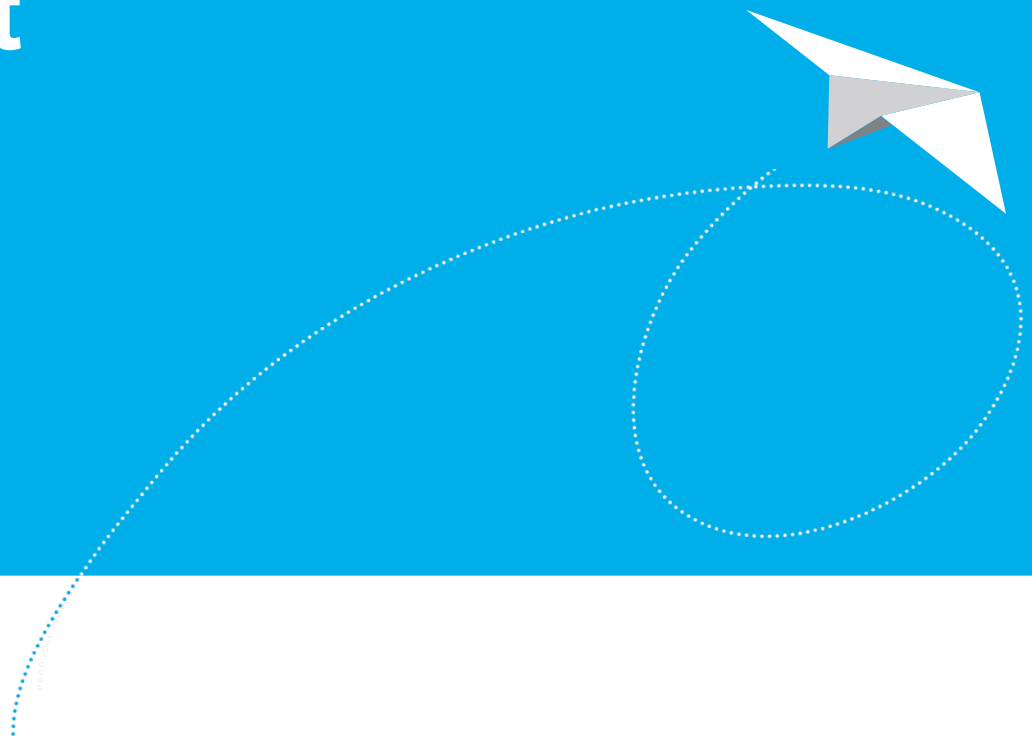
blue yonder

Forward looking. Forward thinking.

Agile Software Development

Dr. Manuel Bähr



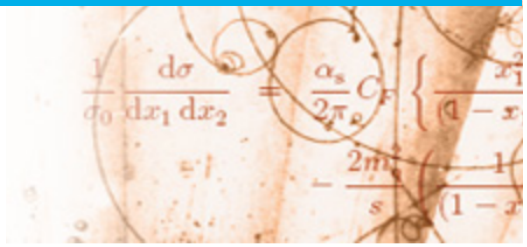
blue yonder

Forward looking. Forward thinking.

Software engineering – From academia to industry

Dr. Manuel Bähr

I came the same way



- Ph.D. in physics
- Team Leader Technology Development at Blue Yonder

blue yonder

Blue Yonder – forward looking, forward thinking

Started as a spin-off from the University of Karlsruhe, Germany.

Founded by Prof. Michael Feindt.

Initially: Prediction of particle properties.

Now about 100 employees of which ~60 have a PhD, mainly from physics.

3 Offices:

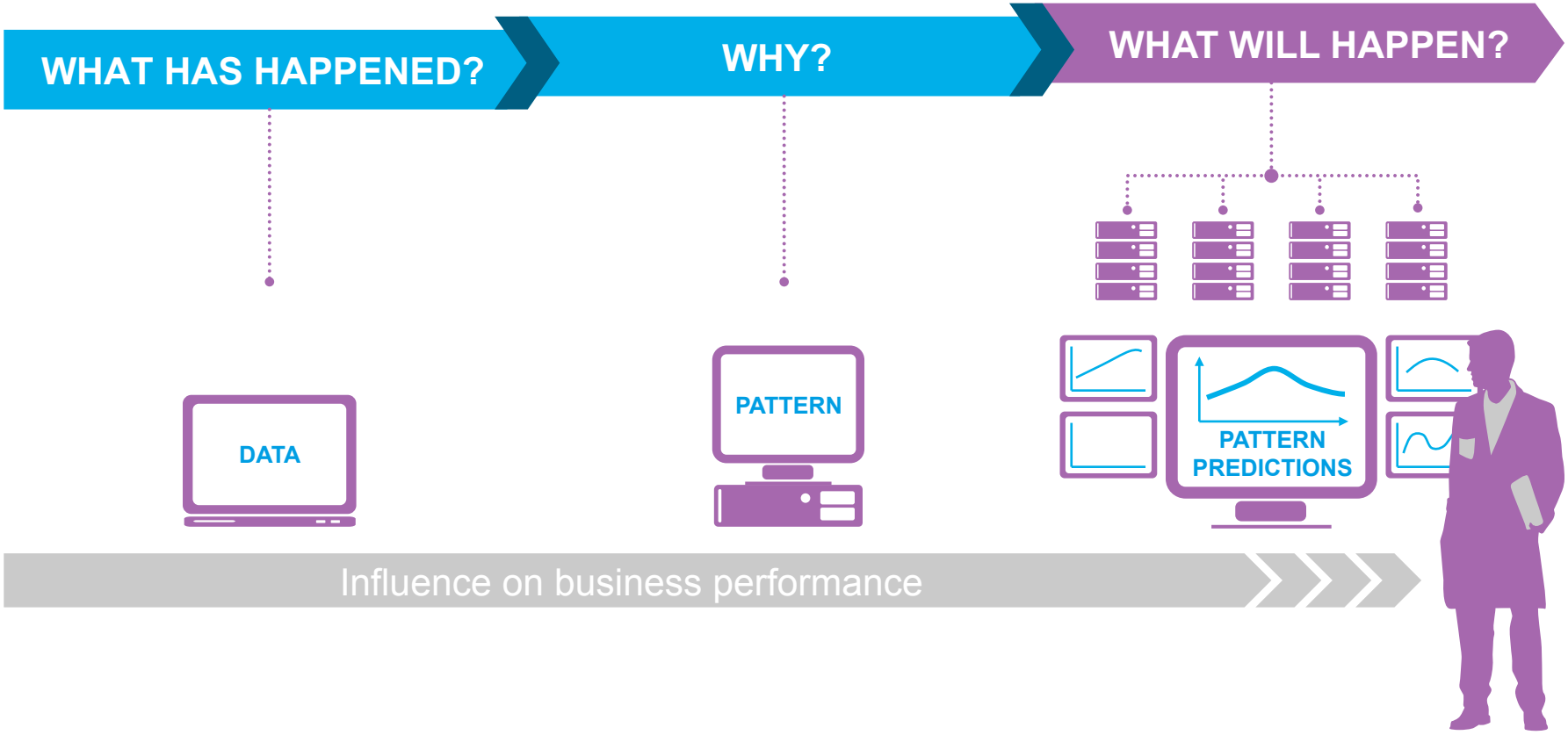
Karlsruhe, Hamburg (Germany)

London (UK)

What do we need for perfect decisions?

