Lecture 21: Process Improvement

Basics of Process Modeling

background in industrial process improvement and statistical control

definitions

Managing Process Change

The quest for continuous process improvement

Humphrey's Capability Maturity Model (CMM)

Towards Zero-Defect Software

lessons from NASA's Software Engineering Lab

Basics of process modeling

Software Process

- "the collection of related activities, events, mechanisms, tasks, and procedures seen as a coherent process for production of a software system to meet a given need"
- "Software processes are software too!"

Benefits of explicitly modeling the process:

- improved communication among team
- process reuse: successes can be repeated
- process improvement: can ensure lessons learnt are incorporated after each project
  - A software development project has two main outputs: a product and some experience
  - The experience is often thrown away
  - Individuals may remember and apply the lessons (but individuals move on, or don't have the authority to change things)

Underlying principle

- Fix the process not the product

Source: *Adapted from Blum, 1992, p473-479. See also van Vliet, 1999, sections 6.3 and 6.6*

Background

Industrial Engineering

- Product Inspection (1920s)
  - examine intermediate and final products to detect defects
- Process Control (1960s)
  - monitor defect rates to identify defective process elements & control the process
- Design Improvement (1980s)
  - engineering the process and the product to minimize the potential for defects

Deming and TQM

- Use statistical methods to analyze industrial production processes
- Identify causes of defects and eliminate them
- Basic principles are counter-intuitive:
  - in the event of a defect (sample product out of bounds)…
  - …don't adjust the controller or you'll make things worse.
  - Instead, analyze the process and improve it

Adapted to Software

- No variability among individual product instances
- All defects are design errors (no manufacturing errors)
- Process improvement principles still apply (to the design process!)

Process Modeling & improvement

Process Description

understand and describe current practices

Process Definition

Prescribe a process that reflects the organization's goals

Process customization

adapt the prescribed process model for each individual project

Process enactment

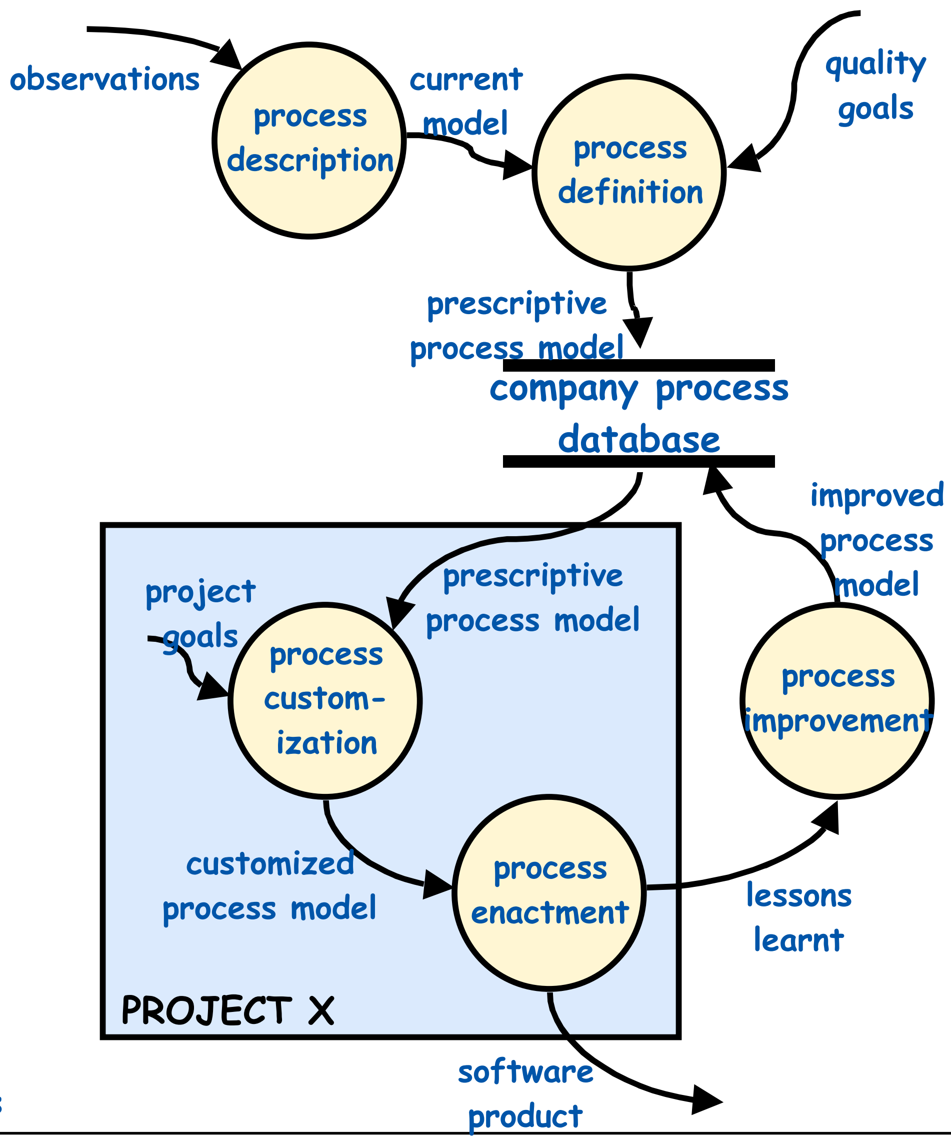
Carry out the process

- I.e. develop the software!
- collect process data

Process improvement

use lessons learnt from each project to improve the prescriptive model

- e.g. analyze defects to eliminate causes



Managing Process Change

Source: Adapted from Humphrey, 1989, chapter 1.

