Title: Topic 3 Software process models (Topic03 Slide 1).

Topic 3: Lecture Notes (instructions for the lecturer)

Author of the topic: Klaus Bothe (Berlin)
English version: Katerina Zdravkova, Vangel Ajanovski (Skopje)
Author of the lecture notes: Klaus Bothe (Berlin), 15th Nov 2002

About the subject of this topic:
Each software development process moves through several phases. Each phase is connected with particular activities which lead to corresponding outputs (results or documents).

There are several software process models, among them the famous ‘waterfall model.’ This waterfall model forms the basis of several other models derived from it. In addition, this waterfall model determines the structure of software engineering textbooks as well as of this SE course.

To do:
- If new models are added, should we update the slide 10? YES
- Too much emphasis on phases and not on transitions between phases.
Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents an overview of models in point b).

Not each of the models mentioned there will be discussed in more detail in the current topic.

The following is no more true (I added all models here in topic 3., K.B.):

Some models are included in later topics: V model in topic 19 (Systematic testing).
Title: Process models: used terminology (Topic03 Slide 3).

Contents:
This is a list of terms which are similar or even equivalent to the term `software process model`.

- Software process models
- Phase models
- Life cycle models
- Software development models
- Project models

(Note: terms are similar, but with differences; e.g. life cycle model: development and maintenance, software development model: without maintenance)
Title: Why process models? (Topic 03 Slide 4).

Contents:
The IEEE definition demands a systematic approach to the development of software. "Systematic" means to have a clear understanding of what has to be done during this process, i.e., what steps are necessary and which results are required.
Title: Relevance of process models (Topic03 Slide 5).

Contents:
This slide summarizes 4 reasons why we need software process models.

Remark:
ISO 9000 will be introduced in Topic 24 (Quality of software development process and its standardization)
What is a process model?

- **Process model**
  - *In general*: Development plan, which specifies the general process of developing the product.
  - *More precisely*: Definition that states, which **activities** are to be performed, by which **person**, acting in which **role**; in which order the activities will be performed, and which **products** will be **developed** and how to **evaluate** them.

- **Role**
  - coworker, who accomplishes a certain activity
    - e.g. test engineer, project leader, design specialist, programmer, software ergonom

Title: What is a process model? (Topic03 Slide 6)

Contents:
Until now, we argued for software process models more or less intuitively. Now it is time to say what we understand by a process model.

At this slide the term is defined rather generally, independent of software development. Thus, this definition also applies to car manufacturing, street building, and other areas including SW development.

Answer to the question:
Of course, the definition of a process model is completely independent of software production. Nevertheless, we will apply this definition to our field.
Title: Performing an activity: basic scheme (activity subprocess of a process model) (Topic03 Slide 7).

Contents:

This basic scheme for the description of activities will be detailed or reused in succeeding topics of this SE course. Then, it will be applied to specific phases of the SW development process, e.g. to the definition phase or the implementation phase.

This general scheme connects the components of a process model: activities, persons, input and output documents (artifacts: models and documents), tools, methods, guidelines ...
Software artifacts

› Software artifact
   • Product of art created by humans
   • Can be a document, a model, or a program
   • Examples:
     - Document, e.g. requirements specification
     - Model, e.g. OOA model
     - Program, e.g. C++-Program.

› Software product
   • Set of software artifacts

*) US: artifact  UK: artefact

Title: Software artifacts  (Topic03 Slide 8).

US: artifact  UK: artefact

Contents:
This is a supplement to the preceding slide: The term of "software artifact" is being considered.

In summary, a software product is more than the program code: It is a set of artifacts which will include also documents and models, i.e. products of the early phases of SW development.
Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents an overview of models in point b).
Not each of the models mentioned there will be discussed in more detail in the current topic.
Some models are included in later topics: V model in topic 19 (Systematic testing)
Title: Most familiar process models: Overview (Topic03 Slide 10).

Contents:

This is a list of well-known software process models.

- The classical phase model is the earliest and most fundamental model
- The iterative phase model generalizes it.
- The V model adds the activity of validating the different products of the phase model,
- The spiral model is an abstract model — a meta model.
- Prototyping is a specific model which is complementary to the phase models.