Data Mining and conventional Business Intelligence

Predictive Analytics



blue yonder

Decisions at scale from blue yonder

moving billions in value for our clients

Replenishment for a grocery chain (24/7 SaaS operations)	Automation increased from 61% to 95%
Supply chain predictions (24/7 SaaS operations)	**> 620,000,000** predictions every day
Dynamic pricing for a major online shop	10% revenue increase after 4 weeks
Customer life cycle management	6% revenue increase within 3 months

blue yonder

Forward looking. Forward thinking.

What is necessary to be sucessful?

blue yonder

Forward looking. Forward thinking.

+ Build the right thing

+ Build the thing right

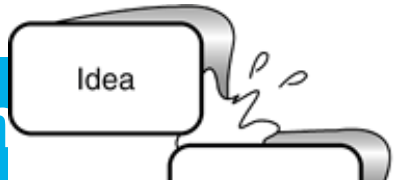
blue yonder

Forward looking. Forward thinking.

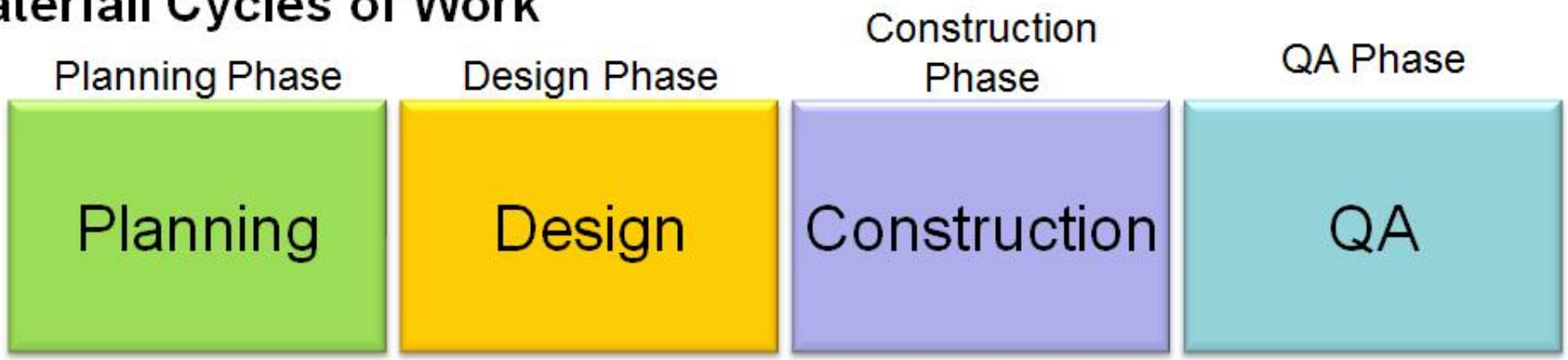
How to build the right thing?

Learn about TPS, Agile Manifesto, Lean Startup, Scrum

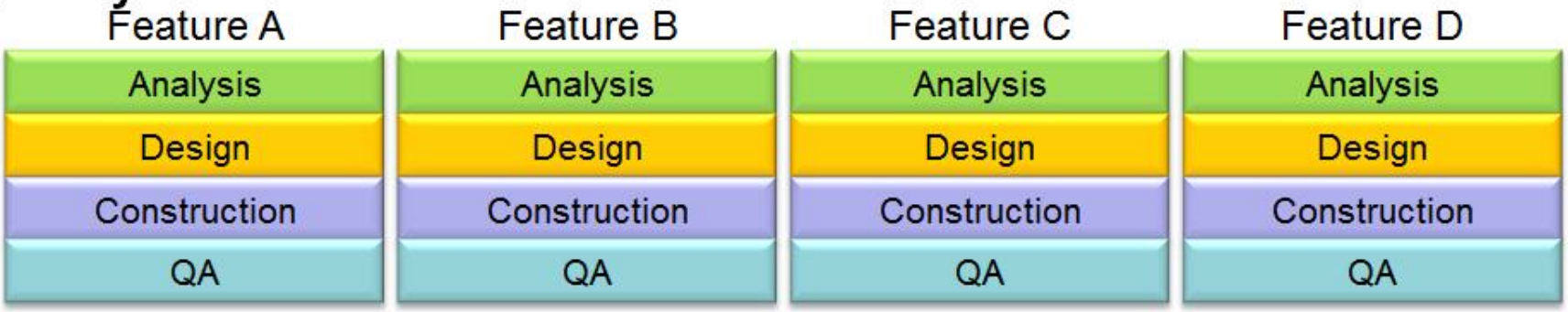
Software Developmen



Waterfall Cycles of Work



Agile Cycles of Work



taken from http://www.bigvisible.com

blue yonder

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.

Lean Software Development

Application of the „Toyota Production System“ (TPS 1948-1975) to software development. Key principles:

- In a process of constant improvement, **eliminate**
  - Waste (anything that is not adding value to the customer)
  - Variation & overload (workload of employees)
- Respect for people

Ways to eliminate waste

- Decide late
  - Requirements change
- Deliver fast
  - Feedback on implementation
- Build quality in
  - Defects are waste in the first place

Prevent overload and variation

- Pull instead of push process
- Minimize „work in progress“
- Never change scope of „work in progress“

Scrum

- Example of an agile software development method that also implements lean principles
- Based on Takeuchi & Nonaka (1986) „New New product development game“. First publication of „Scrum“ by Ken Schwaber and Jeff Sutherland in 1995.
- Other agile methods include: Kanban, Extreme Programming, Crystal Clear etc.

