Course "Softwareprozesse"

Introduction

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

- Assumed knowledge
- What we talk about, what not
- Software development process
- Learning goals

- Classical view vs. modern view
  - Epistemological stances
  - Means of process description
  - Cultural backgrounds
- Definition "culture"
- Royce's 1970 "waterfall" article

Assumed prior knowledge

- This course assumes you are roughly familiar with the following notions:
  - requirements elicitation, requirements specification
  - architectural design, lower-level design
  - implementation, coding
  - software quality
  - analytic quality assurance: testing, review, static analysis
  - constructive quality assurance: process design and improvement
    - software development process model: waterfall model, agile, Scrum
      - contrast of phase models vs. iterative models
    - activities, roles, artifact types
  - project management: planning, control, coordination, communication, risk management

Please close any gaps using conventional (not: agile-only) Software Engineering teaching materials, e.g. FU Berlin V-SWT

What types of SW development we talk about the most

Freie Universität Berlin

- Small to modest scale
  - 3 to 100 people
- Mostly information systems
  - as opposed to e.g. embedded software or games
- Modest criticality
  - may be business-critical, but:
  - no lives depend on it
  - occasional individual transaction failure is bearable

(the modern view is much more helpful here than the classical view)

(we will define those below)

What types of SW dev. we talk about much less

- Large-scale development
  - 100s or 1000s of developers
  - **but** our processes apply as elements in large-scale development
- Systems Engineering
  - concurrent development of HW and SW, e.g. an autonomous car
  - **but** our processes often apply in subprojects in systems engineering
- High-risk application domains
  - e.g. aerospace, many medical devices, many financial services, …
  - **but** our processes often apply in projects attached to high-risk developments

(modern view and classical view are both helpful here)

The software process is the sequence of events
that happen during software development.

Fundamental discriminations:

- **Orientation in Time:**
  - oriented towards the past: **descriptive** software process
  - oriented towards the future: **prescriptive** software process
- **Level of detail**:
  - high level of detail ("fine grain"), low abstraction
  - lower level of detail ("coarse grain"), higher abstraction

- Combining these 2 factors produces four types of discussion:

4 types of software process discussion

- descriptive, high detail:
  - Detailed description of a past process
  - Useful for learning for *process improvement at the team level* or project level: Postmortems, Retrospectives; research
- descriptive, high abstraction:
  - Summarizing description of many past processes
  - Useful for learning for *process improvement at the organization level*: Experience factory, CMMI Level 5; research
- prescriptive, high detail:
  - project plan
- prescriptive, low detail:
  - software development process model ("**process model**")
  - *Our main perspective in this course*

Learning goals for the course

1. Understand a portfolio of software process elements and ideas (level 2 in the Bloom taxonomy)
   - and their rationale, pros, and cons
2. Recognize faults in process implementations and recognize bad process choices (levels 4, 6)
3. Argue how and why these are broken
   - referring to process goals and their relative priorities (4, 6)
4. Suggest improvement steps
   - based on the process elements portfolio (5, 6)
5. Exercises: Practice some process elements
   - (3, 4, 5)

Bloom taxonomy:
6. evaluation
5. synthesis
4. analysis
3. application
2. comprehension
1. knowledge

- **Classical view** (e.g. "Waterfall SW development"):
  - Software development should be a highly structured process.
  - For a given type of SW, there is a Right Way to develop it, which organizations must approximate as best they can.
  - Software development is good if it realizes the given set of requirements with high quality in the planned time and budget.

- **Modern view** (e.g. "Agile SW development"):
  - Software development is a difficult process – too difficult to completely pre-structure it.
  - Each team must find its own way of working in a given project.
  - Software development is good if it produces software of high value quickly and sustainably.

- One can learn a lot from contrasting these views
  - So that is what we will do throughout the course

Meta-view and assumptions underlying this course

- The two views represent two different "cultures"
- Each makes certain assumptions
- These assumptions are not always correct
  - They are sometimes good,
    sometimes dubious,
    and sometimes nonsense
- To avoid failure, we must check our assumptions
- To check assumptions, we must become aware of them
  - so we will keep asking ourselves for assumptions made

- Statements on software engineering should be interpreted in light of the culture (and assumptions) they come from

Reprise

- **Classical view** (e.g. "Waterfall SW development"):
  - Software development should be a highly structured process.
  - For a given type of SW, there is a Right Way to develop it, which organizations must approximate as best they can.
  - Software development is good if it realizes the given set of requirements with high quality in the planned time and budget.

- **Modern view** (e.g. "Agile SW development"):
  - Software development is a difficult process – too difficult to completely pre-structure it.
  - Each team must find its own way of working in a given project.
  - Software development is good if it produces software of high value quickly and sustainably.

