3rd World Congress for Software Quality

„Shifting the Risk“

“Reflections about the different risk strategies in the sequential (waterfall) and iterative software development approach”

DI. Andreas Nehfort

andreas@nehfort.at www.nehfort.at

Introduction

- Introduction
- Characteristics of sequential and iterative SW development
- Motivation & initial statement
- The main difference:
  - The risk management strategy of the sequential approach
  - The risk management strategy of the iterative approach
- The consequences:
  - Some consequences of the sequential approach
  - Some consequences of the iterative approach
- Conclusions

Personal Info: Andreas Nehfort

Head of Nehfort IT-Consulting (since 1986)

My Focus: IT Processes - Consulting & Trainings

- Assessment Based Process Improvement
  - CMMI & SPiCE/ISO15504
- Agile Processes
- IT Project Management & IT Quality Management
- Software Requirements Analysis & Management

Software Process Assessments:

- INTACS certified SPICE / ISO 15504 Assessor
- CMMI Assessor

Software Process Models - Survey

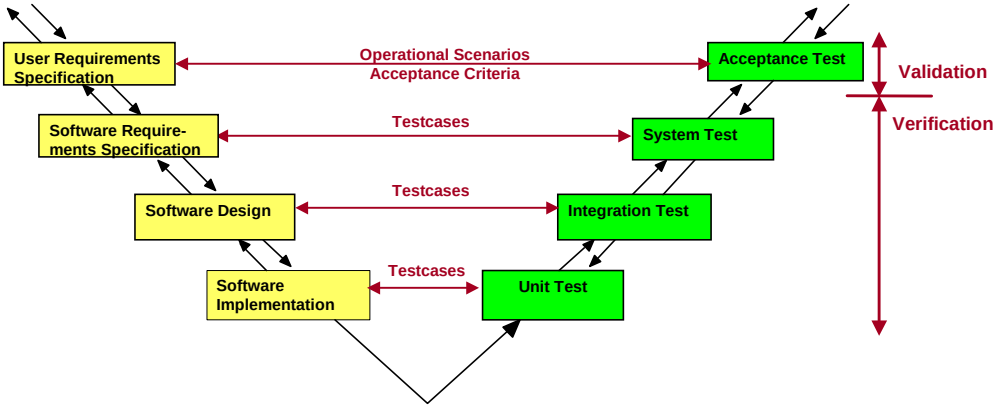
Sequential SW development approach:

- Waterfall
- V-Model in different variants

Iterative SW development approach:

- RUP - Rational Unified Process
- MSF - Microsoft Solution Framework
- Agile Processes like:
  - XP - Extreme Programming (Kent Beck, Martin Fowler)
  - FDD - Feature Driven Development (Peter Coad)

Sequential SW Development → V-Modell



The basic idea: One step behind the next

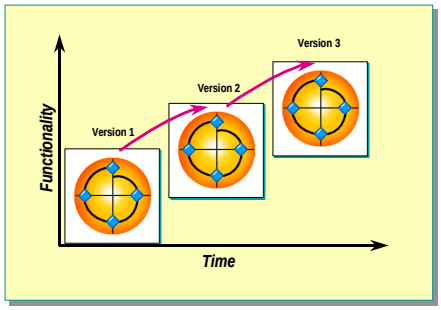
Top Ten Principles of conventional (Waterfall) Software Management

1. **Freeze requirements before design**
2. **Avoid coding prior to detailed design review**
3. **Use a high-order programming language**
4. **Complete unit testing before integration**
5. Maintain detailed traceability among all artifacts
6. Document & maintain the design
7. Assess quality with an independent team
8. Inspect everything
9. Plan everything early with high fidelity
10. Control source code baselines rigorously

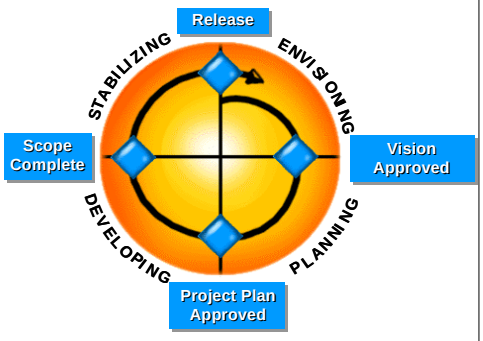
Rational Software White Paper 1998: Best Practices for SW Development Teams

