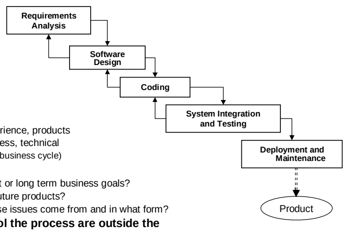
OMSE 551: Week 2
Process Engineering I

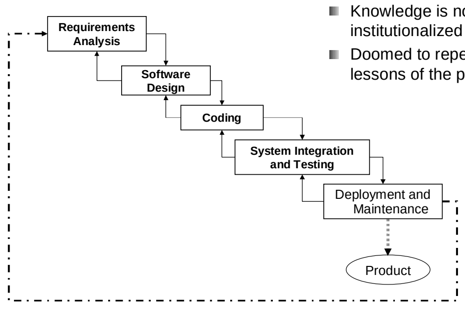
- Celsius-Tech Discussion
- Why study processes?
- Process improvement
- Process definition and modeling
- Process as product

Review: We Only Look at Part of the Problem



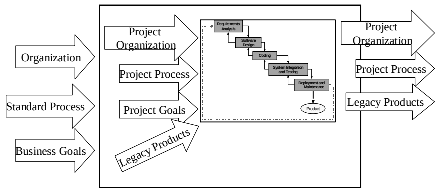
- Know that choices here
  - Are affected by prior choices, experience, products
  - Will affect future capabilities: business, technical
    - E.g., architecture (architectural business cycle)
- Omits critical **context information**
  - How do the activities relate to short or long term business goals?
  - How do the decisions here affect future products?
  - Where does information about these issues come from and in what form?
- **Factors affecting ability to control the process are outside the scope**
  - **Factors <u>affecting</u> the process are implicit**
  - **Factors <u>affected by</u> the process are nowhere evident**

Consequence:
Merry-Go-Round of Sequential Development



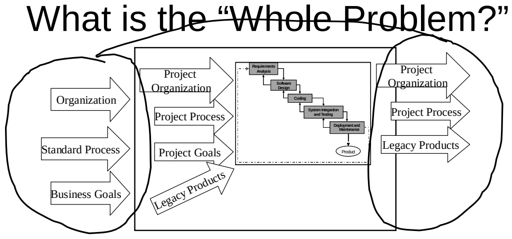
- Knowledge is not institutionalized (tacit)
- Doomed to repeat lessons of the past

What is the “Whole Problem?”



- **Typically treat product development as a *distinct concern***
  - Each development is treated relatively independently from other developments
  - Products and processes developed to meet project-only goals
- **Typically neither is nor should be a distinct concern**
  - Everything has a context that cannot be ignored
  - Necessarily acquires people, products, process, organization from context
  - Influences organization, process, products downstream

What is the “Whole Problem?”



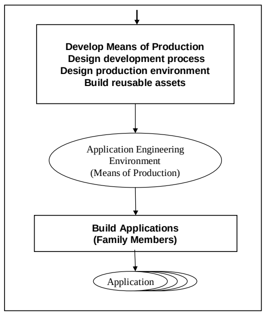
- Relationship between inputs and outputs is not controlled
  - Actually a feedback loop over time
  - Result is disconnect between inputs and outputs
  - Output properties are inconsistent with long term goals
- Developing strategically requires closing the loop (I.e. exercising control across development cycles)

Discussion: Lessons from Celsius-Tech

- What had to change in their approach?
- How did the products change?
  - Kinds of products produced
  - Quality measures?
- How did the process change?
  - Inputs? Outputs? Goals? Measures of goodness?
- How did the organization change?
  - Roles?
  - Organizational structure?
  - Relationships?

Means of Production as Product

- A product-line development strategy sets out to *maximize reuse* over a family of similar software systems.
  - Reuse is planned and systematically executed
  - Common assets are developed as a distinct product
  - The process focuses on developing means of production before any particular product


Develop Means of Production
Design development process
Design production environment
Build reusable assets
Application Engineering Environment (Means of Production)
Build Applications (Family Members)
Application

A Product-Line Process

Products: Reuse in Context

- Success comes from reusing context, not just components

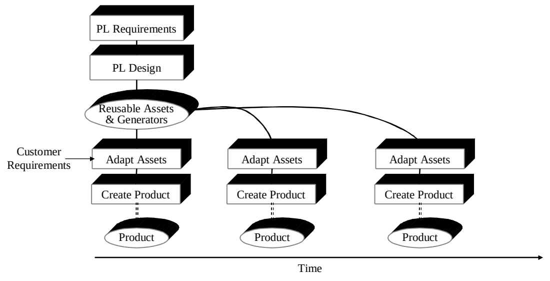
"The unit of reuse is a system function, a thread of related functionality that comprises elements from different layers in the architecture.

System functions are pre-integrated—that is, the components they comprise have been assembled, compiled together, tested individually, and tested as a unit.