No more valid:

†The V model will be discussed in detail in Topic 19 (systematic testing)†

It will be discussed in this topic

Remark:

In the literature, there are several additional process models:
- reuse-based development (I. Sommerville)
- transformational model (C. Ghezzi et al.)
- iterative enhancement (P. Jalote)
Models are always idealized, they give only a framework and must allow flexible working
   ➔ *must not hinder work*

Different classes of SW development problems
   ➔ *different suitable models*

  e.g. AI program: gradual evolution
   ➔ *Prototyping model is often used*
3. Process models

a) Why process models?
b) Overview of existing models
c) Classical and iterative phase model
   (waterfall model)
d) Alternative phases in phase models
e) Iterative-incremental software development
f) V model
g) Prototyping
h) Spiral model

Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents after the motivation in point a) an overview of models in point b).
Not each of the models mentioned there will be discussed in more detail in the current topic.
Some models are included in later topics: V model in topic 19 (Systematic testing)
Title: Software development process until 1970: considered as a black box (Topic03 Slide 13).

Contents:
What is the essential problem to be investigated in this topic? This picture highlights the fact that until 1970 (the beginning of SE as a discipline) the SW development process has been considered as a black box. Consequently,

- everybody could fill it with his/her own methods
- it was completely individual
- there were neither standard activities nor standard products
- there was no repeatable process – more a chaotic one.

Methodology:
Both questions are not questions to the audience.
What happens inside? should indicate our task to detail this black box.
Really: only until 1970? should be provocatively: Of course, also today there are still companies with a chaotic SW development.
Title: The classical waterfall model (1970) (Topic03 Slide 14).

Contents:
The whole SW development is divided into 4 basic phases (which can be detailed into sub-phases). Each phase is connected with particular activities resulting in particular products.
The last 5th phase concerns activities after delivery of the final product rather than SW development itself. That’s why, the dashed line. Details of each SW development phase are summarized at the next slide.
Title: Phases of SW development: subject and products (overview) (Topic03 Slide 15).

Contents:
Each SW development phase is being characterized with its subject and after the arrow with its products.

1. Analysis and Definition is short for Analysis of the problem + Definition of the requirements to the SW. Since the focus is on the external behavior of the SW system, we have to adopt the viewpoint of the user rather than the viewpoint of the SW engineer.

To be successful, this phase demands for an intensive cooperation between the customer (client) and the SW developer (contractor).

2. Design: ...

3. Implementation: ...

4. Testing: ...

Methodology:
Do not go into more detail at this moment. This is only an overview. Tell the students that details will follow in the succeeding topics of the lecture.
Title: Documents of SW development  (Topic03 Slide 17).

Contents:
This is a more informal illustration of the SW development (testing has been excluded). In addition, this figure also illustrates that cooperation is fundamental between the actors. E. g., the customer tells his informal ideas to the analyst who has to form a specification of them. The analyst and the designer cooperate via the specification: The designer reads it, and then she has to produce a design according to the specification. The underlying figure points out a difficulty: Features get lost or will be added because of communication problems.

Remark:
According to reality of SW development, only one woman has been included in this figure. The woman at this figure ‑ a designer ‑ determines the general outline of the software (= design), and the man (= programmer) has to work hard to realize these ideas :-)

Source: Schneider, SEUH 43, p. 123
Title: Information flow in SW projects (Topic03 Slide 18).
Title: The classical waterfall model (1970) (Topic03 Slide 16).

Contents:
Only after having explained the phases of SW development (previous slide), we can discuss the problems connected with this (idealized) model.

Methodology:
This discussion should be interactive and include students since all of them have experience in SW development (at least experience of practical university assignments.)

Answer to the question “Problems?”
This scheme is rather strict, and thus, will not completely correspond to real-life SW development. Nevertheless, this strictness is useful as a principal guidance to understand the SW development process.

Why is it too strict?:

1. No return: One phase will follow immediately after the former phase has been completed. In practice, there should be the possibility to return. E. g. during the implementation, the programmer recognizes inconveniences of the design which has to be repaired,
Title: The classical waterfall model (1970) (Topic03 Slide 16).