Iterative SW Development

Short iterations produce running software with increasing functionality



Simple phase model for each iteration

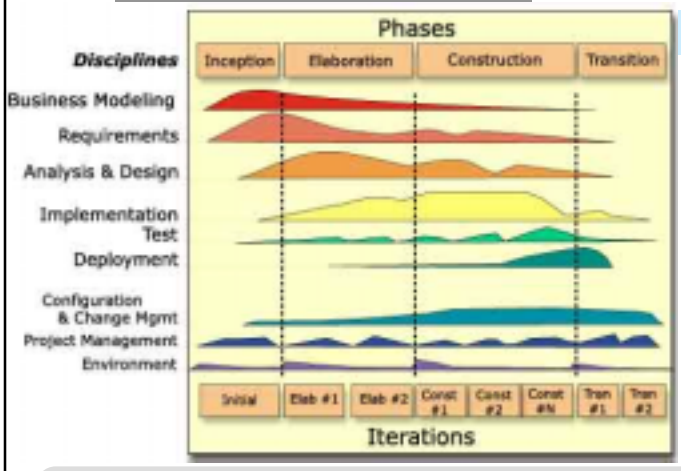


The SW-product is NOT fully defined at the beginning!

RUP – Rational Unified Process

Process Framework

9 Disciplines & Workflows



⇦ 4 Phases & Milestones:

Inception phase:
- Lifecycle Objectives

Elaboration phase:
- Lifecycle Architecture

Construction phase:
- Initial Operational Capability

Transition phase:
- Product Release

Motivation

The trend:

- Iterative SW-Development becomes more and more popular.
- It more and more replaces the sequential approach (V-Model).

My notice:

- Many organisations try to apply iterative SW-Development.
- Many of them mix up aspects of the sequential and the iterative SW development approach.

The result:

- The new procedure is not that different from the old one.
- The new procedure does not work.

Dialogue regarding the V-Model

Mr. Nehfort (asks):	Mr. Meyer (Proponent of the V- Model):
What do you think about sequential versus iterative SW- development?	I am strictly following the V-Model!
Why?	Without clear requirements in the beginning you cannot expect a reasonable solution and there is no chance for a fixed price tender!
How do you deal with the fact, that many customers have problems to specify their requirements in an early phase?	Well, the customer has to decide what he really needs; of course we help him to make a methodical analysis and an orderly requirements specification
But there will still be a requirements creep.	Therefore we implement a professional change management. We also recommend our customers to save 20% of the budget for changes after requirements freeze.
That means: In the analysis phase you typically fix about 80% of the requirements and during the project work you define or redefine the rest?	Yes, that has proved in many projects.

Dialogue regarding the RUP

Mr. Nehfort (asks):	**Mr. Smith (Proponent of the RUP):**
What do you think about sequential versus iterative SW-development?	We have recently established the RUP in our SW development!
Why?	Because it fits better to our situation!
Following the RUP you do not specify all requirements in an early stage. How do you deal with requirements?	Well, we define about 80% of the requirements in the inception phase and about 20% later.

Wow! - Two different approaches – The same outcome!

The difference: The risk strategy

The main difference between

- the sequential software development approach and
- the iterative software development approach

is based on **fundamentally different risk management strategies** regarding:

- The risk to develop the wrong product
  - Inadequate functionality & behaviour
- The risk to develop the product wrong
  - Inadequate design & technical faults.

Risk Strategy: Sequential approach

The sequential SW development approach is based on the following consideration:

- If we know **all the requirements** in an early stage, there is a pretty good chance to develop the software right.
- In other words: Our risk to make **technical faults** (SW developers call them bugs) can be minimized.

The risk management strategy behind this consideration:

“Freeze requirements first”

Freeze requirements first

- Has been the dominating paradigm since the beginning of „Software Engineering“ early in the 1970s!
- Generations of SW developers have been grown up with it!
- For many SW engineering experts it seemed to be like a „law of nature“.

