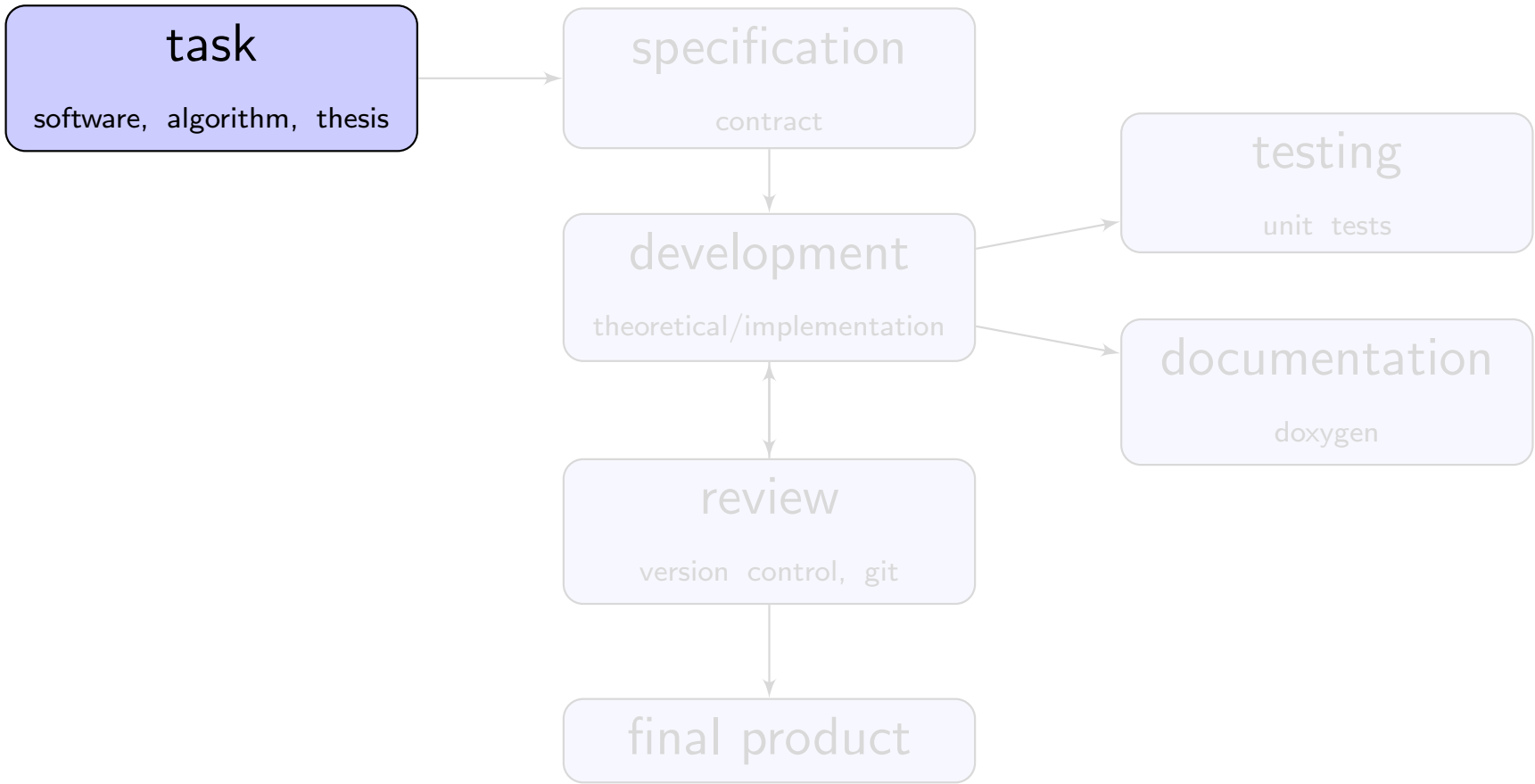
Principles of Software Development

Benjamin Unger

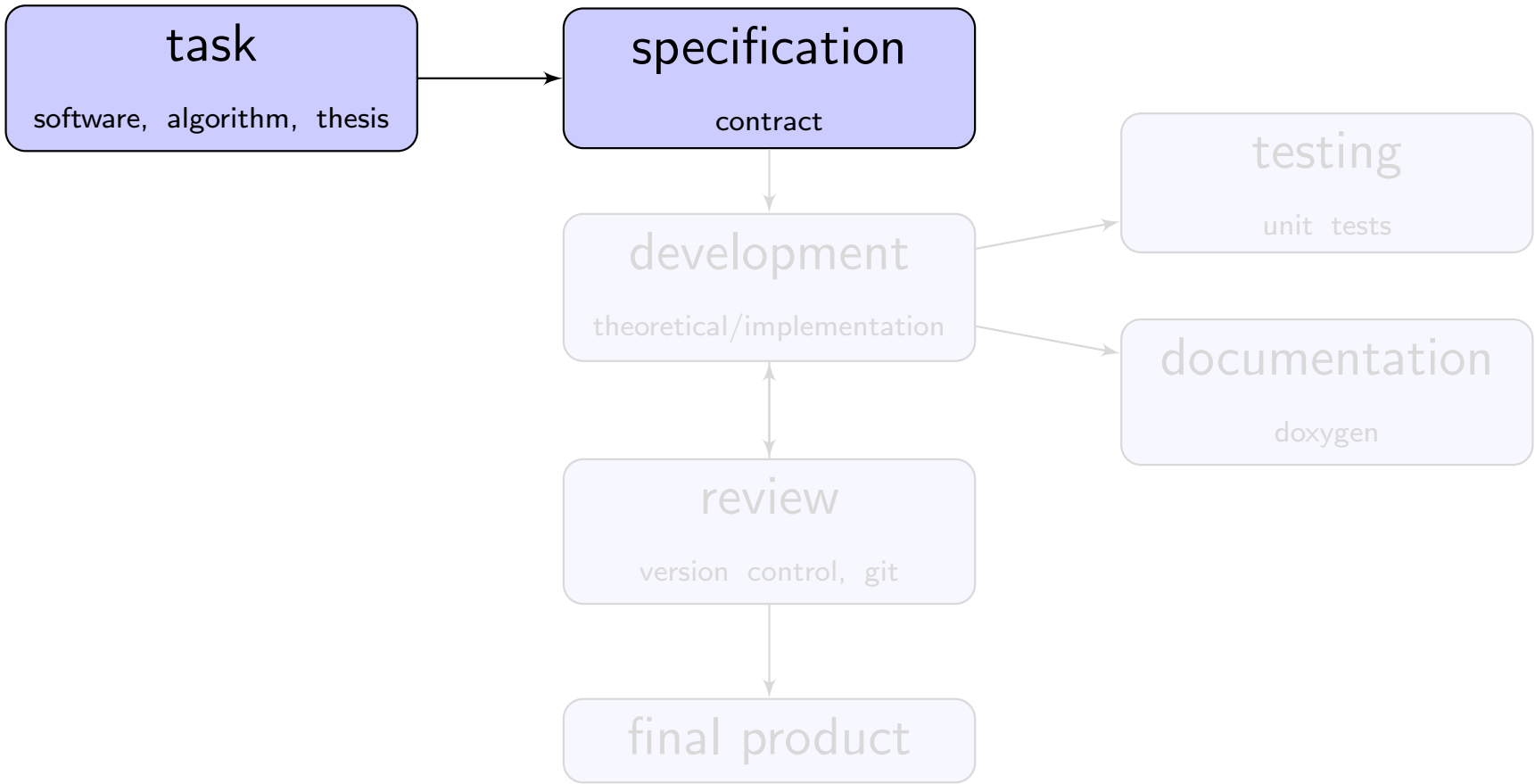
Tools Seminar - TU Berlin

November 17, 2014

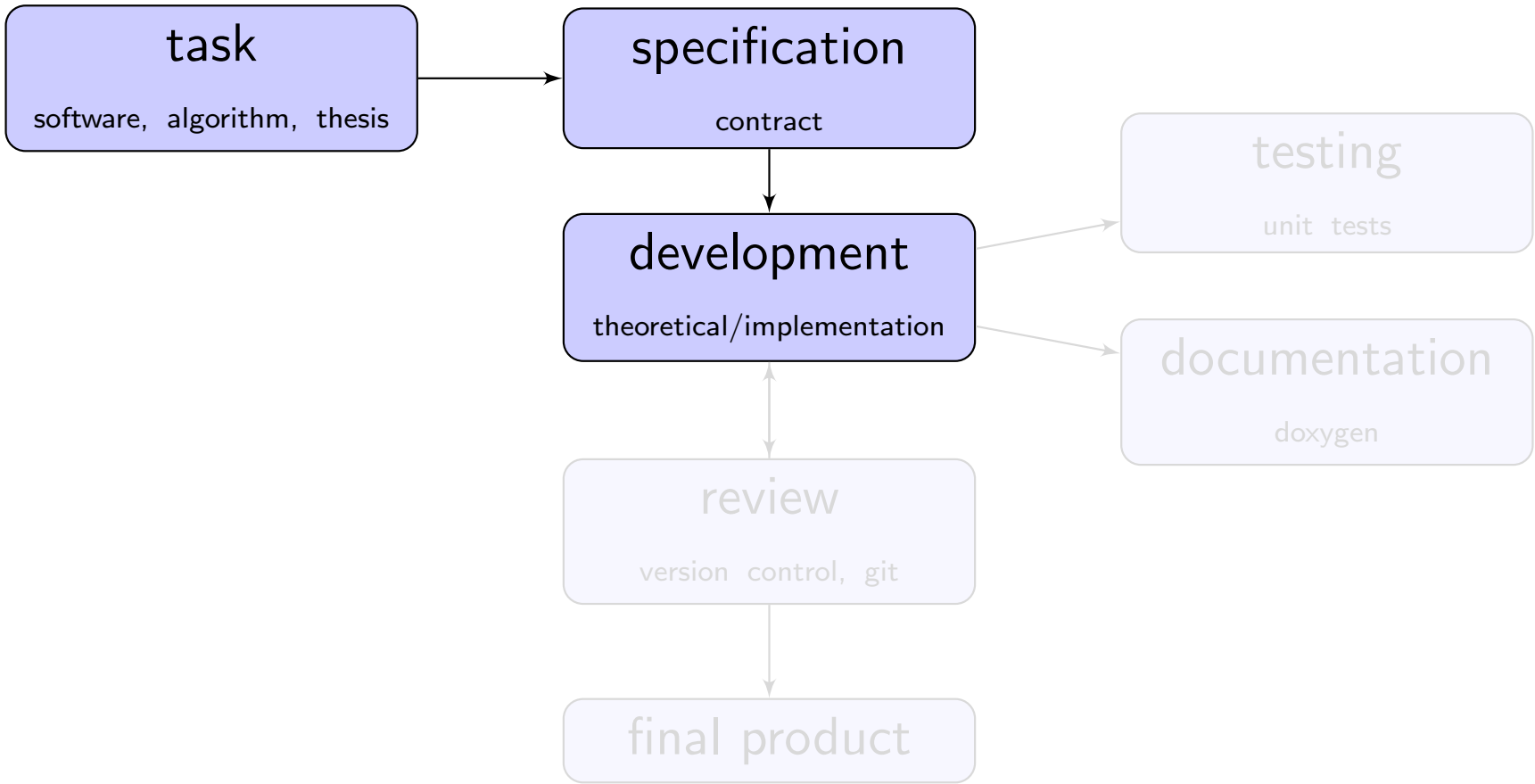
The Situation



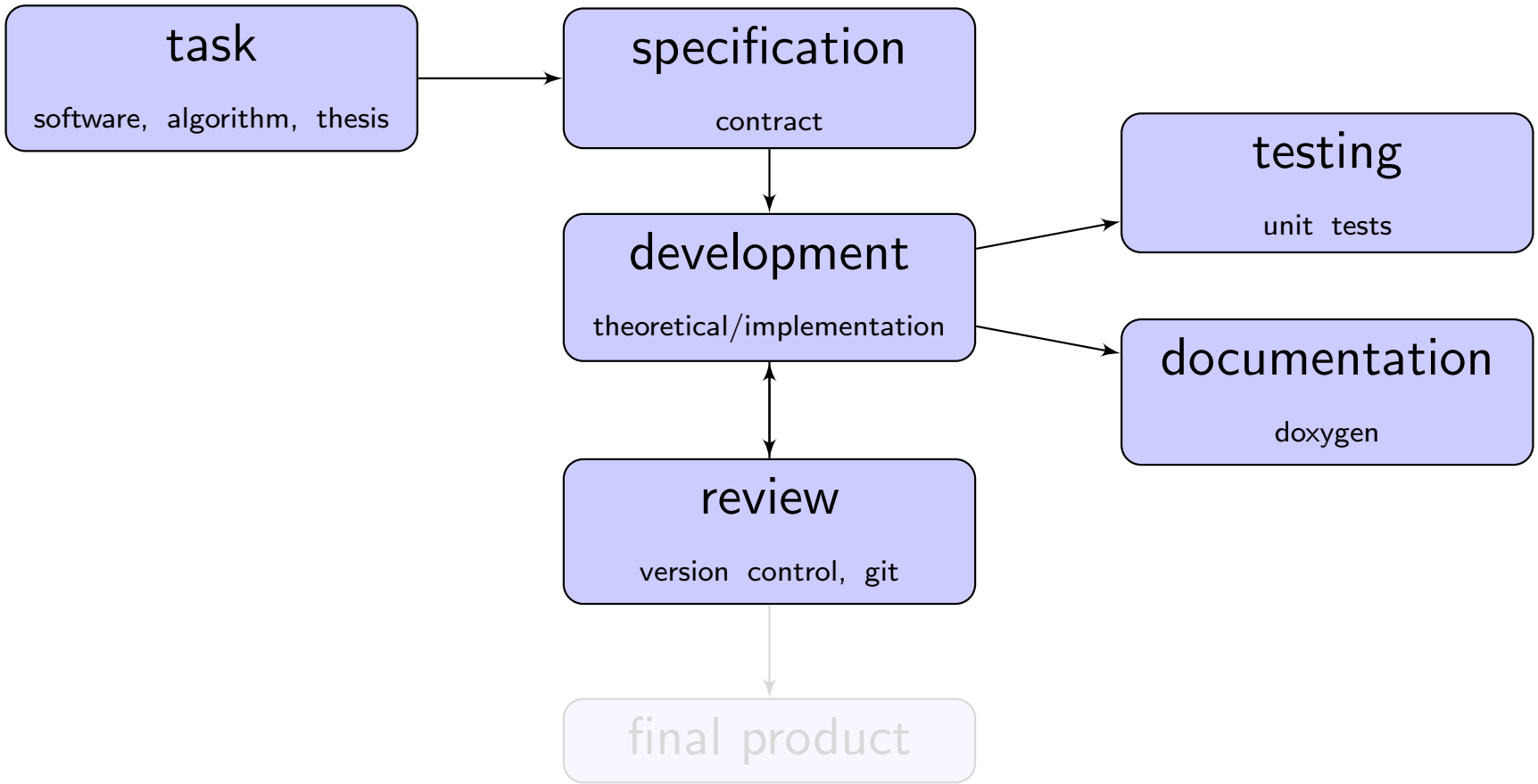
The Situation



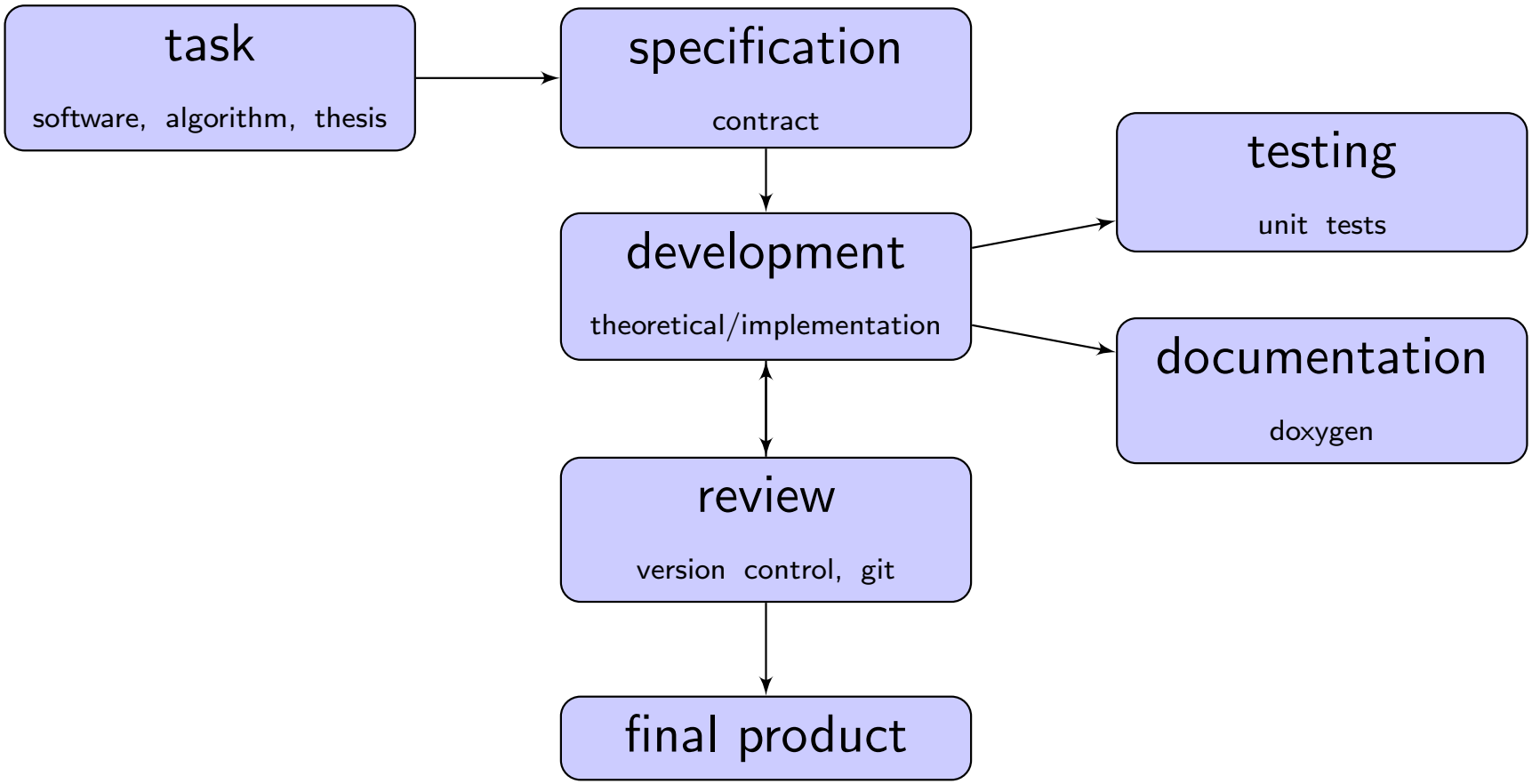
The Situation



The Situation



The Situation



User vs. Functional Specification

User Specification (DIN 69901-5 3.32) dt. Lastenheft

Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines (Projekt-)Auftrags.

Functional Specification (DIN 69901-5 3.40) dt. Pflichtenheft

Vom Auftragnehmer erarbeitete Realisierungsvorgaben auf der Basis des vom Auftraggeber vorgegebenen Lastenheftes.

User specification What does the costumer want?

Functional specification How will the supplier do this?

User vs. Functional Specification

User Specification (DIN 69901-5 3.32) dt. Lastenheft

Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines (Projekt-)Auftrags.

Functional Specification (DIN 69901-5 3.40) dt. Pflichtenheft

Vom Auftragnehmer erarbeitete Realisierungsvorgaben auf der Basis des vom Auftraggeber vorgegebenen Lastenheftes.

User specification What does the costumer want?

Functional specification How will the supplier do this?

User vs. Functional Specification

User Specification (DIN 69901-5 3.32) dt. Lastenheft

Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines (Projekt-)Auftrags.

Functional Specification (DIN 69901-5 3.40) dt. Pflichtenheft

Vom Auftragnehmer erarbeitete Realisierungsvorgaben auf der Basis des vom Auftraggeber vorgegebenen Lastenheftes.

User specification What does the costumer want?

Functional specification How will the supplier do this?

Example

User specification:

- $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^n$ given.
- Find $x \in \mathbb{R}^n$ such that $Ax = b$ for $m \leq n$ and

$$\|x\| \leq \|y\| \qquad \text{for all } y \text{ that solve } Ay = b.$$

- Find $x \in \mathbb{R}^n$ that minimizes

$$\|Ax - b\|^2$$

for $m > n$.

Example

Functional specification:

- $A \in \mathbb{R}^{m,n}$ sparse band matrix, $b \in \mathbb{R}^n$ given.
- Find $x \in \mathbb{R}^n$ such that $Ax = b$ for $m \leq n$ and

$$\|x\|_2 \leq \|y\|_2 \qquad \text{for all } y \text{ that solve } Ay = b.$$

- Find $x \in \mathbb{R}^n$ that minimizes

$$\|Ax - b\|_2^2$$

  for $m > n$.
- Not scalable for multicore (HPC).
- Computations satisfy a relative error $\leq 1e^{-6}$.

Software Design

Software Architecture

Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures.

Model View Controller (MVC)[1]: Separate objects by functionality

- **Model** A model represents knowledge in form of a single object or a structure of objects.
- **View** A view is a (visual) representation of its model that highlights or suppresses certain attributes.
- **Controller** A controller is the link between a user and the system and performs any data transformation (business logic).

[1] [Reenskaug,1979]

Software Design

Software Architecture

Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures.

Model View Controller (MVC)[1]: Separate objects by functionality

Model A model represents knowledge in form of a single object or a structure of objects.

View A view is a (visual) representation of its model that highlights or suppresses certain attributes.

Controller A controller is the link between a user and the system and performs any data transformation (business logic).

[1] [Reenskaug,1979]

Software Design

Software Architecture

Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures.