What are the assumptions made by each view?

Underlying epistemological stance (dt.: erkenntnistheoretischer Standpunkt)

Simplistic formulations

The two views differ on epistemological stance

- Epistemology: "What can we know? How?"
  - https://plato.stanford.edu/entries/epistemology/
- Epistemological stance:
  - The assumptions/beliefs about what we can know and how to get there
    - There are many of those in philosophy, by various names; most names are overloaded many times
  - Most people do not know their stance explicitly
    - but have one nevertheless
- Process models also rarely formulate such a stance
  - but are heavily influenced by one nevertheless,
  - so let's compare our two views' stances:

Classical view epistemological stances

Simplistic formulations

- Rationalism:
  - The main source of (and touchstone of) insight is reason
    - dt.: Hauptquelle und Prüfstein von Wissen ist der Verstand
    - appropriate view in pure mathematics and many areas of theoretical computer science

- Positivism:
  - There is a single truth and we can know it.
    - appropriate for physics
    - appropriate for Informatics where pure algorithms are concerned

Modern view epistemological stances

Simplistic formulations

- Empiricism:
  - The main source of (and touchstone of) insight is observation
    - dt.: Hauptquelle und Prüfstein von Wissen sind Beobachtungen
    - https://plato.stanford.edu/entries/rationalism-empiricism/
    - (does not claim that reasoning is superfluous)
    - appropriate for all natural science
    - appropriate for Informatics wherever emergent phenomena are relevant

- Interpretivism:
  - Understanding the world requires interpretation
  - Some interpretations are better than others, but there is no single right one
    - appropriate for most of social science
    - appropriate for Informatics wherever human beings are involved

Classical view: Means of process description

- The process consists of different (types of) **activities**.
- An activity takes one or more **artifacts** as **inputs** and produces one or more artifacts as **output**.
  - These dependencies impose some ordering on activities.
  - An activity may use one or more software **tools**.
- It is performed by one or more people in different **roles**.
  - The activity itself is often described only vaguely, because it is complex, knowledge-intensive, and variable

- ***Example***: Interactive exploratory testing
  - Performed by: Tester (or: Exploratory Tester)
  - Input: Executable software system, known failure scenarios
  - Activity: Tester attempts to use system such that it fails. Acts based on (1) understanding of requirements, (2) previous uses of system, (3) personal testing skills.
  - Output: List of new failure scenarios.

Classical view: Notes on roles

1. In a project, the same person can act in multiple roles.
2. Many roles are filled by several people,
   others by only one person.

3. There can be different connections of a role to an activity,
   e.g.
   performs, participates, informs, reviews, etc.
   - https://en.wikipedia.org/wiki/Responsibility_assignment_matrix

4. A role assumes a set of *skills* and
   implies a set of *responsibilities*.

Modern view: Means of process description

- Fewer roles
  - e.g. four rather than a dozen or more
- Few activities
- Very few artifact types ("lightweight process")

- Practices
  - strong (but not absolute) behavior suggestions, non-local
  - Example: *"Energized work: Work only as many hours as you can be productive and only as many hours as you can sustain."*
  - sometimes more explicit than activities

- Principles and values
  - global ideas and prescriptions, very abstract

Summary of classical view vs. modern view

Epistem. stance:

rationalism, positivism — empiricism, interpretivism

Means of description:

activities, artifacts, roles — fewer ditto, practices, principles

Ideals:

Central ideas:

View on people:

Cultural background of classical view: An industrial perspective

Simplistic formulations

Claims, to be checked

- Term "Software Engineering" coined 1968
  - when computers were more expensive than programmers.
  - Engineering* ideals:
    - precise planning,
    - getting it right at once (like in civil engineering),
    - quality as fulfilling the specifications (as in industrial manufacturing)
  - The idea of a **specification** is central
    - Product: requirements specification, then design specification
    - Process: project plan

Claim, to be checked

- View on people:
  People are resources by which work gets done
  - People with similar qualification and skill are exchangeable

*not *necessarily* right for software

Cultural background of modern view: A humanist perspective

Freie Universität Berlin

Simplistic formulations

- "Agile Manifesto" formulated 2001
  - by then, programmers are far more expensive than computers
  - Ideals: *"Through [our] work we have come to value:*
    - *Individuals and interactions* *over processes and tools*
    - *Customer collaboration* *over contract negotiation*
    - *Responding to change* *over following a plan"*

humanist preference

industrial/engineering preference

  - Central ideas: **collaboration** and incremental improvement

Claim, to be checked

- View on people:
  People (as developers, customers, users) are the focus of attention
  - They are both drivers and purpose of the process

Summary of classical view vs. modern view

Freie Universität Berlin

Epistem. stance:	
rationalism, positivism	empiricism, interpretivism
Means of description:	
activities, artifacts, roles	fewer ditto, practices, principles
Ideals:	
engineering ideals:	humanist preferences:
planning, getting it right at once quality is meeting the specs	reacting, iterating, strive to produce high value
Central ideas:	
specification	collaboration & iteration
View on people:	
exchangeable resources	drivers and purpose of process

Notation

Our slides will often use the following annotations:

means "a classical-view element or statement"

means "a modern-view element or statement"

means "a mixed element or statement"

means "What is classical or modern here?"

and we point out the aspect(s) or anti-aspect(s)
(e.g. *"This is anti-positivist, because…"*)

Two cultures?