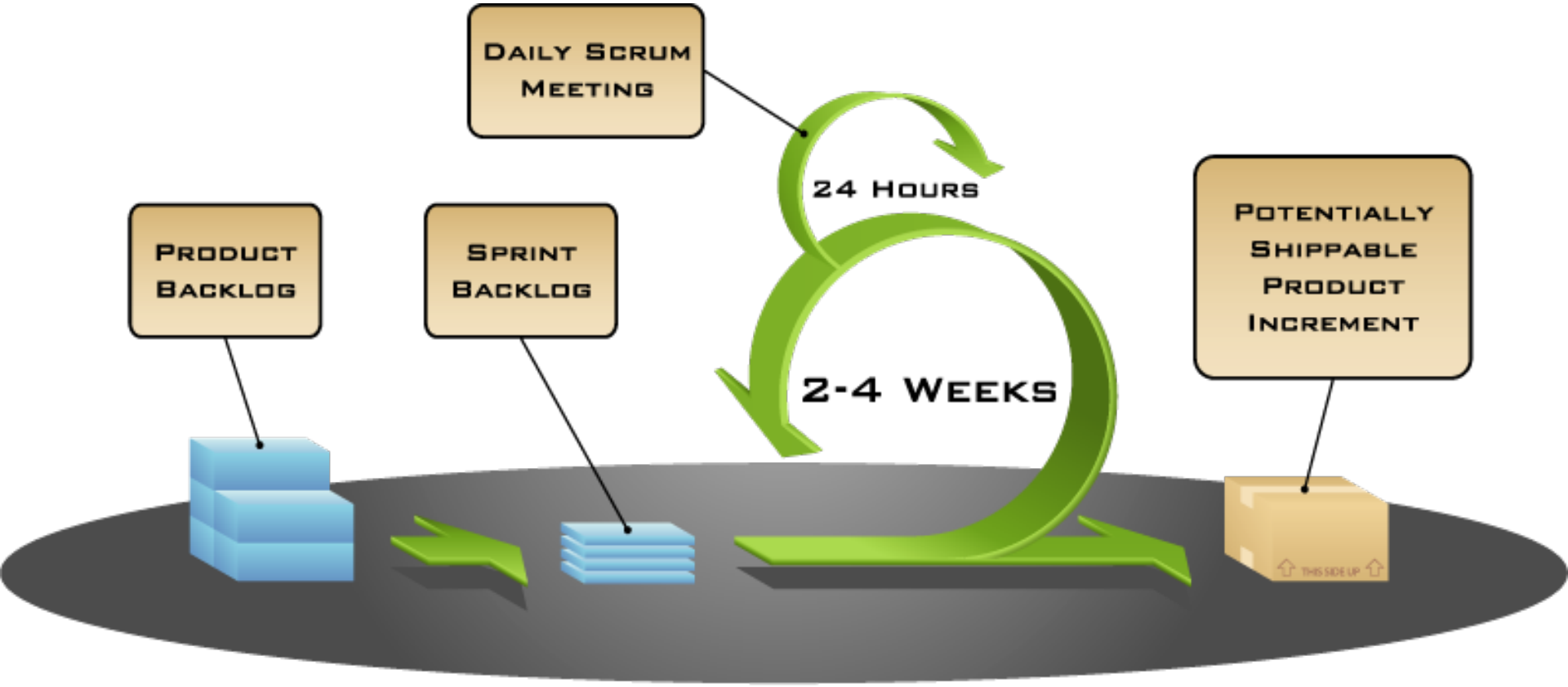
That‘s a scrum!

blue yonder

Basic ideas

- Target system has a complexity which makes a detailed planning ineffective or impossible
- Foster self-organisation of a development team
- Use time boxed iterations to deliver small **fully working increments** of the final product

Basic ideas



blue yonder

Scrum Roles

- **Team**
  Turns **user stories** into shippable pieces of software

- **Product Owner**
  Responsible for maximising Return on Investment

- **Scrum Master**
  Supports the team in removing impediments and applying the Scrum methodology

User story

- As a <user>, I want <feature>, so that <reason>
- As an admin, I want to set internal parameters, so that I can adapt the software to our specific hardware setup.
- As a scientist, I want to do simulations, so that I can publish articles.

Scrum project in a nutshell

- Agree on Goal of the project
- Agree on Definition of Done for user stories (features)

- Collect user stories in a prioritized list (product backlog)
- Team estimates complexity in unit-less quantity (story points)

- Team selects N top user stories to be done in one Sprint
- Team deduces tasks for selected stories
- Team designs, implements, tests and documents

Scrum project in a nutshell

- Team synchronizes at a daily stand-up meeting (the **Scrum**) - limited to 15min
- No scope change during sprint time

- Product Owner accepts fully done stories at sprint end
- **Measure** story points per Sprint
- estimated delivery date for a fixed scope can be calculated
- start next iteration, improve process if possible

Benefits

- Focus, rhythm, clear goals -> necessary for flow
- Strong team bonds – team commitments, team estimates
- Clear separation of „What“ and „How“
- Focus on customer value
- All benefits from lean principles

blue yonder

Forward looking. Forward thinking.