Contents:
Only after having explained the phases of SW development (previous slide), we can discuss the problems connected with this (idealized) model.

Methodology:
This discussion should be interactive and include students since all of them have experience in SW development (at least experience of practical university assignments.)

Answer to the question ‘Problems?’
This scheme is rather strict, and thus, will not completely correspond to real-life SW development. Nevertheless, this strictness is useful as a principal guidance to understand the SW development process.

Why is it too strict?:

1. No return: One phase will follow immediately after the former phase has been completed. In practice, there should be the possibility to return. E. g. during the implementation, the programmer recognizes inconveniencies of the design which has to be repaired,
Title: Iterative phase model (Topic03 Slide 19).

Contents:
This model overcomes some of the problems of the classical phase model in one respect: Return to the previous phase is possible.

An important application in practice is that during the implementation the programmer recognizes that the design (software architecture) is still not suitable, Thus, the design has to be corrected or extended.

The other important application: testing and debugging lead to modification in program code. And of course, during the usage a lot of errors will be found and return to previous phases is necessary.

Methodology:
First speak about the contents of the picture. After that show the question Problems? which should be posed as a question to the students.

Remark: (could also told to the students)
This picture also reveals that the representation of the phases as a waterfall (rather than a phase sequence) is a principal decision: It is easy (for the water and for the software developer) to go on step down. However, return is (for the water as well as for the SW developer) a more expensive direction ...
Title: Cost allocation in the software life cycle (Topic03 Slide 20).

Contents:

Maintenance costs are twice as much as SW development costs.

Coding / programming takes 7% of all software costs or 21% of SW development costs.

The first phase (analysis and definition) is divided into two activities:

- dealing with the analysis of the requirements
- dealing with the specification of these requirements

The same applies to testing which takes 15% of all activities or 45% of SW development activities. Thus, testing is the most expensive phase of SW development about two times as expensive as coding.

Remark (ZB):

This and the following slide show interesting facts about the different phases in software process.
According to this particular source, 20% of all errors are introduced during the analysis and definition phase. However, only 1% of all errors (i.e., 1/20 of the requirements errors) are already detected in this phase. And so on...

Most of the errors are being detected during different phases of testing.

Code review means: only inspecting (reading) the program code.

Field faults means: errors detected during usage in the application field of software after its delivery.

As a conclusion, errors are detected too late leading to large amounts of additional error removal costs.

Remark (ZB):
Besides interesting facts about the different phases in software process, it should be pointed out how important it is to make requirements as precise as possible. We shall soon need this remark in topic 5, before introducing reviews of requirements.
Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents an overview of models in point b). Not each of the models mentioned there will be discussed in more detail in the current topic. Some models are included in later topics: V model in topic 19 (Systematic testing)
Title: Phases according to Jacobson (Topic03 Slide 23).

Contents:
Jacobson is one of the three famous authors of UML (besides Booch and Raumbaugh)

Answer to the question:

- Design has been split. This corresponds to the practice in which, as a first step, a rough design is being developed which has to be detailed later, e. g. with respect to the final programming language.

- Testing is divided into three phases: unit testing, integration testing, system testing each of which is necessary in developing non-trivial practical systems.
Title: Phases according to Denert (Topic03 Slide 24).

Contents:
This is a phase model of a well-known German textbook. The author ľ Denert ľ is the head of an important German software company s d & m (Munich).

Answer to the question:
There are 4 phases each of which covers sub-phases:
The system specification phase corresponds to the analysis and design phase. In this phase, Denert investigates three aspects: user interface, functional model, data model ľ and so on ... 

Methodology:
It is not necessary to explain each term in this picture. The main message is: Phases can be detailed by subphases.
Title: Phases in CASE-Tool MAESTRO II (Topic03 Slide 25).

Contents:
This phase model is connected with a particular CASE tool MAESTRO II – the most popular German CASE tool during the last two decades. Customers of Maestro II are, among others, German Bank, other banks, Allianz and other assurance companies ...

The top line covers 4 phases.
The bottom line lists their results.

