

# Principles of Software Design

## Software development process, Software contracts

Robert Lukočka

lukotka@dcs.fmph.uniba.sk

[www.dcs.fmph.uniba.sk/~lukotka](http://www.dcs.fmph.uniba.sk/~lukotka)

M-255

# What needs to be done for a product to be created?

- (Business analysis)
- Requirements
- Design
- Implementation (Construction)
- Verification and validation
- Deployment
- Maintenance

# Basic SW development methodologies

- Waterfall approach: Every phase starts after the previous one finishes.
- Iterative-incremental approach: You select a small part of the system, build it in a waterfall style.

Maybe the right question is: how many iterations should you do, long should an iteration take?

# Comparison: waterfall vs iterative

## Waterfall approach

- Verification and validation at the end of the process.
  - Errors within any phase are found only at the end of the project (especially bad for errors in requirements).
  - We cannot evaluate whether the objectives of the project are attained during the process.
- For long projects, it is very natural that the requirements change
  - We may not have enough knowledge on how to attain the goals of the project.
  - Outside interference (Change of strategy within the company, legislative change, etc.).
  - Change avoidance may impact the quality of the product.
- Unknowns may exist when the requirements are fixed.
  - Can this even be implemented, how fast, under which design?
- You cannot deploy partial product.

# Comparison: waterfall vs iterative

Waterfall approach:

- + Easier to manage (in most aspects).
- + Easier to determine scope, time and price.
- + Allows to focus on each phase separately.
- + Guarantees that a reasonable documentation exists.

# Comparison: waterfall vs iterative

Iterative approach:

- + Allows to deploy partial product
  - Best way to clarify the requirements and the scope of the project.
- + Deals easier with changing requirements.
- + Faster feedback (not only with respect to requirements)
  - Requires flexible design and higher quality of code
  - It is hard to determine scope, time, and price
    - This is a common requirement in the industry.

# Which approach to use

Iterative approach is typically superior. When is waterfall approach appropriate (or longer iteration / larger increments are appropriate)

- Requires stable and well defined requirements, not many unknowns in the solution.
- It is not easy to test or deploy the solution.
- High requirements on safety and reliability.
- You need to determine price, scope, and time fast (You need to waterfall at least until requirements / analysis).

# Good practices in iterative development

You will hear about this during the whole semester: Some of them

- Close collaboration with the customer.
- Various practices to have software that is easy to change.
- Automate build and testing.
- Frequent delivery.
- ...



# Feedback

Fast feedback is valuable.

- It is easier to correct an error if you detect it fast.
- You can avoid similar errors in the future.
- You can evaluate the degree the project can fulfill its objectives faster and more accurately (fail fast).
- Some feedback is more valuable than other.
  - It is important in which order we build our application.

# Feedback - examples

- The software project with requirements as defined cannot fulfill the business objectives of the customer.
- An error in the requirements.
- Incomplete requirements.
- A “bug” in the implementation.
- Two different parts of the system cannot communicate with each other.
- The architecture cannot fulfill the requirements (e.g. too much network communication required).

# Waterfall - Iterative, what else

Sometime you plan not to use what you have done:

- Prototyping
- Evolutionary

Prototyping - why to build something that will be thrown away?

- Faster and cheaper feedback

# Predictive vs. adaptive

Other two main aspects of running a project

- Predictive planning
  - The focus is on predicting time, and money necessary to produce given scope
  - Works well if there are not too many unknowns.
- Adaptive planning
  - The focus is on what should be build.
  - Constantly adapting the plan according to the feedback.

# Software development process

An functioning organization should have defined processes covering most tasks/projects types

- It is not practical for each project to have its own process.
- An organization may have several processes that share some parts..
- An improvement of the processes should be an integral part of the process.

# Agile

Not a software development process (this is quite common misconception). But it implies

- Iterative development.
- Adaptive planning.

It is defined by the set of principles proposed in [Manifesto for Agile Software Development](#)

# Software contracts

Is there a difference between buying software and tailor-made furniture?

# Software contracts

Is there a difference between buying software and tailor-made furniture?

- Software is an incredibly complex product.
- Even if you know what to build it is hard to specify everything.
- A big issue is the lack of vocabulary (customer is familiar with different domain).
- Very often the effect of the produced software on the customer is unknown.



# Software contracts

How to write a software contract when it is unknown what should be built?

- Contracts are about dividing the risks and profits.

# Software contracts

Fixed price, time, and scope.

- You need to know the scope of the project
- Implies waterfall at least until reasonably detailed requirements are set.
- It is hard to change requirements.
  - Contracts typically contain measures how to change the scope (change boards), but the software builder has mostly an upper hand in the negotiations.
  - Changes can significantly increase the price of such project (Is the price then really fixed?)
- Limited feedback.
- „Perhaps the reason these type of contracts (Fixed price, time, and scope) survive is because they can be defended in court.”  
- Allan Kelly

# Fixed price, time, and scope

- Risk on the software development side is on the developer
- The customer takes the risk that the requirements are correct

Everything is OK if this suits your case.

# Other type of contracts

In the case there are unknowns you cannot fix all of these

- price
- time
- scope

At best, you can pink two of these.

# Other type of contracts

## Iterative development

- Collaboration is encouraged.
- Thus it make sense to share the risks instead of sharply dividing who is responsible for what
  - While collaborating, it is often hard to say who is at fault.
  - Lawyers are expensive, the resources could be better used elsewhere.

# Agile contracts

## Key features

- Risk sharing (both parties are interested in project success)
  - Business goals not attained, late delivery, higher cost, smaller scope, ...
- Ability to terminate the contract at several points without major penalty.
- Contract defines processes, not requirements
  - Fail fast if one side of the contract does not believe the project.
- No detailed requirements just high level requirements and project goals.
- Measure of success is related to project goals, not requirements.

Cooperation is instrumental in such contracts.

# Agile contracts

## Agile contract principles / choices

- Manhour payment.
- Recouring contracts.
- Payment for product (and product) only if satisfactory.
- Capped time / materials.
- Money for Nothing, Change for Free.
- Contract based on business goals.

# Software tenders

How could e.g. a faculty find the company to construct its software systems?



# Software tenders

How could e.g. a faculty find the company to construct its software systems?

- Fixed time price and scope contracts are prevalent.
- Such contract are usually won by companies who know how to milk the contractor on changes.

# Software tenders

- Fixed price and time.
  - Project size estimation by estimating representative epics.
  - Parallel development with various contractors at the beginning.
- Contracts based on measurable business outcomes.
  - Such a contract has various parameters which could be the evaluation criteria
- Agile contracts.
  - How to find a contractor in a meaningful way while preserving required transparency?

# Software tenders

Most important, to do any of these:

- Smaller projects
- Modular contracting
- Requires an architect on the customer side


Moreover


- Standard tools
- Open API's.
- Open source solutions should be preferred.

# Resources

- Martin Fowler - Waterfall process
- Martin Fowler - Scope Limbering
- Martin Fowler - Fixed price contracts
- <https://developex.com/blog/10-contracts-for-your-next-agile-software-project/> Peter Stevens: 10 Contracts for your next Agile Software Project

# References I

 [Martin Fowler - Waterfall process](#)

 [SWEBOK V3 - Chapters 7 and 8](#)