- If the above claims are correct,
  the two views imply two different cultures
  of software development

- We will analyze each process we discuss
  for traces of these two cultures:
  - How much classical-view culture is visible in the process?
  - How much modern-view culture is visible in the process?

Definition "Culture" [ShwBel15]

- *"Explicit and implicit patterns of behavior"*
  - that constitute achievements of a human group and
- *"traditional ideas and […] their attached values"*.
  - *"Conventional understandings manifest in act and artifact."*
- ***"'culture' refers to community-specific ideas about what is true, good, beautiful, and efficient."***

Discussion:

- Each culture may have different strengths
  - so learning about several cultures can be helpful.
  - Cultures can be mixed.
- Whether one culture is preferable over another depends on
  - context (e.g. team, code base, rqmts., surrounding organization)
  - the values held by the observer.

Exercise: The waterfall process ("phase model") is an incorrect legend

The hallmark of the classical view

- Royce is often claimed to have proposed the Waterfall process
  - W.W. Royce: Managing the development of large SW systems, IEEE WESCON 1970
- but in fact he warns of it:
  - *"the implementation described above is risky and invites failure"*
- and goes on to describe a sort of iterative development:
  - 1. Preliminary program design during requirements analysis
  - 3. Plan to throw version 1 away, to reduce risk.
  - 4. Test thoroughly
  - 5. Involve the customer in requirements interpretation
- Domain is "spacecraft mission planning, commanding and post-flight analysis."; context is the 1960s weak computers

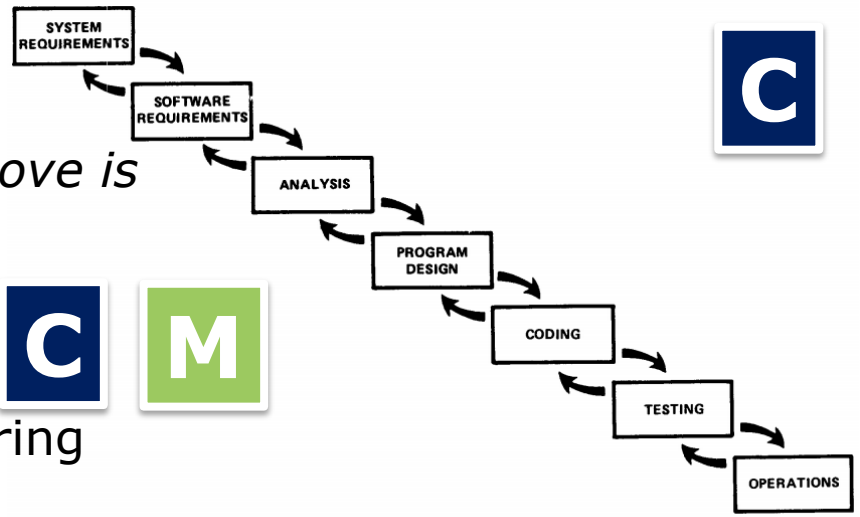

Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.