Answer to the question:
- The first phase (analysis and definition) is divided into two phases.
- Implementation and test (together with integration) are included in one phase

MAESTRO II is offered for the development of huge SW systems in which the activities connected with the external behavior take a large part that’s why the division into two phases
Title: Unified Software Development Process (USDP) (Topic03 Slide 26).

Contents:

USDP is the software development process connected with the object-oriented SW development in UML.

The only specific aspect is the division of the first phase 'Analysis and Definition' (in the waterfall model) into two phases 'Requirements' and 'Analysis'. The first phase results in use case diagrams, the second in class diagrams.

Remark:

Use case diagrams and class diagrams will only be introduced in later topics.
Title: Phases according to Balzert (Topic03 Slide 27).

Contents:
This is the inner front cover of Balzert’s textbook giving an overview of the book contents. In the middle, the phases of SW development defined by Balzert are summarized. These phases determine the structure of the textbook in a high degree.

Methodology:
Details of these phases should not be explained now -> see: next slide
Title: Phases according to Balzert (detail) (Topic03 Slide 28).

Contents:
The phases in detail

Answer to the question:
- First phase ‘analysis and design’ two phases (however different from MAESTRO II)
- Testing not included: however, considered as a part of the implementation phase
- An additional phase ‘acceptance and introduction’
Title: Phase model in the Space shuttle Program (NASA) (Topic03 Slide 29).

Contents:
Between 1981 and 2001, NASA has developed 22 releases of the space shuttle software (i.e. about one per year). Such a release is also called an operational increment (OI) because each release increments the former release, i.e. new functionality has been added. Each new release covers about 5.7% of new or modified system code.

The software is expected to evolve at least to the year 2020.

To each release corresponds a lifecycle covering the four phases indicated in this picture: development, ...

The graphics indicates: about 3 releases were in various phases of development and up to 4 releases were active at the same time.
Consequently, debugging activities in one release have an effect to the corresponding other active releases.
Title: Special phase model (Topic03 Slide 30).

Contents:
This special phase model should be discussed with the students.

Methodology:
The answer to the two questions have not been included in the slides. Consequently, during the next semesters reuse of slides and questions is supported.

Answer to the question **What is remarkable?**
- This phase model includes detailed information about the length of the phases.
- This phase model contains coverage information:
  - Which phase overlaps with other phases?
    - Overlapping is rather strong.
- The whole process takes exactly one year.
- Test is only a very short phase: 3 days.
- The terms partly differ from the terms introduced so far.

Answer to the question **How to be interpreted?**
This phase model does not concern the software production!
This phase model is dedicated to the production of ships.
Title: Phase model of Samsonite (Topic03 Slide 31).

Contents:
Samsonite produces bags, suitcases and the like.
Because the products should have a high quality, there is a strong phase model of product development covering 4 phases: design, manufacturing, test, certification.
Very surprising for such a simple product like a suitcase!
Title: Test of suitcases  (Topic03 Slide 32).

Contents:
The test phase is described for Samsonite products. Four test situations are described with detailed aspects to be tested covering particular quantitative statements.

Remark:
This and the former picture are delivered together with Samsonite products to the customers.
Conclusion: process models are not an invention of software engineering
3. Process models

a) Why process models?
b) Overview of existing models
c) Classical and iterative phase model
   (waterfall model)
d) Alternative phases in phase models
e) Iterative-incremental software development
f) V model
g) Prototyping
h) Spiral model

Remark:
This topic presents an overview of models in point b).
Not each of the models mentioned there will be discussed in more detail in the current topic.
Some models are included in later topics: V model in topic 19 (Systematic testing)
The fundamental problem of this approach is that it pushes risk forward in time, where it’s costly to undo mistakes from earlier phases. An initial design will likely be flawed with respect to its key requirements, and furthermore, the late discovery of design defects tends to result in costly overruns and/or project cancellation. The waterfall approach tends to mask the real risks to a project until it is too late to do anything meaningful about them.

To continue the example. At the end of two years, since we have not actually executed anything yet, we have just as much risk remaining in the project as we had when we started.

Animation note: the green line is drawn automatically.
Iterative and incremental development

Iteration – development in steps, gradual product improvement
Increment – independent part of product that is developed separately (e.g. by a team)

Cockburn, 2008
In the iterative approach, the waterfall is applied to a single increment of the system at a time. Each iteration produces an executable.