Extending lean ideas

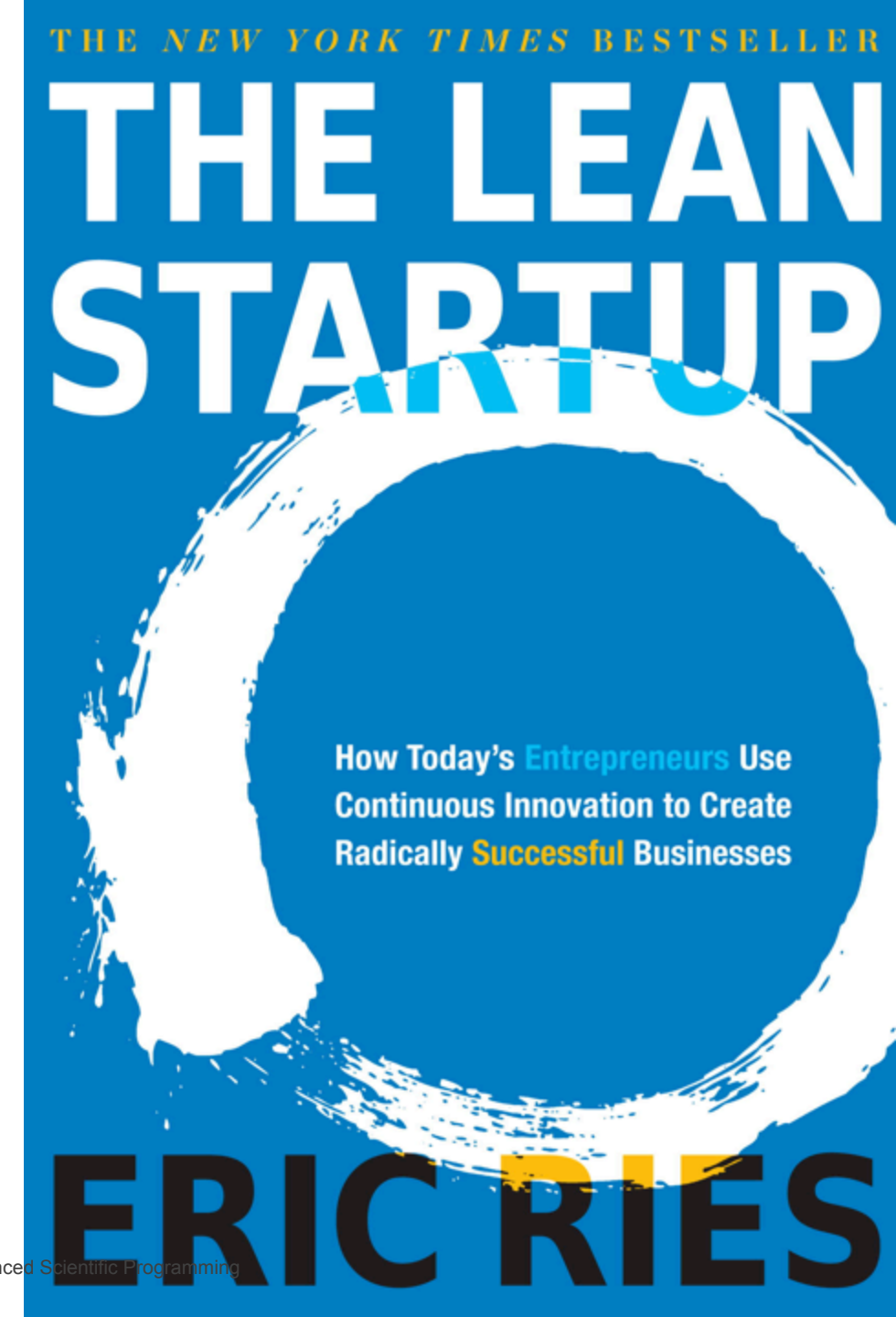
blue yonder

Forward looking. Forward thinking.

Lean Startup movement: Reduce failure rate for startups

A startup is a human institution designed to deliver a new product or service under conditions of extreme uncertainty.

- Sole purpose of the startup: Learn how to build sustainable business
- Your product is your business model
- formulate hypothesis and run **experiments**
- Learn as fast as possible



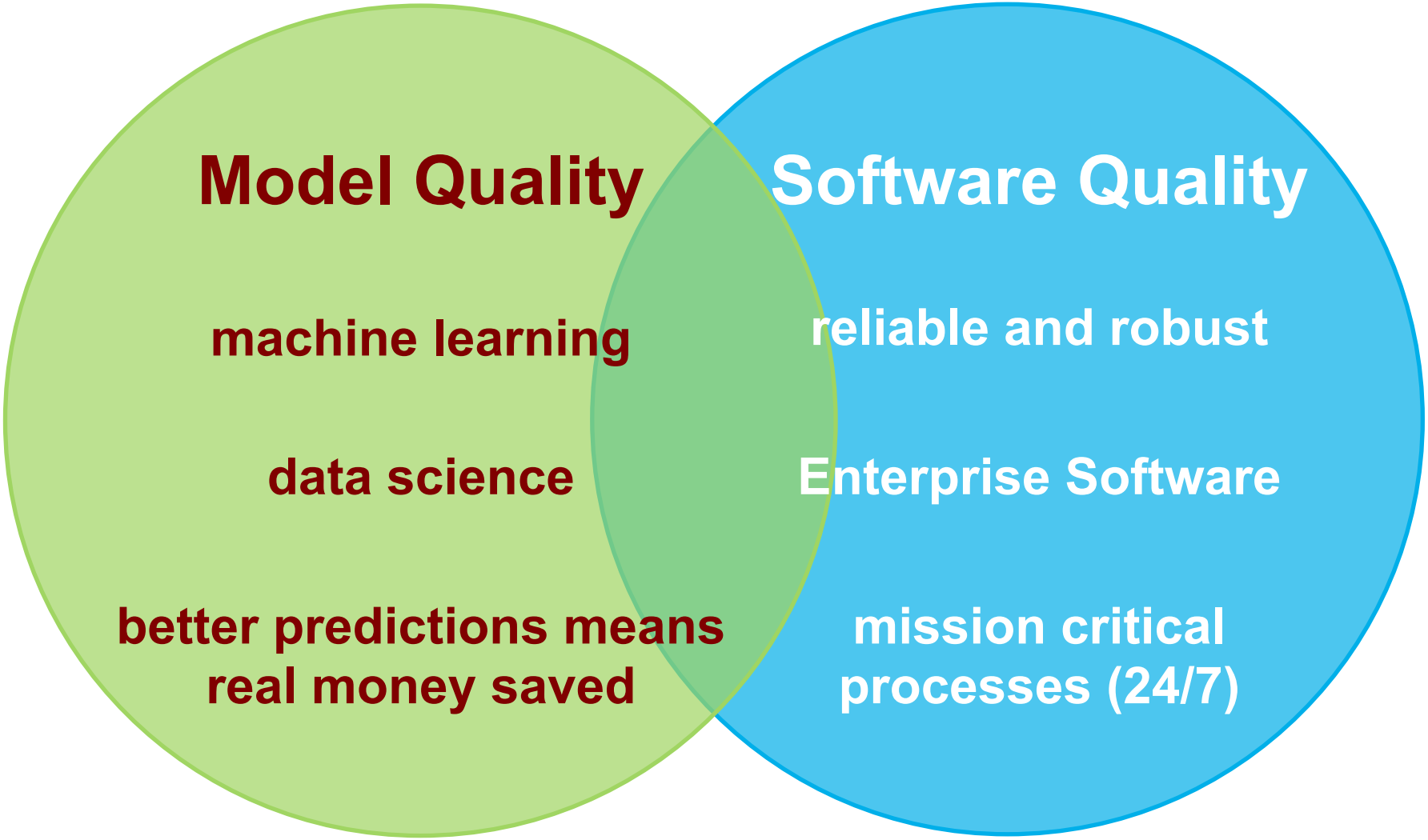
blue yonder

Forward looking. Forward thinking.

