ETSN05 – Lecture 5

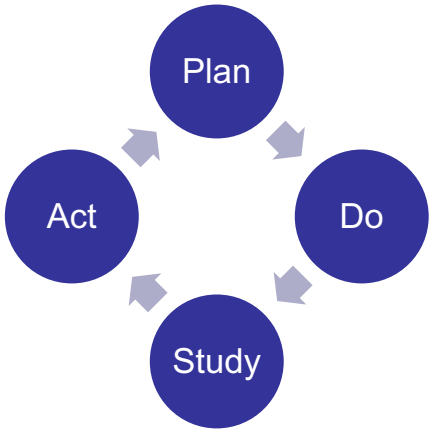
1

Today

- About final report and individual assignment
- Short introduction to “problems”
- Discussion about “typical problems in the course”

2

Plan-Do-Study-Act

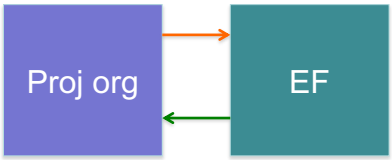


E.g. Bergman, Klefsjö, Quality, Studentliteratur, 1994.

3

Quality Improvement Paradigm

1. Characterize
2. Set goals
3. Choose model
4. Execute
5. Analyze
6. Package



V. Basili, G. Caldiera, D. Rombach, Experience Factory, in J.J. Marciniak (ed) Encyclopedia of Software Engineering, pp. 469-576, Wiley, 1994.
(Also summarized in C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, "Experimentation in Software Engineering", Springer, 2012.)

4

Learning organization

- "A learning organisation is an organisation skilled at creating, acquiring, and transferring knowledge, and at modifying its behaviour to reflect new knowledge and insights"

D. A. Garvin, "Building a Learning Organization", in Harward Business Review on Knowledge Management, pp. 47–80, Harward Business School Press, Boston, USA, 1998.

- Requires: systematic problem solving, experimentation, learning from past experiences, learning from others, and transferring knowledge

5

Postmortem analysis

focus
knowledge management

Postmortem: Never Leave a Project without It

Andreas Birk, *sdrm*
Torgeir Dingsøyr, *Sintef Telecom and Informatics*
Tor Stålhane, *Norwegian University of Science and Technology*

In every software project, the team members gain new knowledge and experience that can benefit future projects and each member's own professional development. Unfortunately, much of this knowledge remains unnoticed and is never shared between individuals or teams.

Our experience with project *postmortem analysis* proves that it is an excellent method for knowledge management,[1] which captures experience and improvement suggestions from completed projects and works even in small- and medium- size companies that cannot afford extensive KM investments. However, PMA has been mainly advocated for situations such as completion of large projects, learning from success, or recovering from failure.[2–4]

When used appropriately, PMA ensures that team members recognize and remember what they learned during a project. Individuals share their experiences with the team and communicate them to other project groups. Additionally, PMA identifies improvement opportunities and provides a

can apply a number of fairly simple methods, such as the KJ method (after Japanese ethnologist Jiro Kawakita)[7] that collects and structures the data from a group of people.

Preparation

When we conduct PMA in software companies, two software process improvement group members work as facilitators together with two to all project team members. Facilitators organize the analysis, steer the discussion, and document the results. They can be employees in the company where the PMA is

- IEEE Software, May/June 2002

6

Postmortem analysis cont.

- "Ensures that the team members recognize and remember what they learned during the project"
- "Identifies improvement opportunities and provides a means to initiate sustained change"

7

Three sections of the PFR

- Historical overview of the project
  - Figures, tables, diagrams, etc
  - Comparison between actual values and estimations
- Evaluation of what went well and what went not so well
  - Analyze reasons for problems/issues/etc
- Software process improvement proposals

Read PH:8.10

8

Include at least

- Effort per phase
- Start and end dates for each phase
- Effort per document
- Start and end dates for each document
- Effort for different activities in each phase
- Effort per group & week
- Analysis of problem reports in phases

9

Final report evaluation

- extent
  - how far does the report get in:
    WHAT has happened (data reporting)
    WHY it happened so (cause analysis)
    WHAT can be done to improve (SPI)
- quality
  - how well are the chosen topics reported

10

PFR development process

- Ongoing now… you need to collect data and you need to analyze/discuss the data (and I know you are doing that)
- PL main authors but they need input from everyone
- Informal reviews are recommended → they improve quality → shared responsibility

11

Individual report

See course homepage – pdf document with instructions.

Objectives:

- to stimulate reflection on large-scale software development continuously through the course,
- to encourage a viewpoint on industrial practice in large-scale software development,
- to build on and integrate with what you have learned from previous courses

Identify relevant areas and motivate your conclusions.

