

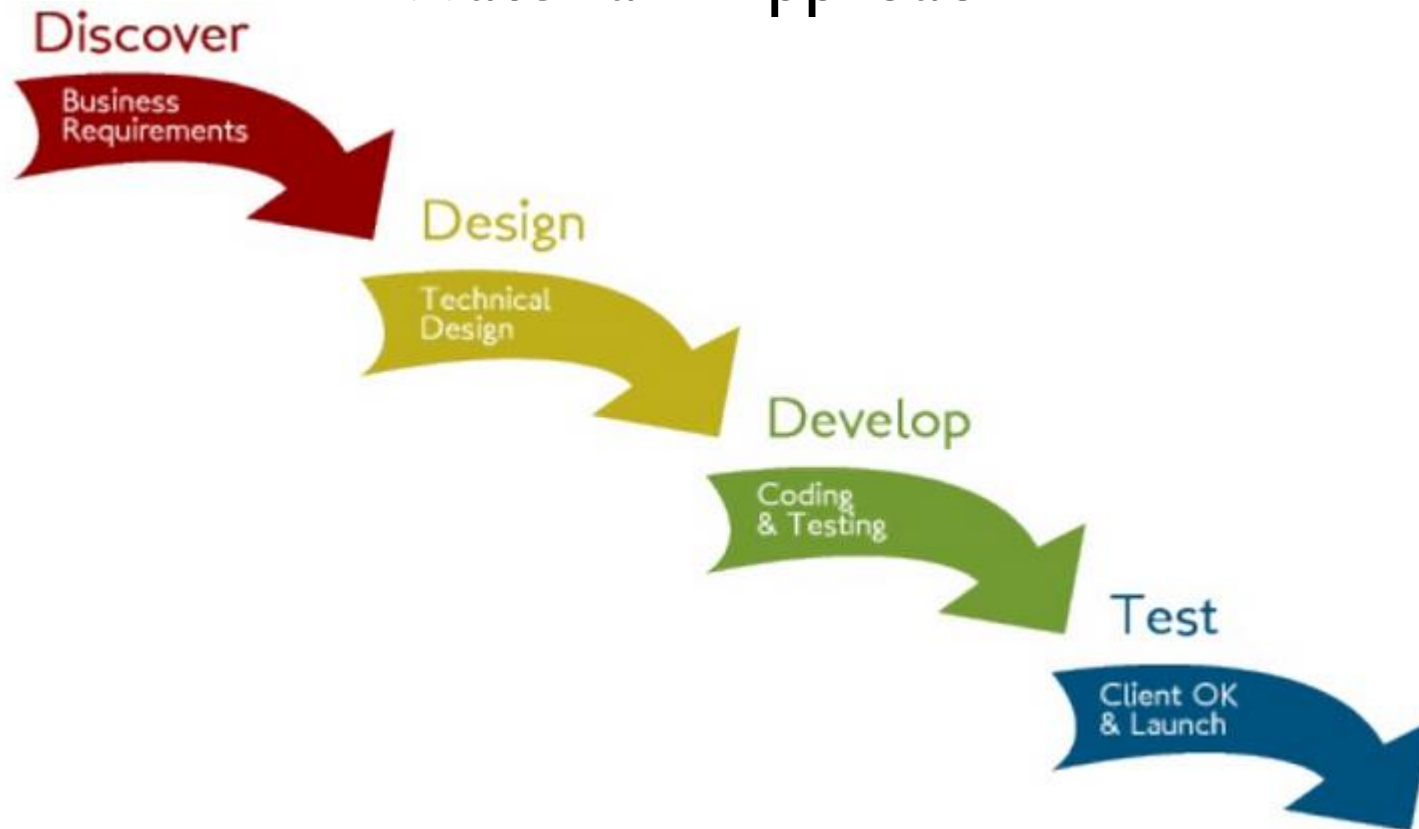
# AGILE SOFTWARE DEVELOPMENT

A NEW PARADIGM



# TRADITIONAL SOFTWARE DEVELOPMENT

## “Waterfall” Approach




Source: Synerzip. "Building Enterprise Class Software". March 2013.