How to build the thing right

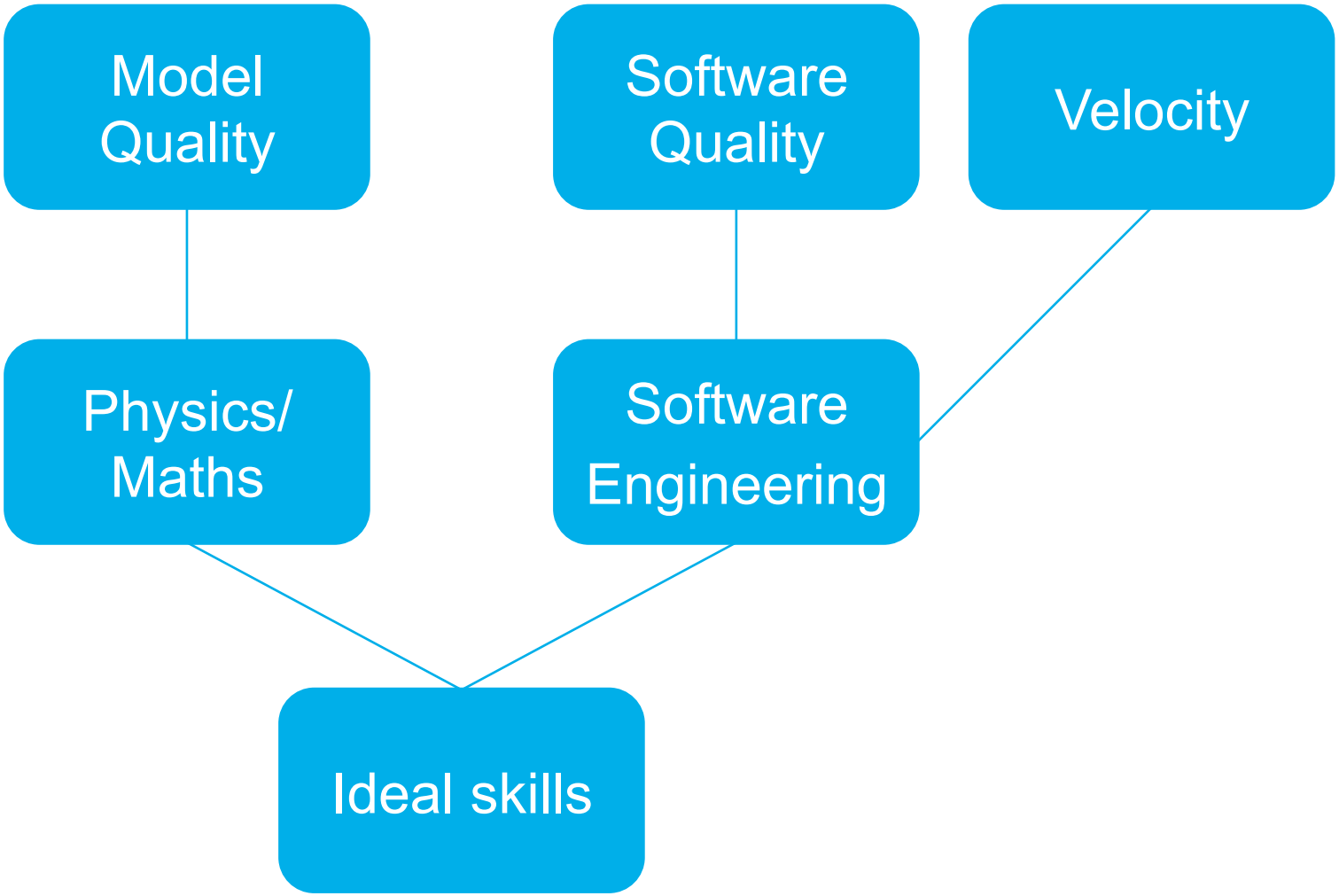
Learn about TDD, CleanCode, CI

Our unique selling point



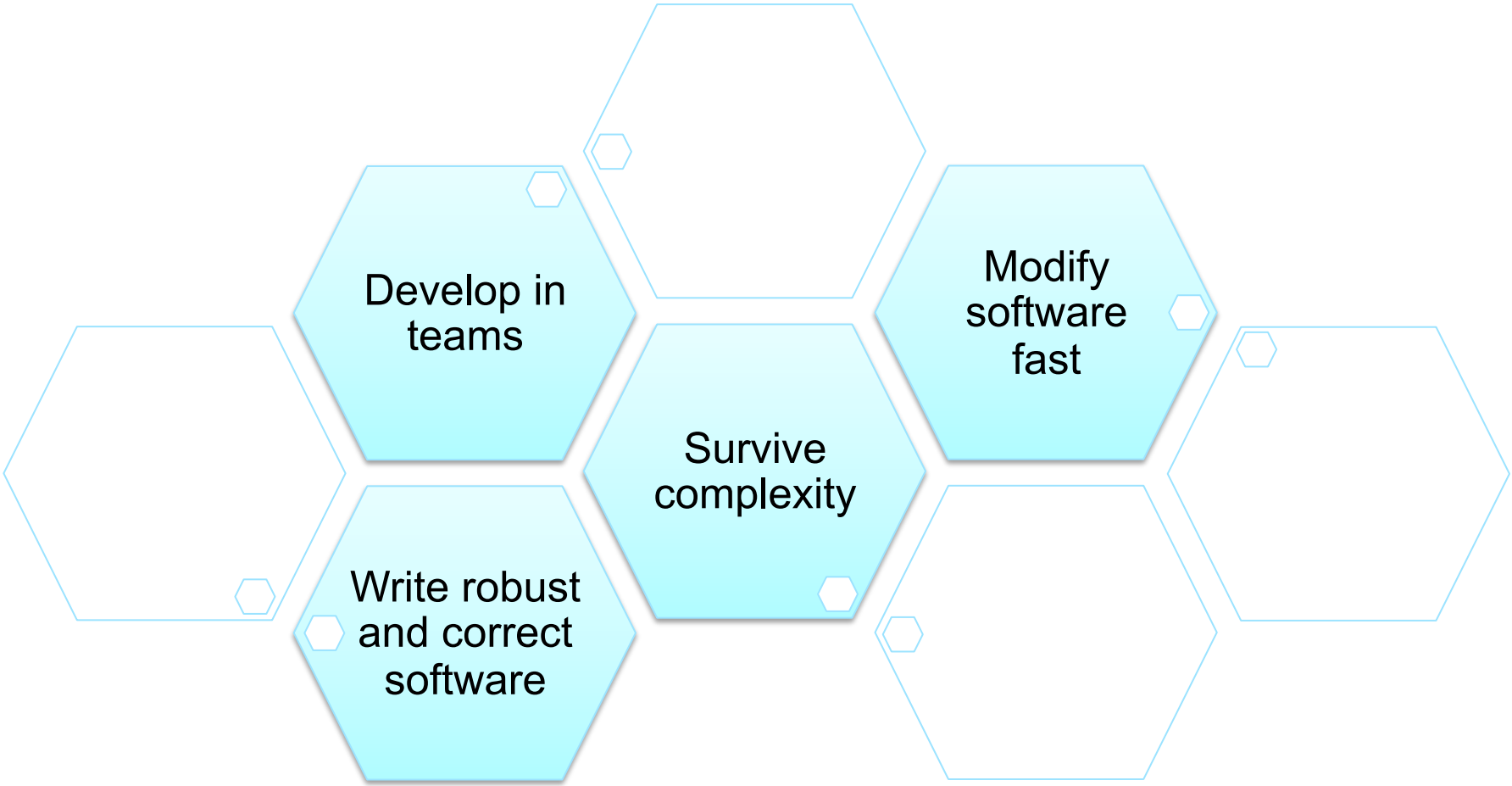
blue yonder

Skill check



blue yonder

Aspects of software engineering



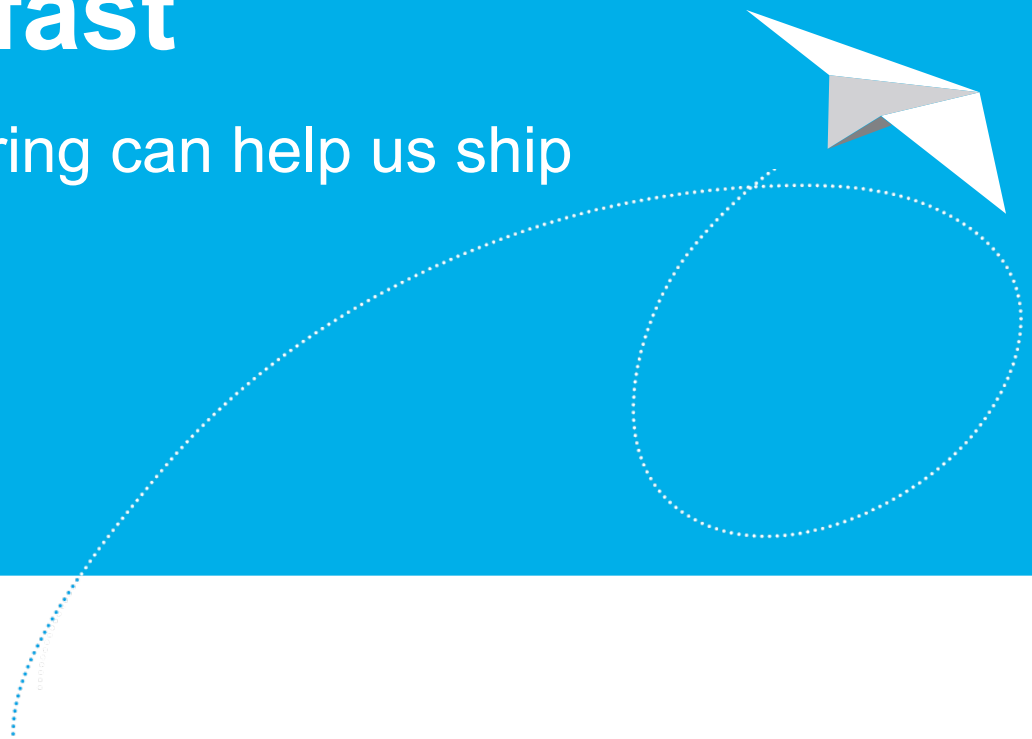
blue yonder

blue yonder

Forward looking. Forward thinking.

We want to get things done fast

How software engineering can help us ship **high quality** fast.



What do you spent time on during development?

