

Dogma is the established belief or doctrine held by a religion, or a particular group or organization.[1] It is authoritative and not to be disputed, doubted, or diverged from, by the practitioners or believers. Although it generally refers to religious beliefs that are accepted without reason or evidence, they can refer to acceptable opinions of philosophers or philosophical schools, public decrees, or issued decisions of political authorities.[2] The term derives from Greek δόγμα "that which seems to one, opinion or belief"[3] and that from δοκεω (dokeo), "to think, to suppose, to imagine".[4] Dogma came to signify laws or ordinances adjudged and imposed upon others by the First Century. The plural is either dogmas or dogmata, from Greek δόγματα. Today, it is sometimes used as a synonym for systematic theology.
Crusader Agile
Scrumdamentalism

Flow of Conversation in a Daily Pseudo-Scrum Meeting

Flow of Conversation in a Daily (real) Scrum Meeting
Stand up meetings
Sit down meetings...
Scrumdamentalism

As a librarian, I want to be able to search for books by publication year.
Agilists against Zenifying

Just write down small things on small papers. It's your kaizen.
Agilists against Zenifying

Don’t just write down small things on small papers. Write code. It’s your job.
There is no so thing as one-size-fits-all agile
Teams and roles
Customer, Coach, Developer
The Team is responsible for delivering the product. A Team is typically made up of 5–9 people with cross-functional skills who do the actual work (analyse, design, develop, test, technical communication, document, etc.). It is recommended that the Team be self-organizing and self-led, but often work with some form of project or team management.
Customer, User, Domain Expert
Project Manager, Coach, Developer, Tester

Create project proposal

Write project plan

Deliver working software

Stabilize software

Maintain software
Multiple roles
Teams?
What is the key to being successful as a team?
Collaboration
Self-organization
But what happens to old roles?
An example team

A typical Scrum team?

Product owner /1
Business analyst /2
Information analyst /2
  SAP CRM /1
  SAP XI/ BPM /2
  SAP ABAP /1
UI developer /1
.NET developer /1
Java developer /1
  Tester /2
Scrum master /1
  Agile coach /1
Rowing Contest Collaboration

Enterprise Architects
End Users
Development Team
Test Team
Offshore Development Team
The Bob-the-Builder-Syndrome

Can we build it? Yes, we can!

@ahooogendoorn
The backlog

Where does it come from?
And on the seventh day Ken created the backlog
The automagical backlog
The automagical backlog
Preliminary iterations
Preliminary iterations
Preliminary iterations

- Create project proposal
- Write project plan
- Deliver working software
- Stabilize software
- Maintain software
Documentation Frenzy
As a librarian, I want to be able to search for books by publication year.
User stories

As a [type of user]
I want [some goal]
so that [some reason].
User stories 😞

As proven by
I can login
After I buy something I
get a confirmation email
But if your IT landscape looks like this...

@ahooogendoorn
Index cards might just not do the trick

As a librarian, I want to be able to search for books by publication year.

@ahooogendoorn
User stories are merely meant to get the conversation going?

As a librarian, I want to be able to search for books by publication year.
So what about documentation?

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
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.
The agile manifesto doesn’t say no documentation (or modeling)
As a librarian, I want to be able to search for books by publication year.

Will you document to maintain?
Eventually your software will go into maintenance (hopefully)
## Levels of requirements

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kite</td>
<td>Requirements</td>
</tr>
<tr>
<td>Sea</td>
<td>User Goal</td>
</tr>
<tr>
<td>Fish</td>
<td>Sub Function</td>
</tr>
<tr>
<td>Clam</td>
<td>Too Low</td>
</tr>
</tbody>
</table>

@aahoogendoorn
Huge cases

Hard to build, impossible to test
Different levels of use cases

User goal

Sub function

Using this approach, very limited use is made of the UML use case diagram modeling technique. Requirements are captured in text, and are mostly described in Word documents.

Smart use cases

Alternatively, when the use cases at user goal level are identified (even better: the elementary business processes for the project have been identified), it is possible to use the use case diagram technique to add use cases at first level to the diagram.
Smart use cases
Work item life cycle
Quality?