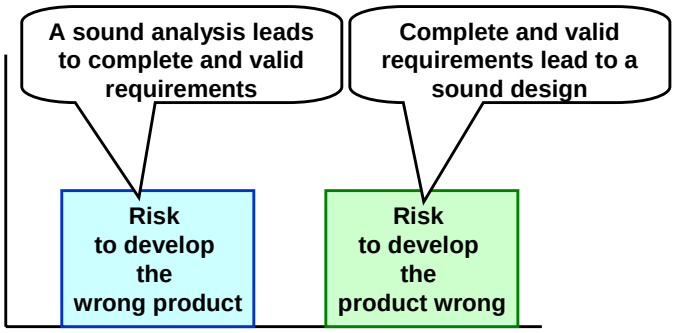
On the other hand: It was not that successful!

- Our customers did not like it! (Just as well as many SW developers!)

Alistair Cockburn:

- The people on the projects **were not interested** in learning our system!
- They were successfully able to ignore us, and we´re still delivering software, anyway!

Intended risk characteristic of the sequential approach



The sequential dilemma

The more

- a SW application shall be innovative or
- an application field evolves dynamically,

the less

- we know about the required functionality and behaviour in an early stage of the project!

If we - nevertheless - try to fix the requirements early,

- they may become moving targets and/or
- we may develop the wrong product

The Standish Groups Chaos Report

The Standish Group's Chaos Reports demonstrate impressively, that SW projects are not that successful:

	Projects succeeded	Projects challenged	Projects failed
1994	16%	53%	31%
2000	28%	49%	23%
2004	29%	53%	18%

The Standish Group International Inc.: www.standishgroup.com

In 1994 the "Projects challenged"
- had an average cost overrun of +189% and
- had implemented about 61% of the features & functions initially specified.

In 2000 the "Projects challenged"
- had an average cost overrun of +45% and
- had implemented about 67% of the features & functions initially specified.

Risk Strategy: Iterative approach

The answer to the sequential dilemma:
- If we dont`t know the requirements precisely or
- the requirements may change frequently

then we can minimize our risk
- when we dicide as late as possible,
- when we enable fast feedback,
- when we enable fast „Return on Investment"!

The risk management strategy behind this consideration:

"Decide late - try out early"

The Consequences of "freeze requirements first"

Even if we have built the software right, we often did not develop the right software:

- Our customer's understanding about their requirements has changed.
- The business has evolved during the project.
- Consequently the software requirements have changed.

Professional Provisions

The SW project managers' approach to handle their risk:

- If we can force our customers to freeze the requirements in an early stage of the project, **we can transfer the risk** having the wrong requirements **to our customer**!

Also the customers have found a way to handle their risk:

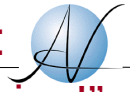
- If we have to freeze the requirements in an early stage, the contractor shall **freeze our costs** in a fixed price contract.
- Via validation test we can force him to deliver valuable software.

A fair deal: A defined solution for a fixed price!

There are still some tricky problems

- The sequential dilemma will still cause problems:
  - The customer may define the wrong requirements.
  - The SW-vendor may misunderstand the requirements.
- The time between requirements specification and testing or operation is quite long –
  - we have to spend a lot of time and money before we can prove the results and get a return on investment.
- Requirements may change:
  - The longer the project lasts the higher the risk.

A seemingly attractive work around:



“Let us make the specifications less precise!”

This saves time and money and gives room for interpretation:

- **So both sides expect to reduce their own risk.**

The customer’s risk:

- To be bound to the wrong or insufficient requirements, he had to define in an early stage of the project.

The SW developer’s risk:

- To be bound to requirements he has misunderstood and therefore consequently implemented an insufficient solution.

Of course they don’t call it “less precise specification”
they call it: “Time pressure”, “competence in the application field”, “trust”, …

One more tricky side effect of "freeze requirements first"

