Course Application Design

Creating beautiful, extendible and reliable applications

Michiel Noback
Institute for Life Sciences and Technology
Hanze University of Applied Sciences

Course contents

- In this course, we'll deal with a selection of topics relevant to creating applications for end-users
- The story told here will revolve around the primary aspects of ***agile software development***

The agile manifesto

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.

http://agilemanifesto.org

Principles behind the Agile Manifesto (1)

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Principles behind the Agile Manifesto (2)

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

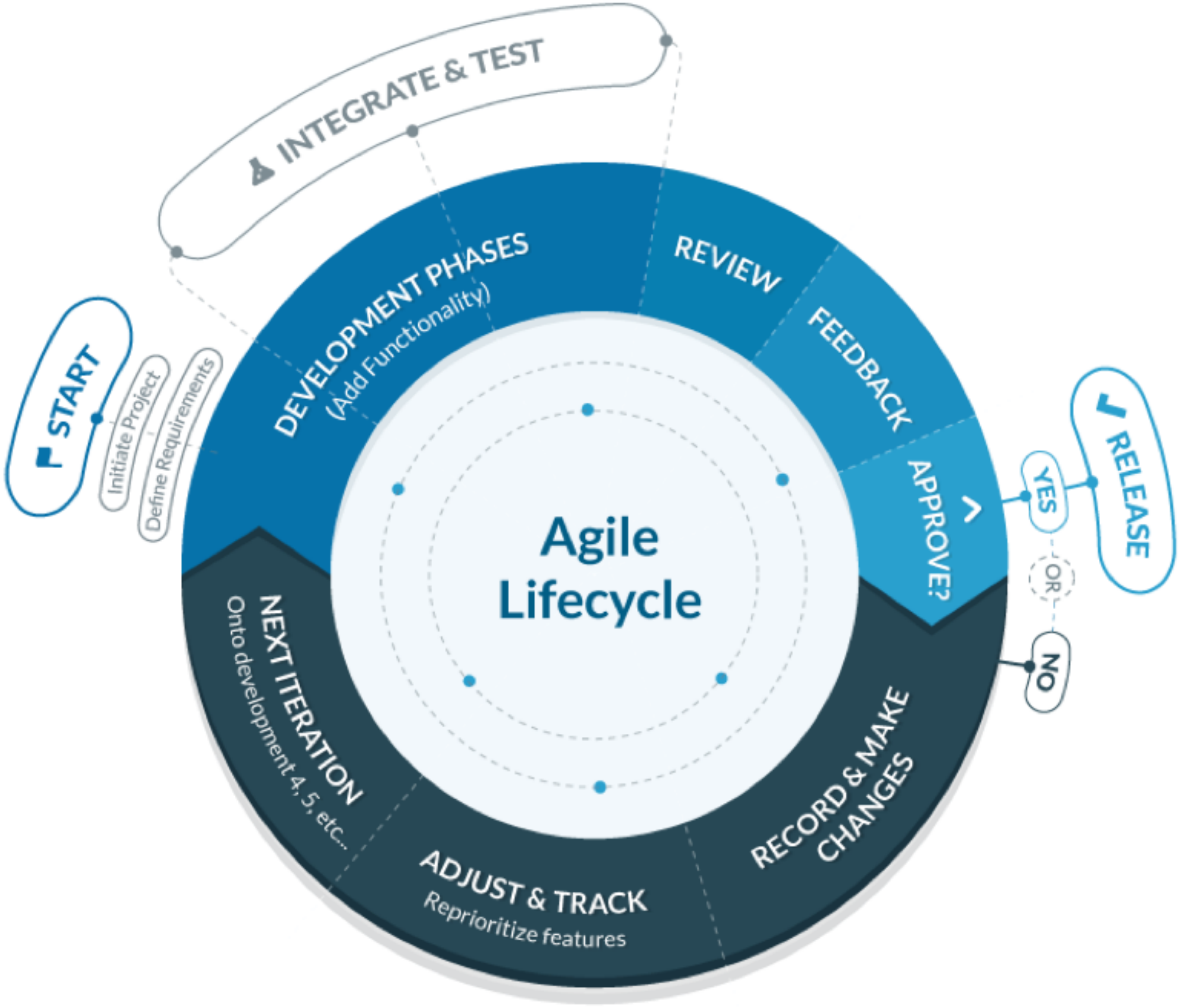
Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The agile development process



Object-Oriented design tools

- Encapsulation & Abstraction
- Polymorphism
- SOLID
- Design patterns
- Test-Driven Design (TDD)

Next part: advanced aspects

Symptoms of poor design

When we do our job well, we’ll be able to prevent these 7 symptoms:

- Rigidity – The design is hard to change
- Fragility – the design is easy to break
- Immobility – the design is hard to reuse
- Viscosity – It is hard to do the right thing
- Needless complexity – Overdesign
- Needless repetition – Mouse abuse
- Opacity – disorganized expression

Agile Software Development *Principles, Patterns, and Practices* - Robert C. Martin

Rigidity

- The design is hard to change
  - changes propagate via dependencies to other modules
  - no continuity in the code
- Reluctance to change anything becomes the policy
- Telltale sign: ‘Huh, it was a lot more complicated than I thought.’

http://staff.cs.utu.fi/~jounsmed/doos_06/slides/slides_060321.pdf

Fragility

- The design is easy to break
  - changes cause cascading effects to many places
  - the code breaks in unexpected places that have no conceptual relationship with the changed area
- fixing the problems causes new problems
- Telltale signs
  - some modules are constantly on the bug list
  - time is used finding bugs, not fixing them
  - programmers are reluctant to change anything in the code

Immobility

- The design is hard to reuse
  - the code is so tangled that it is the code is so tangled that it is impossible to reuse anything
- Telltale sign: a module could be reused but the effort and risk of separating it from the original environment is too high

Viscosity

- Viscosity of the software
  - changes or additions are easier to implement by doing the wrong thing
- Viscosity of the environment
  - the development environment is slow and inefficient
  - high compile times, long feedback time in testing,laborious integration in a multi-team project
- Telltale signs
  - when a change is needed, you are tempted to hack rather than to preserve the original design
  - you are reluctant to execute a fast feedback loop and instead tend to code larger pieces

Needless complexity

- Design contains elements that are not currently useful
  - too much anticipation of future needs
  - developers try to protect themselves against probable future changes
  - agile principles state that you should never anticipate future needs
- Extra complexity is needed only when designing an application framework or designing an application framework or customizable component
- Telltale sign: investing in uncertainty

Needless repetition

- The same code appears over and over again, in slightly different forms
  - developers are missing an abstraction
  - bugs found in a repeating unit have to be fixed in every repetition
- Telltale sign: overuse of cut-and-paste

Opacity

- The tendency of a module to become more difficult to understand
  - every module gets more opaque over time
  - a constant effort is needed to keep the code readable
    - easy to understand
    - communicates its design
- Telltale sign: you are reluctant to fix somebody else’s code – or even your own!