Planning	Changing	Documenting	Thinking
Bugfixing	Testing	Coding	Reading
Reviewing	Discussing	Learning	...

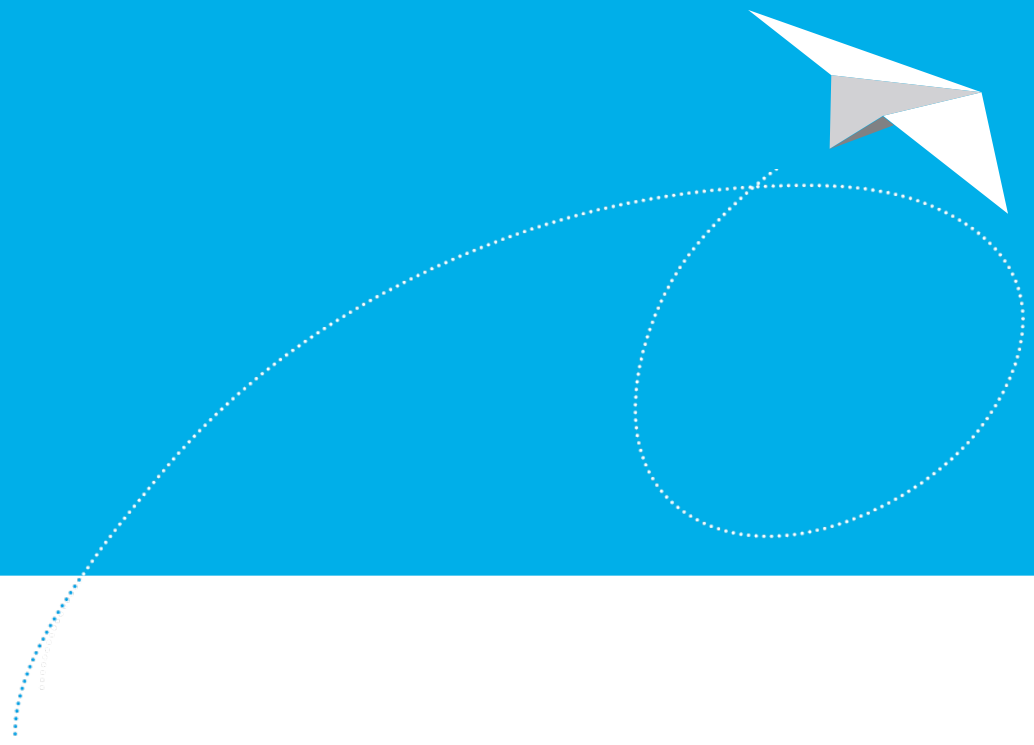
What do you spent time on during development?

Planning	Changing	Documenting	Thinking
Bugfixing	Testing	Coding	Reading
Reviewing	Discussing	Learning	...

blue yonder

Forward looking. Forward thinking.

cheap reads
cheap writes
no defects

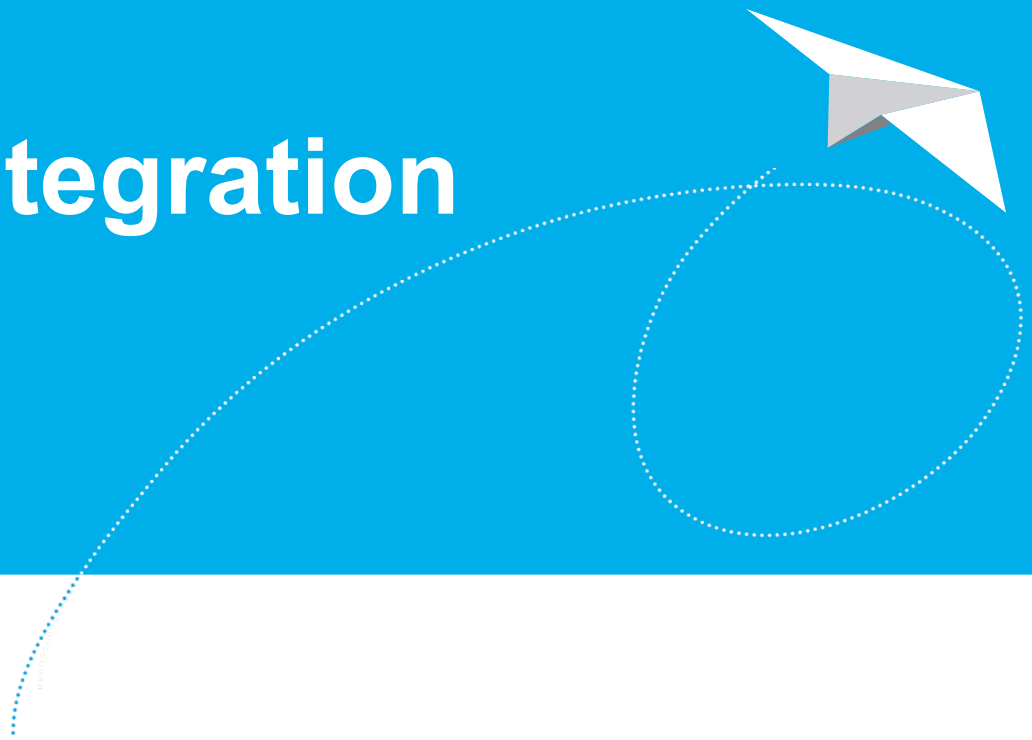


blue yonder

Forward looking. Forward thinking.