# THE AGILE MANIFESTO

- **Individuals and interactions** over processes and tools:
  - In agile development, self-organization and motivation are important, as are interactions like co-location and pair-programming
- **Working software** over comprehensive documentation:
  - Working software will be more useful and welcome than just presenting documents to clients in meetings
- **Customer collaboration** over contract negotiation:
  - Requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important
- **Responding to change** over following a plan:
  - Agile development is focused on quick responses to change and continuous development



# PRINCIPLES

- 
1. **High customer satisfaction** through early and continuous delivery of working software
    - ➔ The product is continuously adjusted to fit the customer's requirements
  
  2. **Welcome changing requirements** even late in the process and keep the software flexible
    - ➔ Customer satisfaction is no. 1 goal
  
  3. Working software is **delivered frequently (weeks instead of months)**
    - ➔ Frequent mini-builds allow bug tracking and sorrow testing (both technical and business related)



#### 4. **Working software is principal measure of progress**

→ New ideas are immediately integrated into the working beta

#### 5. Sustainable development at **constant pace**

→ The application is continuously expanded and one sprint is followed by the next in order to keep the project timeline

#### 6. **Close daily cooperation** between business and the developing teams

→ Avoid divergence between development teams and the business need



## 7. **Face to Face** as the best form of communication

→ Avoid misunderstandings and foster collaboration and knowledge sharing

## 8. Projects are built around **motivated individuals**, who should be trusted

→ Extensive and time consuming control structures are unnecessary, flat hierarchies are possible

## 9. Continuous attention to **technical excellence** and good design

→ Reduces required re-working (Get it right the first time – Principle)



## **10. Simplicity** is important

→ Unnecessary features and gimmicks are avoided in order to improve speed

## **11. Self-organizing teams**

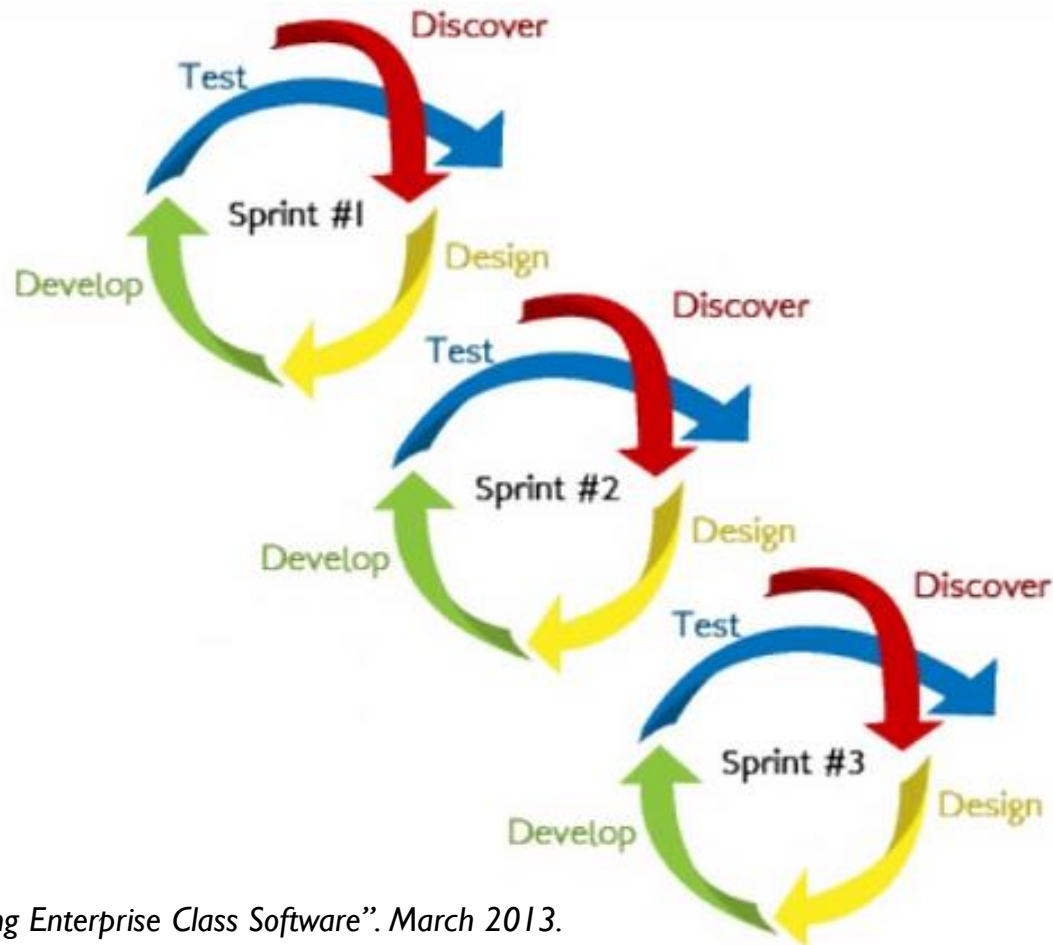
→ Team roles can be assigned flexibly to best fit changing requirements

## **12. Regular adaption to changing requirements**

→ Flexible software design allows quick and cheap reaction to changing business needs



# RESULT: A MORE INTEGRATED APPROACH



Source: Synerzip. "Building Enterprise Class Software". March 2013.