Model View Controller (MVC)[1]: Separate objects by functionality

Model A model represents knowledge in form of a single object or a structure of objects.

View A view is a (visual) representation of its model that highlights or suppresses certain attributes.

Controller A controller is the link between a user and the system and performs any data transformation (business logic).

[1] [Reenskaug,1979]

Software Design

Software Architecture

Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures.

Model View Controller (MVC)[1]: Separate objects by functionality

- **Model** A model represents knowledge in form of a single object or a structure of objects.
- **View** A view is a (visual) representation of its model that highlights or suppresses certain attributes.
- **Controller** A controller is the link between a user and the system and performs any data transformation (business logic).

[1] [Reenskaug,1979]

Software Design

Software Architecture

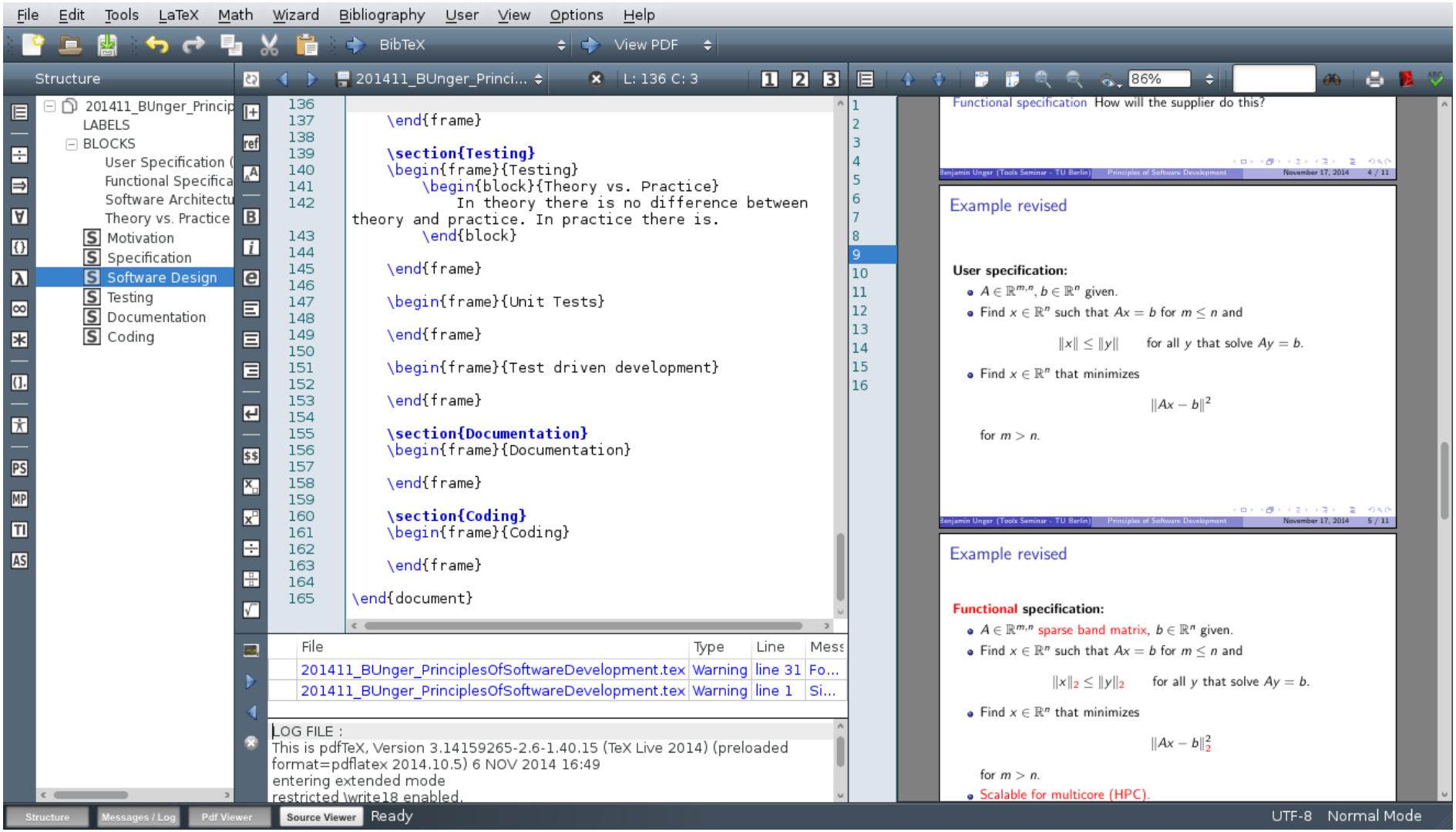
Software architecture is the high level structure of a software system, the discipline of creating such structures, and the documentation of these structures.