When the system function is checked out of the asset repository, it is ready for use. In this way, CelsiusTech is not only reusing components, it is also reusing the integration, component test, and unit test effort that would otherwise have to be needlessly repeated for each application."

- Did not require new technology

Product-Line Development Over Time



CT Project Organization (Original)

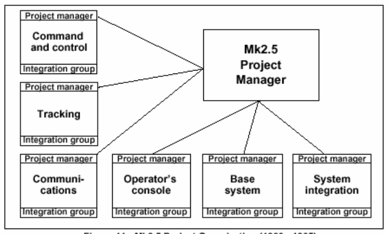


Figure 11: Mk2.5 Project Organization (1980 - 1985)

- Typical “stovepipe” organization
  - Disconnected from business goals, future plans
  - No structure associated with strategic view

Product-Line Development

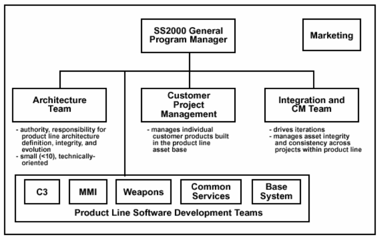

SS2000 General Program Manager
Marketing
Architecture Team
- authority, responsibility for product line architecture definition, integrity, and evolution
- small (<10), technically-oriented
Customer Project Management
- manages individual customer products built in the product line asset base
Integration and CM Team
- drives iterations
- manages asset integrity and consistency across projects within product line
C3
MMI
Weapons
Common Services
Base System
Product Line Software Development Teams

Figure 12: SS2000 Organization, 1987 - 1991

- Initial product line organization
  - Now have organizational/technical structures responsible for strategic view and products (dominated by architecture)
  - Note that Marketing now appears since they must be involved in decisions

Advance P-L Organization

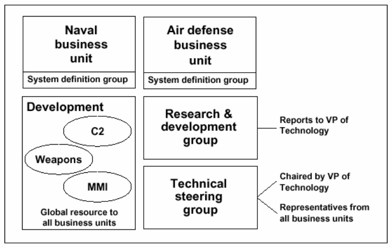
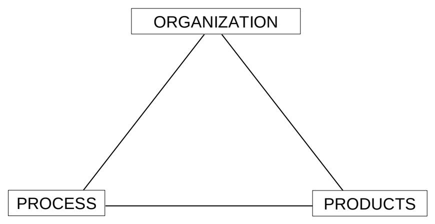

Naval business unit
System definition group
Air defense business unit
System definition group
Development
C2
Weapons
MMI
Global resource to all business units
Research & development group
Reports to VP of Technology
Technical steering group
Chaired by VP of Technology
Representatives from all business units

Figure 13: SS2000 Organization Since 1994

- Organization by market
  - Now long-term business goals actually drive the structure
  - Organizations units own goals in specific domains

All Three Had to Change



- How must each reflect the others? (pairwise)
- How can each impede the others?
  - e.g could the first organization have implemented the product line?
- Success requires alignment (consistency)

Software Process Engineering

- Perspectives on current views of software processes
- The need to specify ideal ("rational") processes
- Why should we view processes as products?

Celsius Tech Lessons

- Observe that PL approach required significant process changes
  - Q1: What process-related activities were required?
    - What processes were created?
    - How were they communicated? Enforced?
    - How are processes evolved to meet changing needs without disrupting the PL?
  - Q2: What skills or capabilities are needed?
    - Process development? Verification? Communication?
- Implications =>
  - Strategic software engineering requires process development skills
  - Suggests need for different view of processes (process as product)

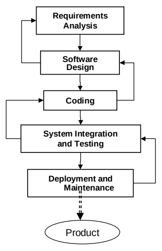
What is a “software process”?

- [Webster’s] – “A series of actions or operations proceeding to an end.”
- [Paulk, et al] – a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products…”
- [Weiss, Lai] – “A process is a sequence of decision-making activities. For any engineering process, the engineers need to know what decisions they can make, when they can make them, what the results mean, and how they should be represented.”†

  † We will use this one for 551

Original Rationale for Process View

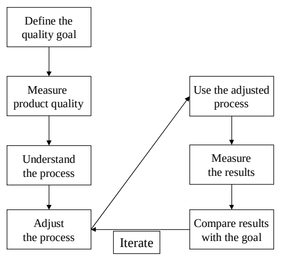
- Developed as a tool for gaining and maintaining control over complex software development efforts
- Application of “divide-and-conquer” to software development activities and products
  - Identify distinct phases of development and distinct products
    - E.g., Requirements phase – understand the problem to be solved
    - Product – Software Requirements Specification
  - Assumption: simpler to address each phase separately
    - E.g., Elicit, specify, and validate requirements before doing design
    - True to the extent dependencies between phases and products are limited (same as for modules)
    - Since “coupling” between process phases is actually high there are inherent problems in this assumption



Recent Focus: “Process Improvement”