Iterative processes were developed in response to these waterfall characteristics. With an iterative process, the waterfall steps are applied iteratively. Instead of developing the whole system in lockstep, an increment (that is, a subset of system functionality) is selected and developed, then another increment, etc. The selection of the increment to be developed first is based on risk, the highest priority risks first. To address the select risk(s), a subset of use-cases are chosen. The minimal set of use-cases that will allow objective verification (i.e., through a set of executable tests) that this risk has been addressed, is developed. Then the next increment is selected to address the next highest risk, and so on. The waterfall is thus applied at the increment level and the system evolves incrementally.
The most critical characteristic of iterative development is that project risk is driven down in early iterations before significant implementation investments are made.

Going back to our 2-year project that actually takes 3 years: It may still take 3 years, but at the end of 2 years, we have removed most of the risk and have something that executes, although in a limited way. We are in a much stronger position.

Animation notes: The risk lines are automatically drawn after the axes are drawn.
The iterative approach is supported by the Rational Unified Process (RUP), a full lifecycle software engineering process which embodies all of Rational’s recommended best practices together with the Unified Modeling Language (UML). RUP provides detailed instructions on how to develop a system iteratively. RUP describes the activities, workers, and artifacts of each iteration. It places the iterations in the context of a software development cycle consisting of four phases with well-defined objectives, boundaries, and milestones. RUP will be described in detail in a later module in this course.

Note that the process workflows shown in the graphic depicting the iterative model match the waterfall very closely. Taking a vertical slice corresponding to one iteration, one can see that all of the workflows apply to each iteration, although the relative emphasis shifts from phase to phase. We will come back to the iterative model graphic frequently throughout the course in explaining how the iterative approach is applied in practice. It is our roadmap into the process.
Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents after the motivation in point a) an overview of models in point b).
Not each of the models mentioned there will be discussed in more detail in the current topic.
Some models are included in later topics: V model in topic 19 (Systematic testing)
V-Model

- Extension of the Waterfall model: Quality assurance (test) of all partial products included
- Origin: Boehm 1981
- Mandatory, standardized process model for „Bundeswehr“ (ca. 1992)
  later: federal administration, Industry
  (coherent standard for all public sectors)
- V-Model’97: conformance to ISO 9000, OO techniques, standard software...
Distribution of test phases

Title: Phases of iterative prototyping (Topic03 Slide 35).

Contents:
This is a phase model of the book Design for Multimedia Learning published by Prentice Hall Europe. The figure is from the first edition published in 1997.

Methodology:
It is good to start with an introduction of the phases. Then, the flow of the iterative prototyping is discussed.

Answer to the question:
1. A natural form of project development based on interleaved form.
2. Defines an early prototype which usually ends with a demo version of the product (system).
3. The evaluation feeds back to guide design decisions.
4. Dependent on the active involvement of the users. In many occasions, end users are part of the design team.
Title: Dynamic Systems Development Method (DSDM) (Topic03 Slide 36).

Contents:
Flexible methods of project development based upon iterative prototyping are known as RAD (Rapid Application Development).

DSDM is a RAD method accepted by at least 200 software development companies and universities. DSDM is based on nine principles. The most important are: requires iterative prototyping, requires a high involvement of the users (they are part of the design team), it is product-oriented.

Boyle: The method has become popular in commercial software development. It has been developed in reaction to the perceived limitations and rigidities of established variants on the waterfall approach.

Remark:
DSDM life cycle passes through four main stages:
Å Feasibility and business study
Å Functional prototype iterations
Å Design prototype iterations
Å Placing the system in user environment

Evaluation is a parallel activity throughout the whole life cycle.
3. Process models

a) Why process models?
b) Overview of existing models
c) Classical and iterative phase model (waterfall model)
d) Alternative phases in phase models
e) Iterative-incremental software development
f) V model
g) Prototyping
h) Spiral model

Title: 3. Process models (Topic03 Slide 2).

Remark:
This topic presents after the motivation in point a) an overview of models in point b).
Not each of the models mentioned there will be discussed in more detail in the current topic.
Some models are included in later topics: V model in topic 19 (Systematic testing)
Title: Prototyping (Topic03 Slide 34).