Model View Controller (MVC)[1]: Separate objects by functionality

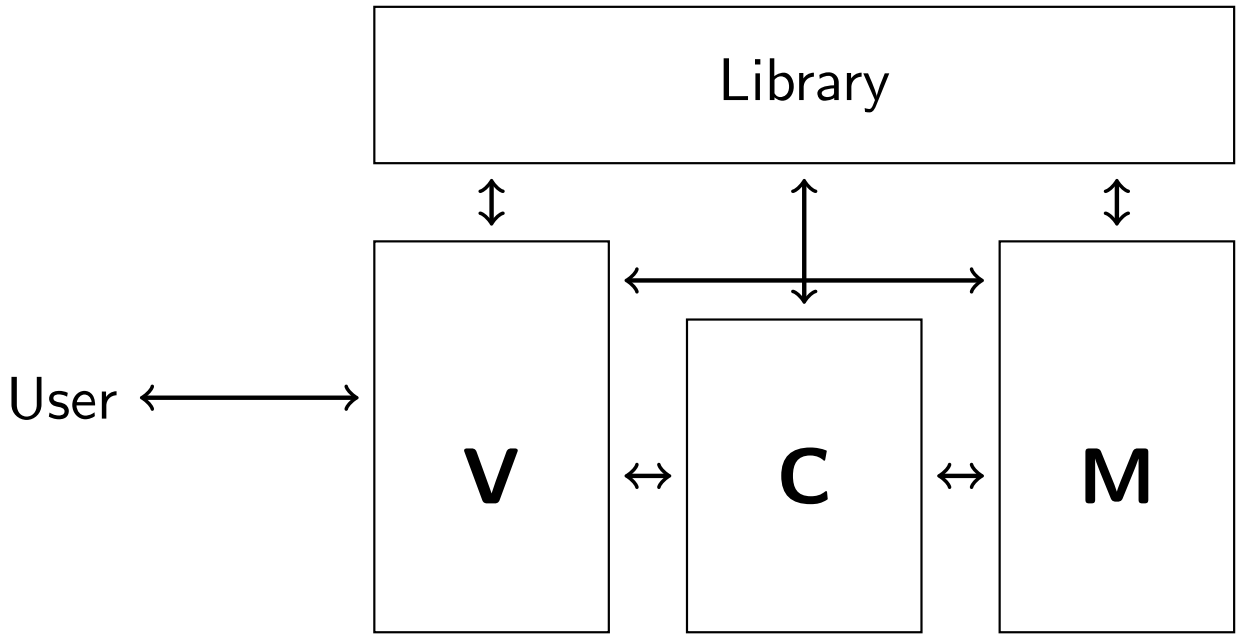
- **Model** A model represents knowledge in form of a single object or a structure of objects. Data only
- **View** A view is a (visual) representation of its model that highlights or suppresses certain attributes. User interface
- **Controller** A controller is the link between a user and the system and performs any data transformation (business logic).

[1] [Reenskaug,1979]

Model View Controller



Why MVC?

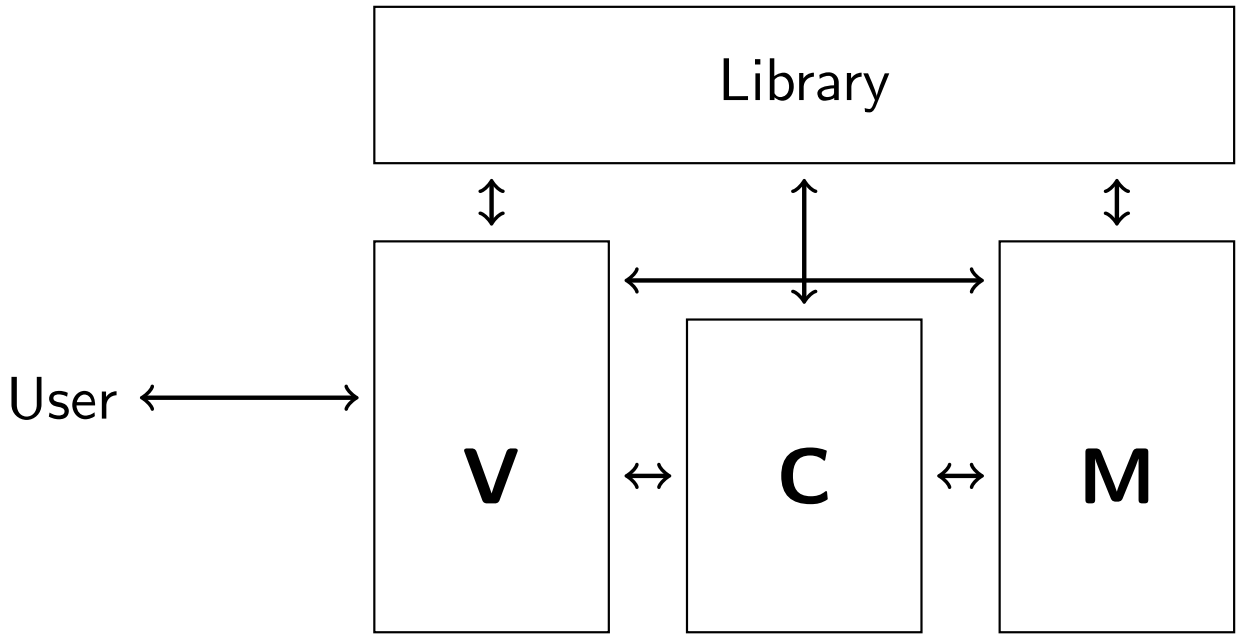


- Change of layer does not affect system
- Keep output, data and logic separated
- Supports DRY

Don't Repeat Yourself

Refactor code in own views, models, controllers or libraries.
It is not DRY if it is not **O**bject **O**riented **P**rogrammend.

Why MVC?