12

Analyse:

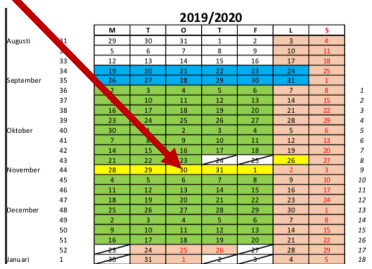
- What was challenging and what was easy in your own work?
- What was challenging and what was easy in your fellow project members work?
- What are the similarities and differences between the controlled course situation and industrial practice?
- What is realistic about the course setting, and what is not so realistic?
- What role does the scale (in terms of size and complexity) play in the challenges you have had during the project?
- What is easy in a small-scale project while significantly more challenging in a large-scale project? Which problems are *not* more or less difficult to address when carried ou scale setting compared to a smaller scale settin
- Possibilities to introduce more agile practices (a challenges)?

Reference to scientific literature (e.g. conference articles, journal publications) Use e.g. Scopus

13

Individual report deadline etc

- Deadline: Oct 30
  - Then you have all necessary information
- Submission instructions will be posted on course web page
- Don't forget to write your name on the report.



14

Examples of real difficulties in real industrial projects

1. What is actually the best set of requirements?
2. How much uncertainty in effort estimation can we cope with?
3. At what level of detail should we document requirements?
4. How to minimize waiting time for other parts to be ready before we can start our part?
5. How to make more parts in parallel without generating confusion and unnecessary rework?
6. How to know when the product is reliable enough to be released?
7. How to incorporate changes without generating spaghetti and excessive cost of rework?

15

Some examples of “problems” that probably are not very big in your project

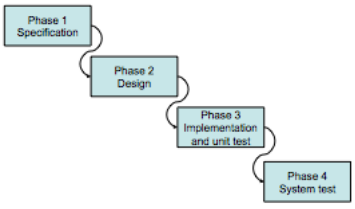
16

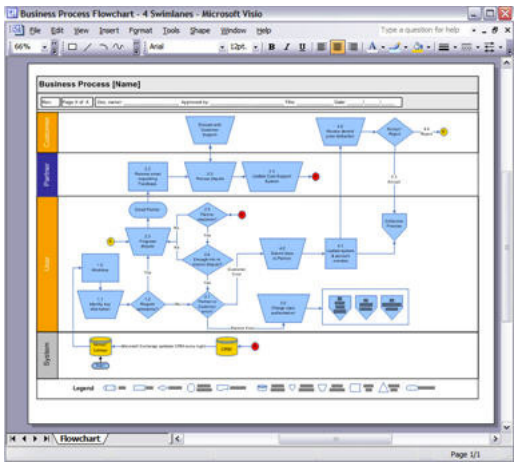
Multi-project environment

Risks:
- Sub-optimization
- Uncoordinated
- …

17

Elementary or advanced process?



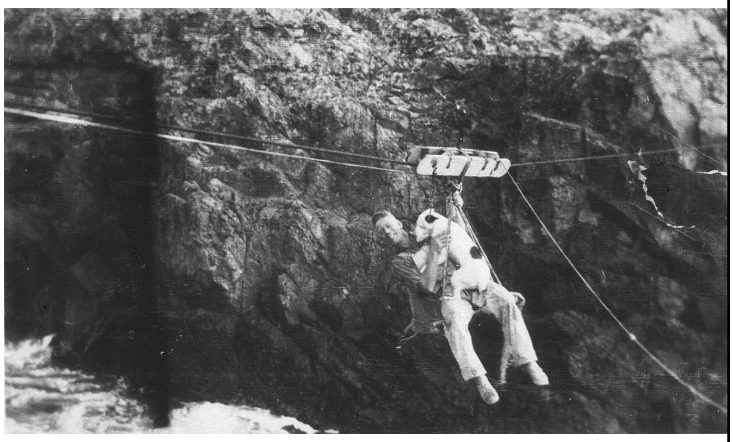


CC BY-NC 2.0
i a walsh

18

Chasm between marketing and development

Difference between development and operations



19

Living with changing requirements

20

Requirements are invented rather than discovered

Innovation capability is critical

21

…and an example of a problem that you maybe have seen:

- It is unclear what the requirements are, and what the customer really expects…

22

Eliciting and understanding is difficult because

- Stakeholders often don't know what they want from the computer system
- Stakeholders naturally express requirements in their own terms and with implicit knowledge of their own work
- Different stakeholders have different requirements, which they express in different ways
- Political factors may influence the requirements
- The economic and business environment in which the analysis takes place is dynamic → requirements may change during the project

Sommerville, Software Engineering, 8:th ed, Addisson-Wesley, 2007

23

Conclusion

- Large-scale development require processes, organizations and tools that can cope with increasing complexity
- Most of you will work in
  - Large companies
  - Small- or medium-size companies that deliver software or systems to large companies
- Your engineering skills depend also on
  - How you can combine technology and economics / organization understanding / business process understanding
  - Work in teams and with big organizations

24

Discussion
(collection of typical problems)

For each slide, identify/discuss

- Underlying problem/issue/area
- Is this a realistic/real problem?
- What process improvements could be made?

25