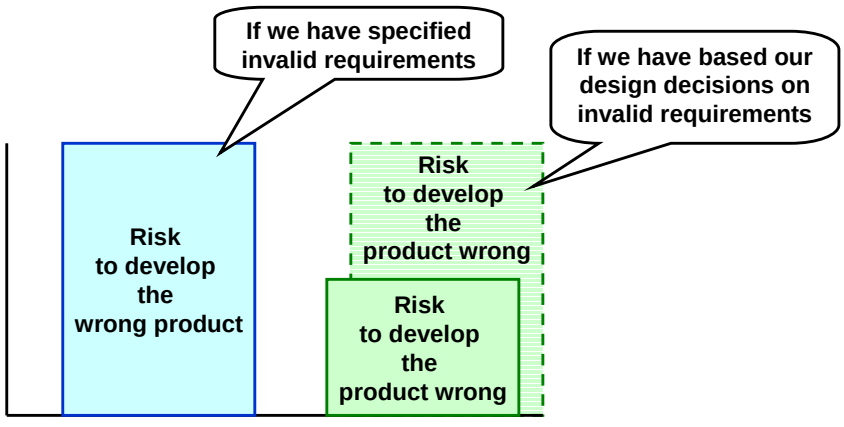
”At the end of the project you will get no more than you have specified at the beginning – anything else will cost extra time and extra money!”

What would you do at the beginning of the project?

- Right - you will specify a little bit more – to be on the safe side and leave no doubt!

The Chaos Report suggests that most projects can be finalized with about 60% – 70% of the initially defined functionality!

Resulting risk characteristic of the sequential approach



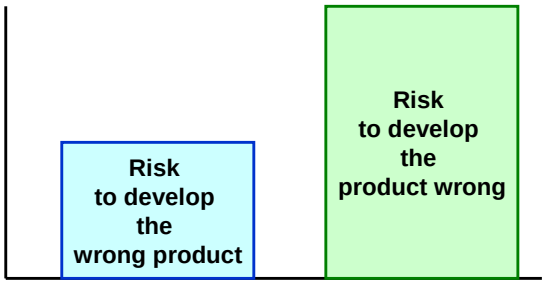
“Decide late - try out early”
Some Consequences

- Design and implementation are based on incomplete or fragmentary requirements.
- We must dramatically reduce the cycle time between
  - definition of functionality and
  - getting feedback by testable or applicable software.
- Requirements lose their role as stabilizing fixed point for project management.
- Testing becomes a control process!
  - Test results govern the planning for the next iteration.

Development
based on incomplete requirements

We make architectural decisions and start implementing based on incomplete / fragmentary requirements.

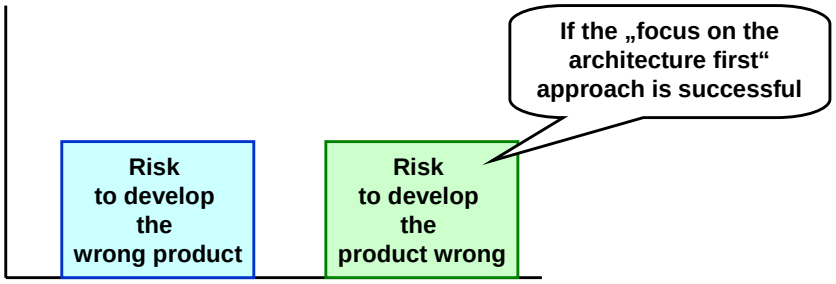
This in the first cut increases the risk of wrong design decisions.



Initial risk characteristic of the iterative approach

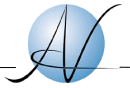
“Focus on the architecture first”

“Focus on requirements first” in the sequential process
is replaced by
“Focus on the architecture first” in the iterative process!



Resulting risk characteristic of the iterative approach

Reduced cycle times for early feedback



How can we reduce cycle times?

- Working faster - Not the solution!
- Making shortcuts in the sequential process - Not the solution!
- Reducing complexity - A reasonable way!

Split the development in smaller pieces:

Analyze a little – design a little – code a little – test a little

(Grady Booch has visionary figured out this concept in the early 80ies)

Planning without frozen requirements

- „Frozen requirements“ are replaced by a „projects vision“:
  - It figures out what the customer wants to achieve.
  - It is a representation of the software´s business value.
  - It justifies the invested budget.
  - It is the only medium- and long-term guideline for the evolution of the system.
- A detailed project plan is replaced by:
  - A consequent risk management process.
  - Planning discipline: Fixed time frame for each iteration.
  - A strict prioritization of desired SW features.

What is different?

Vision:

- Every developer has to care about the vision!

  (Having detailed requirements many SW developers dont`t care about the customer`s vision and business value.)