Contents:
Prototyping is different from the phase models introduced so far.
Due to the fact that requirements at the beginning of a project are often rather imprecise, an operational prototype is being implemented.

By this procedure, prototyping aids the construction of the requirements document by illustrating the desired requirements by means of the prototype. Hence, prototyping and phase models can be considered complementary.
### 3. Process models

- **a)** Why process models?
- **b)** Overview of existing models
- **c)** Classical and iterative phase model  
  (waterfall model)
- **d)** Alternative phases in phase models
- **e)** Iterative-incremental software development
- **f)** V model
- **g)** Prototyping
- **h)** Spiral model

**Title:** 3. Process models (Topic03 Slide 2).

**Remark:**

This topic presents an overview of models in point b).

Not each of the models mentioned there will be discussed in more detail in the current topic.

Some models are included in later topics: V model in topic 19 (Systematic testing)
Title: Introduction to spiral model (Topic03 Slide 38).

Contents:
This picture is intended as a helpful introduction to the spiral model.
5 phases are being distinguished which sound slightly different from the notions from the waterfall model.

Answer:
The phases are very abstract.
The activities of a phase do not refer directly to the software development: Risk analysis, for example, may refer also to the production of ships or suitcases. Risk analysis, in the environment of software development, may refer to each phase we have known so far.
Nevertheless, the spiral model is a special model for SW development it however, a rather abstract one.
The backward arrow indicates some kind of cycle in this process it and this cycle will be expressed in the form of a spiral.
Title: Spiral model due to Boehm (1988) (Topic03 Slide 39).

Contents:
The cyclic process of software development as a spiral.
Some remarks can be found at the next slide.
Title: The essence of the spiral model (Topic03 Slide 40).
Title: The phase model as a special case of the spiral model (Topic03 Slide 41).

Contents:
Since the phase model is a rather abstract one, it can be specialised: Either to an ordinary phase model (waterfall phases) or to an evolutionary model (next slide).

Each turn corresponds to a waterfall phase with (five) similar general activities:
Problem analysis, risk analysis, realization, planning and review as activities of the phases analysis and definition, design, coding and testing, operation and maintenance.
Title: Evolutionary software development as a special case of the spiral model (Topic03 Slide 42).

Contents:
Each turn corresponds to a step in the evolutionary software development. For example, one turn can produce a new prototype covering our (five) general activities.
IEEE Std 1074-1997

IEEE Std 1074-1997
(Revision of IEEE Std 1074-1995; Replaces IEEE Std 1074.1-1995)

IEEE Standard for Developing
Software Life Cycle Processes

Sponsor
Software Engineering Standards Committee
of the
IEEE Computer Society
Approved 9 December 1997
IEEE Standards Board

Abstract:
A process for creating a software life cycle process is provided. Although this standard
is directed primarily at the process architect, it is useful to any organization that is responsible for
managing and performing software projects.

Keywords:
software life cycle, software life cycle model, software life cycle process
### Messages

<table>
<thead>
<tr>
<th>Phase models are decisive for the project success and prevent from chaotic software development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase models are one of the fundamentals of SE</td>
</tr>
<tr>
<td>The waterfall model is the</td>
</tr>
<tr>
<td>• oldest one</td>
</tr>
<tr>
<td>• is not ideal</td>
</tr>
<tr>
<td>• has drawbacks in practice</td>
</tr>
<tr>
<td>• is fundamental for all other models</td>
</tr>
<tr>
<td>• is used as the reference model in many approaches and textbooks</td>
</tr>
<tr>
<td>Different phase models serve different needs of a special project, e.g.</td>
</tr>
<tr>
<td>• Prototyping in case requirements are not clear in the beginning</td>
</tr>
<tr>
<td>• For large projects: very detailed phases (Denert, MAESTRO)</td>
</tr>
<tr>
<td>• For large projects: iterative-incremental (RUP)</td>
</tr>
<tr>
<td>Phase models are not unique for software development. In all engineering disciplines, phase models are crucial, e.g. car manufacturing, ship production, suitcase construction ...</td>
</tr>
</tbody>
</table>