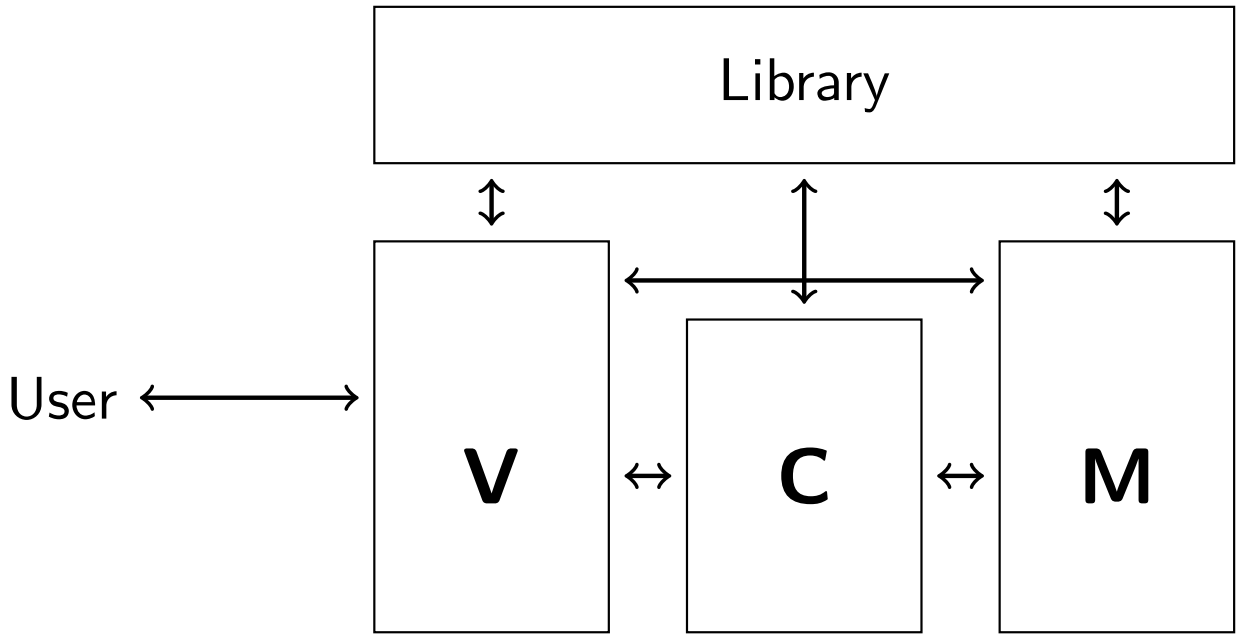


- Change of layer does not affect system
- Keep output, data and logic separated
- Supports DRY

Don't Repeat Yourself

Refactor code in own views, models, controllers or libraries.
It is not DRY if it is not **O**bject **O**riented **P**rogrammend.

Why MVC?



- Change of layer does not affect system
- Keep output, data and logic separated
- Supports DRY

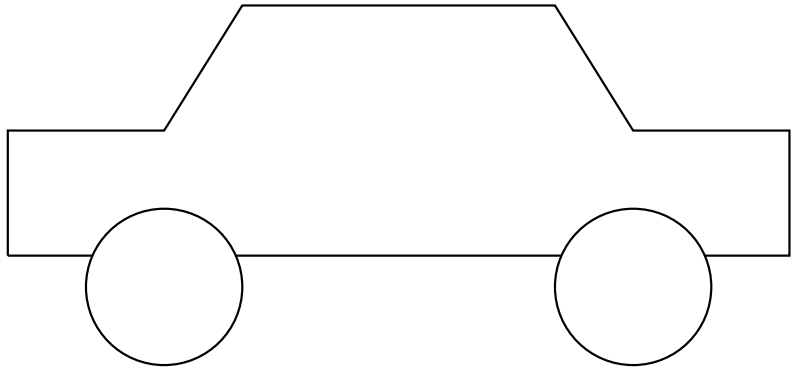
Don't **R**epeat **Y**ourself

Refactor code in own views, models, controllers or libraries.
It is not DRY if it is not **O**bject **O**riented **P**rogrammend.

Object Oriented Programming (OOP)

Properties

- number of seats
- motor power
- current speed
- wheel direction
- gear



Methods

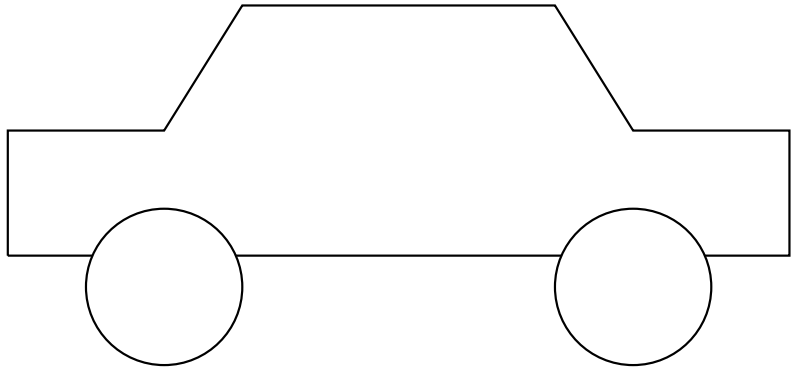
- Get, increase or reduce speed
- Get mileage, gasoline stand, ...
- Change direction, gear

$\Longrightarrow$ Program in a way that reflects reality

Object Oriented Programming (OOP)

Properties

- number of seats
- motor power
- current speed
- wheel direction
- gear



Methods

- Get, increase or reduce speed
- Get mileage, gasoline stand, ...
- Change direction, gear

$\Longrightarrow$ Program in a way that reflects reality

Testing

Theory vs. Practice

In theory there is no difference between theory and practice. In practice there is.

Test your code

- regularly in an
- automatized and
- randomized way.

If it's worth building, it's worth testing.
If it's not worth testing, why are you wasting your time working on it?

Testing

Theory vs. Practice

In theory there is no difference between theory and practice. In practice there is.

Test your code

- regularly in an
- automatized and
- randomized way.

If it's worth building, it's worth testing.
If it's not worth testing, why are you wasting your time working on it?

Testing

Theory vs. Practice

In theory there is no difference between theory and practice. In practice there is.