Planning:

- Sequential process:
  - Fixed target: The requirements (→ functionality)
  - Dependent variable: Time and resources
- Iterative process:
  - Fixed target: Timeframe and resources
  - Dependent variable: Implemented features

Top Ten Principles of modern (iterative) Software Management

1. **Focus the process on the architecture first**
2. **Attack risks early with an iterative lifecycle**
3. **Emphasize component based development**
4. **Establish a change management environment**
5. Enhance change freedom with tools for round-trip engineering
6. Use rigorously a model-based design notation
7. Instrument the process for objective quality control
8. Use demonstration-based assessment of intermediate artifacts
9. Plan releases with evolving level of detail
10. Establish a scaleable, configurable process

Rational Software White Paper 1998: Best Practices for SW Development Teams

Conclusions

Iterative SW development is NOT
a less disciplined variant
of the sequential/incremental SW development approach.

Iterative SW development is following
a different risk strategy.

Iterative SW development
is knowingly shifting risk.

Shifting the risk

The iterative approach has additional risks, e.g.:

- No fixed requirements.
- Problems with fix price tenders.
- Uncertainties regarding estimations and planning, …

The iterative approach can eliminate / mitigate other risks which we have accepted as part of the game, e.g.:

- Customer's uncertainties about required functionality.
- Long elapse time between project start and initial operation.
- Moving target requirements caused by a dynamic evolving business, …

Choosing the right approach is essential

Sequential or iterative SW development governs the basic risks:

- The risk to develop the wrong product.
- The risk to develop the product wrong.

Both approaches may lead you either to low or to high risks, depending on your circumstances:

- An inadequate SW development approach for your situation may lead to high risks in both dimensions.
- An adequate SW development approach for your situation may lead to low risks in both dimensions.

Caution – Avoid the Traps

- Sticking to sequential thinking and trying iterative doing
  - eliminates the stabilizing elements of the sequential process,
  - without having the benefits of the iterative process.
- The iterative approach without
  - clear vision, disciplined iteration planning and time boxing,
  - easily ends up with chaos.
- Picking the best of both worlds
  - easily leads into a process without any risk strategy,
  - you will take the risks of both sides.

“You may forget some critical factors – but they won’t forget you!”

Tom Gilb

The right process for the right project

The **sequential SW-development** approach
- is considered optimal on projects, in which
- clear stated requirements or specific boundary conditions

favor a **highly plan driven** (predictive) **approach.**

The **iterative SW-development** approach
- is considered optimal on projects, in which
- there is enough uncertainty that exploration and progressive understanding of requirements

favor a **highly adaptive approach.**

Some arguments

Very big systems & long term projects:

- An iterative approach can reduce complexity and speed up your project.
- That can help you to bring essential aspects of your system operational (→return on investment) before a change in the basic conditions may wash your project overboard.

Small projects:

- An iterative/agile approach may help you to eliminate (documentation and support) overhead, which has made troubles in the past and
- may speed up your project without increasing your risks.

Some indicators (1)

Is the project bound to a public invitation to tender?

- Yes → No chance for an iterative process!
- No → An iterative process may be taken into account!

Do you have an internal customer?

- Yes → The iterative approach may close the gap between theory & practice you may have had in the past.
- No → The iterative approach requires intensive customer cooperation.

Some indicators (2)

Do you trust that the customer knows pretty well what he needs?

- Yes → A strong indicator for the sequential approach.
- No → The iterative approach could help to handle the problem.

Do you trust in the project team's competence in the application field?

- Yes → The sequential approach may have a good chance to succeed.
- No → The iterative approach could mitigate your problem.

Some indicators (3)

Do you trust, that your customer will accompany the project with competent people to give qualified feedback?

- Yes → Good precondition for iterative SW development.
- No → Precise requirements at the beginning could mitigate your problem.

Are the members of the SW development team interested in the customer's vision?

- Yes → Good precondition for any SW development approach.
- No → Precise requirements at the beginning could mitigate your problem.

Résumé

Keep an eye

- on the situations in which your team succeeds
- and on your troubles.

Consider: Which risk strategy could improve your performance?

- “Freeze requirements first” or “decide late - try out early”?

Make a clear decision,

- keep in mind the implications and
- go the way consequently!

Thank you for your attention!

Questions & Discussion