![Graph showing the increase in effort (AUFWAND) over time (ZEIT) with stages: Analyse, Entwurf, Testen, Betrieb.](image-url)
Quality in iterations

↑

AUFWAND

i1

i2

i3

i4

...  

i\text{n}

ZEIT  \rightarrow
Quality per work item
Smart Use Case Cycle

1. Define work on use case
2. Write test cases
3. Generate and build use case
4. Run test cases
5. Adjust use case
6. Describe use case
7. Write test cases
8. Accept use case
9. Define work on use case
Work item life cycle
Life cycle dashboard
Kanban is "**JUST**" an approach to improve your processes

<table>
<thead>
<tr>
<th>Toyota's six rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not send defective products to the subsequent process</td>
</tr>
<tr>
<td>The subsequent process comes to withdraw only what is needed</td>
</tr>
<tr>
<td>Produce only the exact quantity withdrawn by the subsequent process</td>
</tr>
<tr>
<td>Level the production</td>
</tr>
<tr>
<td>Kanban is a means of fine tuning</td>
</tr>
<tr>
<td>Stabilize and rationalize the process</td>
</tr>
</tbody>
</table>
The theory of constraints

1. **Identify the system’s constraint(s).**
   That which prevents the organization from obtaining more of the goal in a unit of time.

2. **Decide how to exploit the system’s constraint(s).**
   How to get the most out of the constraint.

3. **Subordinate everything else to above decision.**
   Align the whole system or organization to support the decision made above.

4. **Elevate the system’s constraint(s).**
   Make other major changes needed to break the constraint.

5. **Go back to step 1**

And remember: a chain is no stronger than its weakest link
On when to estimate
When?
Again ... preliminary iterations
The overall model

Develop an overall model
- Conduct a domain walkthrough
- Approved by IAR

Build features list
- Determine feature sets and groups
- Approved by IPL

Planning
- Assign sets and classes to partners
- Approved by IDL

Design by Feature
- Design sequence diagrams
- Approved by team leaders
- Implement
- Conduct code inspection and unit testing
- Approved by IUR

Object Model

Categorized Features

Development Plan & Environment Mock-up

Design Packages

User valued functions
Smart use case model

Create project proposal

Write project plan

Deliver working software

Stabilize software

Maintain software
Guesstimation
Apples

Team 1

@aahoogendoorn
Apples and apples

Team 1

Team 2
Apples and oranges
Distributed Apples

Team 1  Team 2  Team 3
Off shore Team
Mandatory burn down chart?

Bad smell: note how the same example is used in everyone’s presentations.

Don’t trust a vendor presentation if it has this example of a burndown chart in it.
We have our ups and downs
Lightweight agile can be to enterprise projects

What Monopoly is to solving the financial crisis
Agile is a sliding scale
Assembling Agile
Static versus Dynamic Agile
Project Approach

```csharp
public interface IApproach
{
    List<ITeam> Teams { get; set; }
    IDashBoard Board { get; set; }
    IUnitOfWork Unit { get; set; }
}

public abstract class Approach : IApproach
{
    public List<ITeam> Teams { get; set; }
    public IDashBoard Board { get; set; }
    public IUnitOfWork Unit { get; set; }
}
```
public class Scrum : Approach
{
    public Scrum()
    {
        Teams = new List<ITeam> { new LocalTeam() };
        Board = new TaskBoard();
        Unit = new UserStory();
    }
}

class ScrumProject
{
    public Scrum Approach = new Scrum();
}
public class Project
{
    public IApproach Approach { get; set; }
}

public class MyProject : Project
{
    public MyProject()
    {
        Approach = new Smart();
        Approach.Board = new KanbanBoard();
        Approach.Teams.Add(new LocalTeam());
        Approach.Teams.Add(new LocalTeam());
        Approach.Teams.Add(new UkranianTeam());
        Approach.Unit = new Feature();
    }
}
Assembling Agile – The basics of agile

1. Short Iterations
2. Collaborative Teams
3. Small Unit of Work
4. Continuous Planning
5. Deliver Early & Often
6. Simplify Communication

@ahooogendoorn
Lightweight Agile
Assembling Agile – Light

- Short Iterations
- Teams
- Collaborative Teams
- Small Unit of Work
- Continuous Planning
- Early & Often
- Simplify
- Communication

As a librarian, I want to be able to search for books by publication year.
Enterprise Agile
Assembling Agile – Enterprise
Institutionalizing agile
Freedom and flexibility
Institutionalizing agile
In retrospective
PS 024
VV 4032
Kyiv (Boryspil)

CHECK-IN TIME TILL
06:50

DEPARTURE TIME
07:20

12.11.12 International Airport Kharkiv 06:05
Agile is no religion

So don’t be a zealot
Agile is a sliding scale
Assembling Agile
Value is found

In all agile approaches
(and yes, even in waterfall)
And please can we cut the fluffiness

And go back to work?
Sander Hoogendoorn
www.sanderhoogendoorn.com
@ahooogendoorn