Test your code

- regularly in an
- automatized and
- randomized way.

If it's worth building, it's worth testing.
If it's not worth testing, why are you wasting your time working on it?

Why should I bother? [Federico Poloni]

"This is just temporary code for testing, it will never be released, so why should I bother?"

- Extremely useful to spot errors immediately
- Forces you to write well-structured code
- ~~Doesn't cost you much time~~
  Saves you a lot of time in the long run! ("future you" will confirm)

Why should I bother? [Federico Poloni]

"This is just temporary code for testing, it will never be released, so why should I bother?"

- Extremely useful to spot errors immediately
- Forces you to write well-structured code
- ~~Doesn't cost you much time~~
  Saves you a lot of time in the long run! ("future you" will confirm)

Example (Matlab)

```
x = solveAxb(A,b);
```

Testsuite

```
classdef TestAxb < matlab.unittest.TestCase
   methods (Test)
     function testDimensions(tc)
       tc.assertError(@()solveAxb(eye(2),rand(3,1)),...
         'solveAxb:wrongDimensions');
     end
   end
end
```

```
run(TestAxb);
```

Matlab Testcases

```
tc.assertEqual(functionCall,trueValue,'RelTol',1e-6);
tc.assertGreaterThan(..);
tc.assertLessThanOrEqual(..);
tc.assertTrue(..);
```

- Works only for Matlab version $\geq$ R2013a
- Detailed documentation: `http://www.mathworks.de/help/matlab/matlab-unit-test-framework.html`
- Before, use xUnit (Steve Eddins, not maintained any more) `http://de.mathworks.com/matlabcentral/fileexchange/22846-matlab-xunit-test-framework`

Test Driven Development

In general, the longer you wait before fixing a bug, the costlier (in time and money) it is to fix. Instead of infinite defects methodology use zero defects methodology.

It is easier to predict how long it will take to write new code than to fix an existing bug.

Test Driven Development

- Write the tests before you add functionality.
- Write the testsets in the specification.
- Add testsets to version control.

Test Driven Development

In general, the longer you wait before fixing a bug, the costlier (in time and money) it is to fix. Instead of infinite defects methodology use zero defects methodology.

It is easier to predict how long it will take to write new code than to fix an existing bug.

Test Driven Development

- Write the tests before you add functionality.
- Write the testsets in the specification.
- Add testsets to version control.

Test Driven Development

In general, the longer you wait before fixing a bug, the costlier (in time and money) it is to fix. Instead of infinite defects methodology use zero defects methodology.

It is easier to predict how long it will take to write new code than to fix an existing bug.

Test Driven Development

- Write the tests before you add functionality.
- Write the testsets in the specification.
- Add testsets to version control.

Coding Principles

Is code for computers or for people?

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler, Refactoring: Improving the Design of Existing Code

Make code readable not only for others, but also for the future you!

Comments in Code

You should write code in such a way that it can be understood without comments.

Debugging is harder than writing code

If you try to be as smart as possible in writing your code, how are you ever going to debug it?

Coding Principles

Is code for computers or for people?

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler, Refactoring: Improving the Design of Existing Code

Make code readable not only for others, but also for the future you!

Comments in Code

You should write code in such a way that it can be understood without comments.

Debugging is harder than writing code

If you try to be as smart as possible in writing your code, how are you ever going to debug it?

Coding Principles

Is code for computers or for people?

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler, Refactoring: Improving the Design of Existing Code

Make code readable not only for others, but also for the future you!

Comments in Code

You should write code in such a way that it can be understood without comments.

Debugging is harder than writing code

If you try to be as smart as possible in writing your code, how are you ever going to debug it?

Conding Principles

Keep **I**t **S**imple **S**tupid [Kelly Johnson]

- First, your code needs to work.
- Second, optimize (refactor) your code.

Every time you touch your code $\Longrightarrow$ make it better (dynamic refactoring).

Know your environment

Each different piece of technology or methodology you learn is another tool in your toolbox.

$\Longrightarrow$ Knowing which tool to apply to which problem can save you a ton of time.

Conding Principles

Keep **I**t **S**imple **S**tupid [Kelly Johnson]

- First, your code needs to work.
- Second, optimize (refactor) your code.

Every time you touch your code $\Longrightarrow$ make it better (dynamic refactoring).

Know your environment

Each different piece of technology or methodology you learn is another tool in your toolbox.

$\Longrightarrow$ Knowing which tool to apply to which problem can save you a ton of time.

Fur further reading

Coding

- Making Wrong Code Look Wrong
  `http://www.joelonsoftware.com/articles/Wrong.html`
- KISS `http://people.apache.org/~fhanik/kiss.html`

Testing

- Introduction to Test Driven Development
  `http://agiledata.org/essays/tdd.html`

Specification

- Painless Functional Specifications `http://www.joelonsoftware.com/articles/fog0000000036.html`

Fur further reading

Coding

- Making Wrong Code Look Wrong
  `http://www.joelonsoftware.com/articles/Wrong.html`
- KISS `http://people.apache.org/~fhanik/kiss.html`

Testing

- Introduction to Test Driven Development
  `http://agiledata.org/essays/tdd.html`

Specification

- Painless Functional Specifications `http://www.joelonsoftware.com/articles/fog0000000036.html`

Fur further reading

Coding

- Making Wrong Code Look Wrong
  `http://www.joelonsoftware.com/articles/Wrong.html`
- KISS `http://people.apache.org/~fhanik/kiss.html`

Testing

- Introduction to Test Driven Development
  `http://agiledata.org/essays/tdd.html`

Specification

- Painless Functional Specifications `http://www.joelonsoftware.com/articles/fog0000000036.html`