Summary of Royce's actual proposal

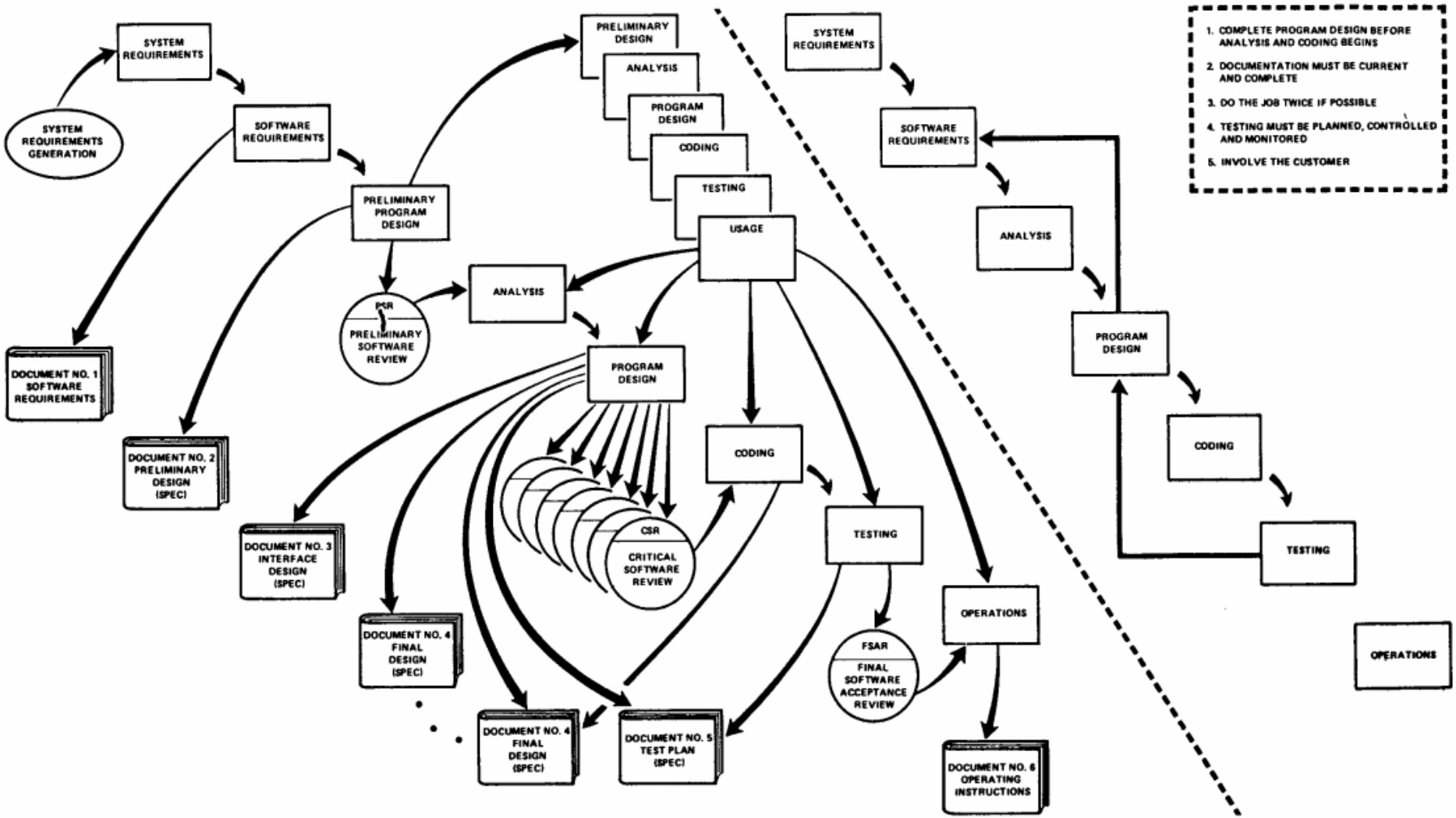

Figure 10. Summary

Some traces of the views in Royce's article

1. Step 1: Preliminary program design:
   - *"Design, define and allocate the data processing modes even at the risk of being wrong."*
2. Step 3: "Do it twice"
   - *"[The people building the prototype] must have an intuitive feel for analysis, coding, and program design."*
3. Step 5: "Involve the customer"
   - *"For some reason what a software design is going to do is subject to wide interpretation even after previous agreement."*
   - *"To give the contractor free rein between requirement definition and operation is inviting trouble."*

(There are lots of traces of classical view as well)

Conclusion: (1) Pay attention to what people *really* think.
(2) Always expect to see a mix of both views.

We make this a habit!

We will frequently put this sign 

and then analyze original statements of creators or fans of a process for traces of classical and modern view.

Note: "classical"? "modern"? On the connotations of the adjectives

Freie Universität Berlin

- *classicus* (lat.) means exemplary (dt: "*mustergültig*")
  - https://en.wiktionary.org/wiki/classicus
  - The adjective is applied to things that are (or once were) considered exemplary or ideal

- *modern*:
  - A term with very diverse meanings. For our purposes:
  - The result of questioning previous "classical" assumptions and forming a more refined view
    - https://de.wikipedia.org/wiki/Moderne

Our pair of classical/modern resembles that of physics:

- Classical Physics (e.g. Newtonian mechanics) is approximately correct from the point of view of
- Modern Physics, with relativistic and quantum phenomena.
  - Classical Physics is not considered wrong, but is incomplete

Learning goals, revisited

As a result from this course

1. you will know a nice inventory of process parts;
2. you can "smell" and diagnose cultural aspects (classical as well as modern) in a process setting;
3. you can judge which behaviors and attitudes in a process are helpful or problematic;
4. you can explain this and hence help organizations to become saner and better.

Thank you!