Getting fast with

Continous Integration



Continuous Integration / Deployment

- Maintain a single source repository
- Automate building
- Automate running your tests (code coverage!)
- Each commit builds and tests on an integration machine
- Automate deployment, so that everyone can get the latest piece of software

Jenkins

- https://travis-ci.org
- http://jenkins-ci.org

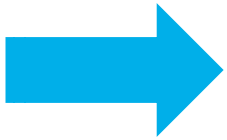
Travis

And many more ...

blue yonder

If something hurts – do it more often!

- Verify correct interaction of all affected modules, subsystems ...
- Eliminate „It works on my machine“
- Never have the same defect twice
- Find defects as quickly as possible after introduction into the codebase
- Eliminate manual processes



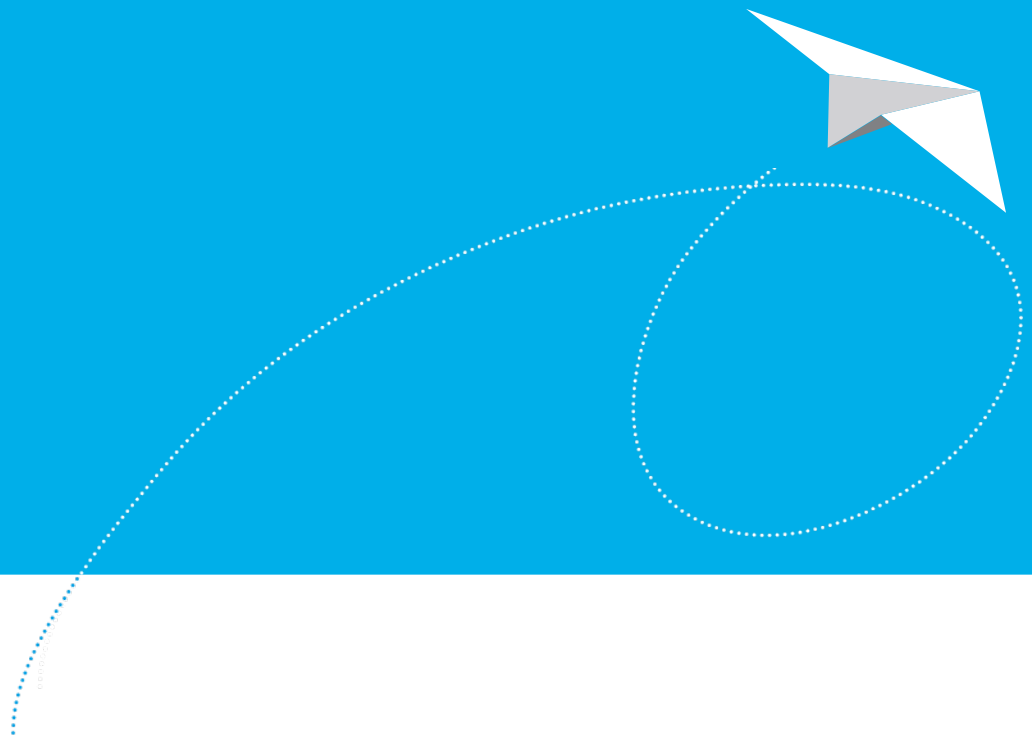
helps with „no defects“

blue yonder

Forward looking. Forward thinking.

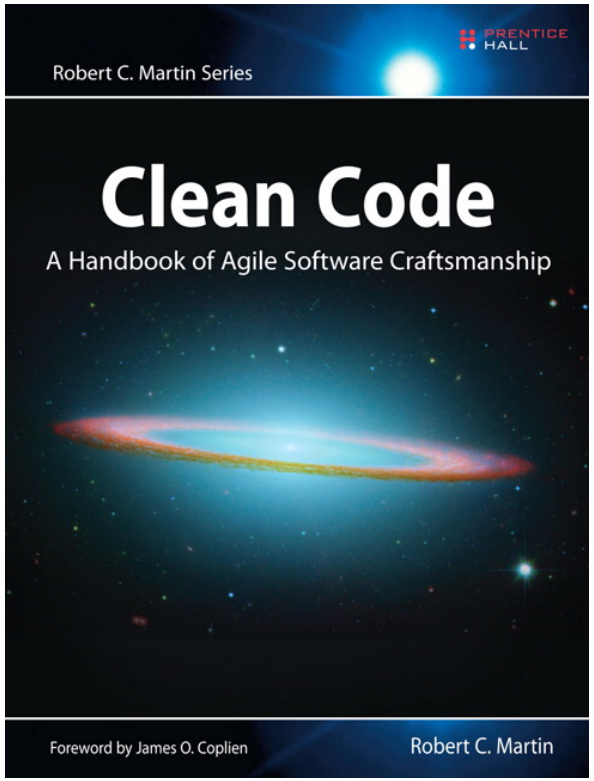
Getting fast with

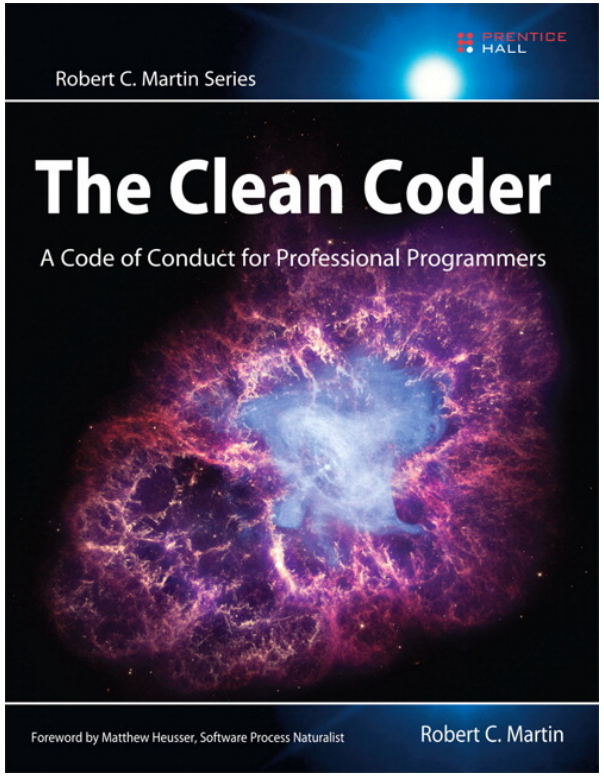
Clean Code



Codebases are like databases
read/write ratio 10:1

Recommended Reads





blue yonder

Getting started with Clean Code principles

- Keep it simple, stupid! (KISS) – readability
- Avoid (premature) optimization – readability
- Don‘t repeat yourself (DRY) – one place to change
- Single responsibility principle – one reason to change