Humphrey's principles:

- Major changes to software processes must start at the top
  - … with senior management leadership
- Ultimately everyone must be involved
- Effective change requires a goal and knowledge of the current process
  - you need a map
  - you need to know where you are on the map!
- Change is continuous
  - process improvement is not a one-shot effort
- Software process change will not be retained without conscious effort and periodic reinforcement
- Software process improvement requires investment

Software Engineering Process Groups (SEPGs)

- Team of people within a company responsible for process improvement
  - identifies key problems, establishes priorities, assigns resources, tracks progress, etc.
- Needs senior management support

Capability Maturity Model

Source: *Adapted from Humphrey, 1989, chapter 1. See also van Vliet, 1999, section 6.6.*

level	characteristic	key challenges
5. optimized	improvement fed back into process	identify process indicators, "empower" individuals
4. managed	(quantitative) measured process	automatic collection of process data, use process data to analyze and modify the process
3. defined	(qualitative) process defined, institutionalized	process measurement, process analysis, quantitative quality plans
2. repeatable	(intuitive) process dependent on individuals	establish process group, identify a process architecture, introduce SE methods and tools
1. initial	ad hoc, chaotic, no cost estimation, planning, management	project management, project planning, configuration management, change control, software quality assurance

Towards Zero Defect Software

Cannot test-in software quality

- testing or inspection cannot improve the quality of a software product
    - (by that stage it is too late)

Defect removal

- Two ways to remove defects:
    - fix the defects in each product (i.e patch the product)
    - fix the process that leads to defects (i.e. prevent them occurring)
- The latter is cost effective as it affects all subsequent projects

Defect prevention (from Humphrey)

- Programmers must evaluate their own errors
- feedback is essential for defect prevention
- there is no single cure-all for defects (must eliminate causes one by one)
- process improvement must be an integral part of the process
- process improvement takes time to learn

SEL Experience Factory

NASA Goddard's Software Engineering Lab (SEL)

- 20 years of measurement and evaluation of software processes
- Large baseline of experience accumulated

Card's conclusions:

- Software engineering without measurement is not engineering
- A software enterprise can collect too much data
  - data collection is expensive (e.g. 5%-10% of development cost)
  - need to know why you're collecting data before you collect it
- Software science models do not appear to have practical usefulness
  - Halstead's, McCabe's complexity models based on theory, not practical utility
- Standards that arbitrarily limit module size seem to be ill-advised
- Information hiding is more important than reducing 'bad' forms of coupling
- Productivity numbers are often crude and may be misleading
  - because they don't distinguish between necessary and unnecessary code
- Delivered source lines of code is not a good measure of work output
- Standards are often too comprehensive
- Unique projects can still be measured against themselves
  - measurement is for controlling and improving, not for comparing projects
- Test coverage is a vital but seldom used measure
- Unreferenced variables are a good indicator of trouble
- Measurement makes productivity and quality improvement meaningful

References

van Vliet, H. "Software Engineering: Principles and Practice (2nd Edition)" Wiley, 1999.

van Vliet gives an overview of quality management practices in chapter 6. He covers ISO 9001 as well as CMM, and has a brief comparison with other international standards including IEEE Std 730, BOOTSTRAP, and SPICE. He also does a nice summary of quality, quoting appropriately from "Zen and the Art of Motorcycle Maintenance" (see lecture 12!). van Vliet also then segues into software measurement issues, including a cute analogy on the uses of measurement, in which we observe that black cows produce more milk than white cows, and use this to conclude that we should paint all the cows black. This summarizes very nicely what quality process management ends up doing when applied blindly!

Humphrey, W. S. "Managing the Software Process". Addison-Wesley, 1989.

This book set out many of the central ideas of process management and process improvement. Humphrey describes the capability maturity model (CMM) in detail. Chapter 1 of this book (in which the CMM is first introduced) is included in the course readings. Of course, Humphrey was one of the main inventors of CMM, and hence he doesn't cover any of it's weaknesses, but van Vliet gives a more balanced coverage.

Blum, B. "Software Engineering: A Holistic View". Oxford University Press, 1992.

Section 6.2.5 is a very sensible overview of process improvement. Since Blum's book was written, the CMM and its variants have been widely adopted across the industry, promoted by the US DoD-sponsored Software Engineering Institute (SEI). The CMM has been elaborated with detailed "key process areas" (KPAs), which have come to be seen as attainment targets by companies, and the CMM has been used as a way of assessing a company's software quality. Unfortunately, along the way, much of the important insights have been lost; very few authors understand what is important about process improvement as well as Blum does.