# APPLICATION CASE: FBAR CORP

# FBAR CORP: THE POLLS APP PROBLEM

## Initial Situation

FBAR CORP

Chief  
HR  
Officer



**FORTUNE<sup>®</sup>**  
**100**  
**BEST**  
**COMPANIES**  
**TO WORK FOR**

# FBAR CORP: THE POLLS APP PROBLEM

## The Requirements List

“Brian,

As Daniel discussed with you earlier, he expects the Polls App project to be completed by 31 August. Let me detail the requirements for the Polls App project for you:

1. The goal of the Polls App is to **improve the employees possibilities for participation** in the company’s decision processes.
2. The App should **allow each employee to participate** in polls that have previously been set up by management (or their assistants, respectively)
3. To keep the **poll simple and intuitive**, Daniel would like to ask the employees **three questions every three months**. The App should allow us to implement a **maximum of five answer options** among which the poll respondent can then select. The first poll needs to go out to the employees in the **first week of September, NO delays will be tolerated!**
4. Daniel wants to see **the results of the polls in a “C-level-compatible” format**. For my needs, it suffices if you simply make the results from the polls **Excel-downloadable** so I can brush them up and convert them into management reports.
5. Since Daniel is a real iPhone-lover, he would absolutely LOVE to see the App **be implemented on the mobile platform** as well so that our employees can download and answer the polls even when they’re not at work.
6. Implementation should focus on a **smooth implementation into our intranet**. Meanwhile, the stability of our intranet is paramount as we have several ongoing production batches that will not tolerate any IT-related delays.
7. The programmers can build the Polls App in **any programming language** that they like, but it is important that the **App is stringently documented** and adheres to our internal and externally given quality standards (see Guideline from the COO as of Feb 21).
8. Unfortunately, Daniel has **not granted any significant budget** for this project. He does allow you to charge 2 FTEs (Full-Time Employees) for the duration of the project though.

I am looking forward to hearing from you about the project progress! Let me know if you have any trouble.

Best,

Cliff’

# FBAR CORP: THE POLLS APP PROBLEM

## Kick-Off

- Discussion of Agile vs Waterfall
- Setting-up Agile Software Development and Polls App Requirements:
  - Programming Sprints
  - Aspired Features and Level of Sophistication
  - Documentation
  - Internal Organization
- **ASD:**
  - General Merits of ASD
  - Self-Organization
  - Working Software rather than Rigid Processes
  - Motivated Individuals

# FBAR CORP: THE POLLS APP PROBLEM

## Week I

- Set-up of the concrete Timetable
- Internal Project-Kick-Off
  - Programming Language
  - UML Model
  - Programming Sprints
  - Start of Coding
- **ASD:**
  - Self-Organization
  - Working Software
  - Frequent Delivery

# FBAR CORP: THE POLLS APP PROBLEM

## Week 2

- Business Need Realignment (No Mobile)
- Key Issues
  - Database Connection
  - Mobile Device Compatibility
- **ASD:**
  - Close Feedback Loops and tests

# FBAR CORP: THE POLLS APP PROBLEM

## Week 3

- First User Acceptance Test – Sprint
  - Early, simple Prototype
  - Actual Users
  - Feedback on Usability, Design, Technical Functionality and Design
- Early Adjustments – Development of only the wanted Features
- **ASD:**
  - Tests and User Involvement & Feedback is key
  - Simplicity: Drop Unwanted Features



# FBAR CORP: THE POLLS APP PROBLEM

## Week 4 & Week 5

- Week 4:
  - Short sprint of two days and test with Brian
  - Clarification of way of illustrating survey results
  - Another two days of programming and successful test with HR department
- Week 5:
  - Adding of new feature: department-wide and department-only poll functionality
  - Due to misunderstandings with Brian, developing wrong functionality
  - At end of week meeting with Brian, clarification of misunderstandings
- **ASD:** Delivery of working software – iterative process, close cooperation, changing requirements

# FBAR CORP: THE POLLS APP PROBLEM

## Week 6

- Resolving the problem of week 5 – User restriction for department-only polls
- Key issues
  - No quick fix possible
  - Close business-cooperation required
- **ASD:**
  - Go back to the latest working version
  - Co-location of development and business team
  - Sorrow testing

# FBAR CORP: THE POLLS APP PROBLEM

## Week 7

- The second large beta test
- Key issues
  - Checking the app both technically and in regards of usability
  - Incorporating feedback also in late stages of development
- **ASD:**
  - Implementation of testing procedures
  - Continuous bug fixing
  - Obtaining and incorporating feedback from test users