- References:
  - www.clean-code-developer.de (sadly only in german)
  - www.clean-cpp.org (Clean Software Development with Modern C++)

Function length

6 lines ought to be enough for everybody

If a function does only one thing

Increased re-use

Better readability

Easier naming

Much easier exception safety

Better testability

Less side effects

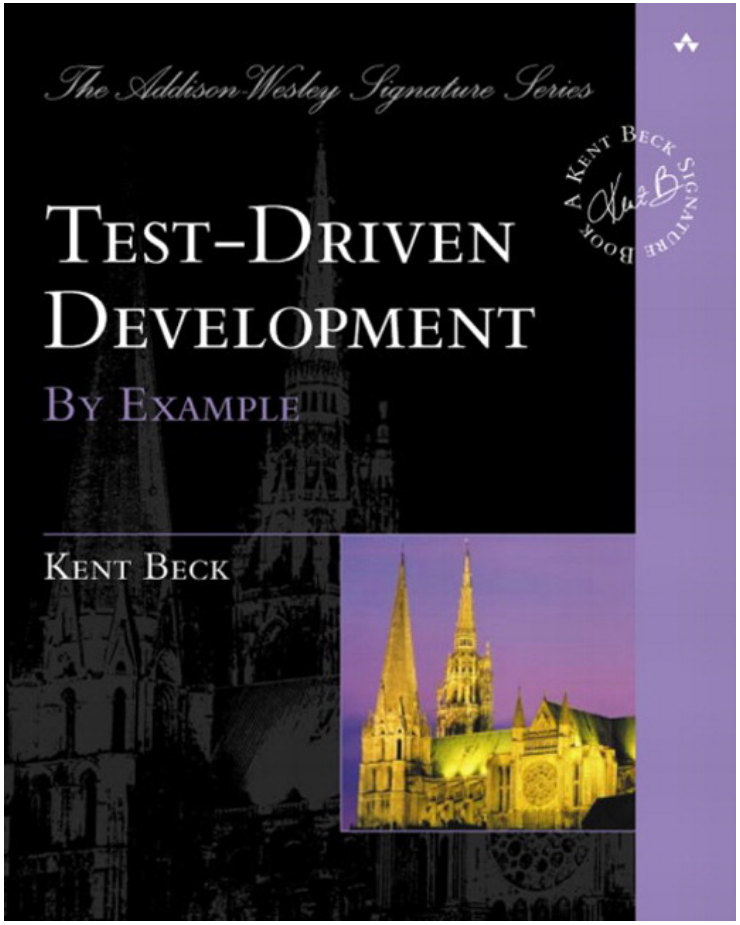
blue yonder

Forward looking. Forward thinking.

Getting fast with

Test-Driven Development

Recommended read



blue yonder

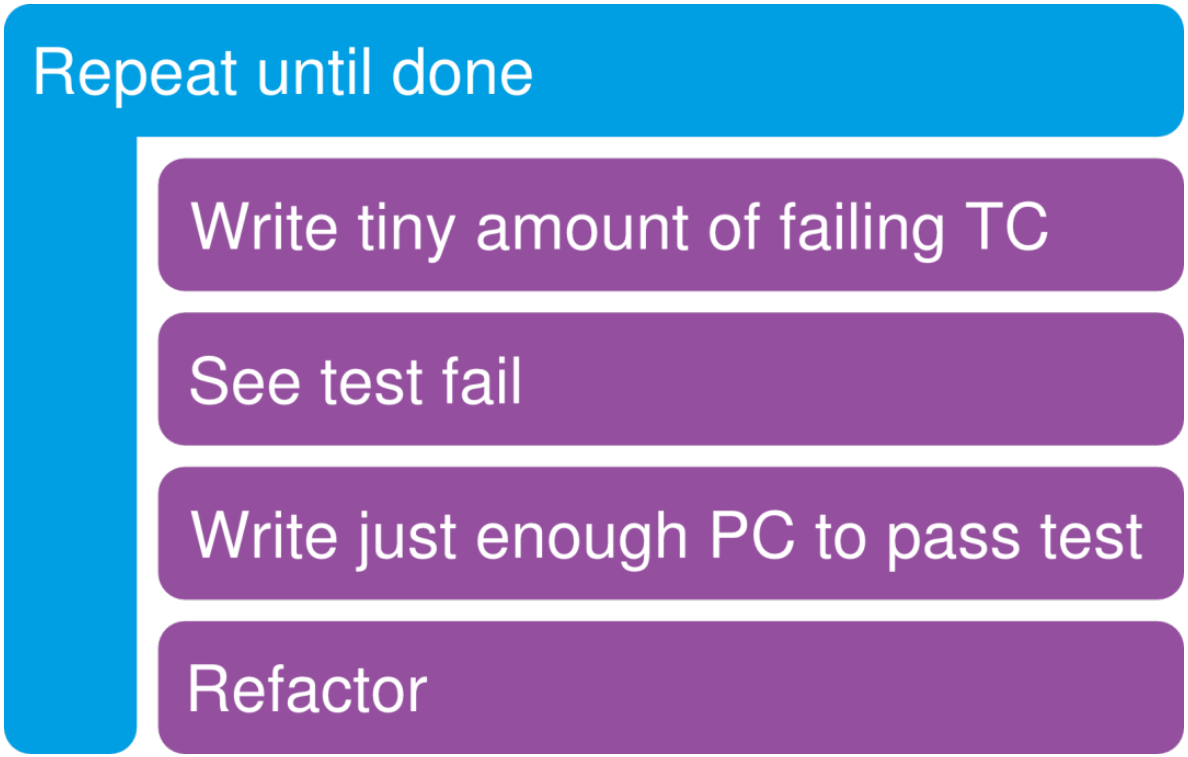
TDD as fundamental programming skill

“I taught Bethany, my oldest daughter, Test-Driven Development as her first programming style when she was about age 12. She thinks you can't type in code unless there is a broken test. The rest of us have to muddle through reminding ourselves to write the tests.“

Kent Beck, Test-Driven Development by Example,
Addison-Wesley Signature, 2002

Red – Green – Refactor

- Test code (TC) drives production code (PC):



blue yonder

Why TDD makes you fast

- Reduce time between defect introduction and discovery to the absolute minimum
- Ensure 100% code coverage
- Get the courage to change existing code (tests would break if you destroy something)
- Tested functions, classes, methods are easier to understand (they even have an executable specification)

blue yonder

Forward looking. Forward thinking.

Conclusions

Conclusions

- The software community (industry and open source) creates a lot of interesting concepts and techniques
- If you develop software regularly, follow these developments to **learn**
- Use them when they are helpful for you

blue yonder

Forward looking. Forward thinking.

Thank you!

For your attention.