# FBAR CORP: THE POLLS APP PROBLEM

## Week 8

- Calibrating the app (and rights structure changes) with C-Level-Input
- Key issues
  - Keep the application flexible and expandable for late-stage changes
  - Restriction of number of polls
  - Close User Interaction
- **ASD:**
  - Frequent calibration of efforts also with top executives (frequent customer touch points)

# FBAR CORP: THE POLLS APP PROBLEM

## Week 9 & Week 10

- Week 9:
  - Final test phase
  - Few optimizations
  - Feedback from Daniel
- Week 10:
  - Final roll out
  - Town hall meetings
  - End of project
- **ASD:** Delivery of working software, close cooperation

# COMPARISON OF ASD AND WATERFALLS PHILOSOPHIES IN KEY SITUATIONS

Situation	Week	ASD Approach	Waterfall Approach
Requirements List	Pre-Kick-Off	<ul style="list-style-type: none"> <li>Only a preliminary idea of the project is required</li> </ul>	<ul style="list-style-type: none"> <li>Holistic representation of the expected outcome required</li> </ul>
Kick-Off Meeting	Kick-Off	<ul style="list-style-type: none"> <li>Self-Organization leads to motivation and optimal resource use</li> </ul>	<ul style="list-style-type: none"> <li>Strict Separation of tasks leads to individual focus and expert knowledge that needs to be managed and shared</li> </ul>
Timetable of the project	Week 1	<ul style="list-style-type: none"> <li>Regular sprints lead to tight time management</li> <li>Deviations of expected results surface early</li> </ul>	<ul style="list-style-type: none"> <li>Very Strict, time management, everything must be planned ahead, low flexibility to incorporate changing requirements</li> </ul>
Problems faced while Mobile Device Programming	Week 2	<ul style="list-style-type: none"> <li>Close feed-back loops and close business interactions allow quick reactions</li> </ul>	<ul style="list-style-type: none"> <li>Misunderstanding not have surfaced until end of project -&gt; Unhappy Customer</li> </ul>
User Acceptance and Functionality Test (Beta Test)	Week 3+7	<ul style="list-style-type: none"> <li>Fast prototype with actual users that provide supportive feedback</li> </ul>	<ul style="list-style-type: none"> <li>Full Running Version Tests at end of project with detailed thus complex questionnaires</li> <li>Problems are detected when changes are already costly</li> </ul>
Implementation of Result Illustration	Week 4	<ul style="list-style-type: none"> <li>Trial &amp; Error, working together with actual users</li> <li>Prevents overspending</li> </ul>	<ul style="list-style-type: none"> <li>Proposal of programmers at end of project, if rejected: new programmer proposal</li> </ul>
Misunderstanding about Subgroup Restrictions	Week 5-6	<ul style="list-style-type: none"> <li>Easier to change</li> <li>Early recognition due to close feedback loops</li> <li>re-start from previous working versions</li> </ul>	<ul style="list-style-type: none"> <li>Would not have surfaced until the very end, lengthy revisits necessary</li> </ul>
Late-Stage Changes	Week 8	<ul style="list-style-type: none"> <li>Changes are allowed and welcomed rather than having to re-do more work later on</li> </ul>	<ul style="list-style-type: none"> <li>Given the rather inflexible set-up, small changes in late stages might lead to necessary high costs</li> </ul>
Final Testing	Week 9	<ul style="list-style-type: none"> <li>Few changes necessary as every part of the software was already audited before</li> </ul>	<ul style="list-style-type: none"> <li>Huge problems can surface, making the aspired end of project date impossible</li> </ul>
Presentations	Week 10	<ul style="list-style-type: none"> <li>Smooth transition as product "poll" is already well-known</li> </ul>	<ul style="list-style-type: none"> <li>Big surprise effects</li> <li>Implementation problems may arise</li> </ul>

# ASD PRINCIPLES IN THE DIFFERENT STAGES OF THE STORY

[illegible]

# SOURCES USED

- <http://www.ambysoft.com/essays/agileManifesto.html>
- <http://de.slideshare.net/HemantElhence/agile-softwaredevelopmentmarch2013new>
- <http://dl.acm.org/citation.cfm?id=1177267>
- <http://geekswithblogs.net/robz/archive/2007/12/28/real-life-examples-of-agile-development.aspx>
- <http://www.allaboutagile.com/agile-development-cycle/>
- <http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t>
- <http://www.l4.in.tum.de/konferenzen/Jass06/courses/3/presentations/AgileSoftwareDevelopment.pdf>
- [http://en.wikipedia.org/wiki/Agile\\_software\\_development](http://en.wikipedia.org/wiki/Agile_software_development)