- Why focus on “process improvement”?
- Software problems characterized by lack of control
  - Development typically chaotic: unpredictable in budget, schedule, and result
  - Process is poorly defined
  - Unclear what to change to improve the result!
- Desire to make software development *systematic*
  - Thorough, methodical, regular, and predictable
  - So how do we get from “chaotic” to “systematic”?
- Implied solution:
  - Must first understand the current process
  - Must determine where it’s going wrong
  - Then can revise to address problems

A Simple Process-Improvement Process



Important Points

- Quality goal and iteration test are defined in terms of the *product* (not the process)
- Must *understand* the process before we can effectively adjust it.
- Measurement is done on the *product* (not the process)

From *Introduction to the Personal Software Process*, W. Humphrey
(borrowed from manufacturing process improvement)

Process vs. Product Improvement

- Are we trying to improve the process or the product?
- Role of product is obscured in Capability Maturity Model (CMM)
- “Process maturity is the extent to which a specific process is explicitly defined, managed, measured, and effective.” [Paulk]
  - Improvement defined in terms of predictability, control, and effectiveness
  - Where Effective is define as: “practiced, documented, enforced, trained, measurable, improvable”
  - Focus of measurement is on the process (not product)

Process vs. Product Improvement (2)

- Upshot: Process improvement and product improvement are related but distinct concerns
- We can improve a product without changing the process
  - E.g., use better designers and programmers
- We can improve a process without changing the product
  - Produce same product in less time or for less money
  - Produce same product with process that is better defined, measured, more predictable, etc.
- We can also change a process to improve product quality

Process vs. Product Improvement (3)

- Need to be clear on the goal – the goal determines what needs to be measured
  - To improve the product, must measure product quality
  - To improve the process, must measure process quality
- For software, measuring product qualities is (often) harder than measuring process qualities
  - Why? Examples?
  - Leads to overemphasis on process definition and measurement
  - Heisenberg applies!
  - May result in process-improvement-death-spiral
- Some simple truths (oft forgotten)
  - A software development process exists *only* to produce a software product
  - What you measure is what you get
  - ***If it's not worth doing, it's not worth doing well!***
    - Improving a process that produces junk just does it more efficiently

Relation to SSE

- Benefits of CMM/Process Improvement
  - Previously had *no* mechanism for judging process quality or setting a baseline for process improvement
  - It is inherently strategic in view
    - Strategic in looking at process as *institutional asset*
    - Strategic in sense that it spans development cycles
- Drawbacks
  - Focus on process alone often does not result in product improvement
  - Assumes a conventional life cycle
  - Assumes process improvement by increments:
    - But, we can't get from tactical view to strategic one by incremental improvements

➔"Process improvement" view is necessary but not sufficient for SSE

Process Specification and Modeling

Why Specify Processes?

- We must first understand a process to improve it
- Representing a process is a necessary step toward understanding (and communicating)
  - Software processes are complex
  - Really helps to visualize it to work with it
  - Must be communicated to many different stakeholders
- Process specification can serves variety of purposes
  - Provides a standard metric for development progress (a yardstick)
  - Provides guidance for enactment (a map)
  - Provides baseline for process improvement or tailoring
  - Criterion for contract award (e.g., CMM level)

Contents of a Process Specification

- Details depend on the purpose of the specification
- In general, contents should answer [Parnas &Clements]:
  - What product we should work on next?
    - Equivalently – what decision(s) must we make next?
  - What kind of person should do the work?
  - What information is needed to do the work?
  - When is the work finished?
  - What criteria must the work product satisfy?
- In personal terms, answers the questions:
  - Is this my job?
  - What do I do next?
  - What do I need to do the work?
  - Am I done yet?
  - Did I do a good job?

Qualities of a Good Process Spec

- Think about how it will be used
  - Guide the process
  - Measure progress
  - Basis for process improvement, modification, tailoring
- Reasonable analogy to an SRS (or any technical spec)
  - What properties would you want?
  - E.g., if you were going to try to emulate Celsius Tech's process in your organization?

Ideal Process Description

- Ideal: Process is like a program
  - Can be followed systematically
  - Result is predictable and inevitable
- …but, people aren't machines
  - Steps cannot be described with complete precision
  - Real processes are neither sequential nor deterministic
  - Things change over time
  - People make mistakes
  - People compromise, adjust, make do, fill in, muddle through, soldier on … where machines fail
- Implication: any (useful) process description we write down is the description of an ideal, not a real process.

Desirability of "Faking it"

- Thesis: It is nonetheless useful to "fake" a rational design process [Parnas & Clements]
  - Endeavor to describe the ideal process (not the real one)
  - Follow the ideal process as closely as possible
  - Write (rewrite) the documentation and other work products as if we had followed the ideal
- Rationale
  - Idealized process can provide guidance
  - Helps come closer to the ideal (emulation)
  - Helps standardize the process (provide a common view of how to proceed and what to produce)
  - Provides a yardstick for assessing progress
  - Provides better products (e.g. final draft not first)

Abstraction and Modeling

- An ideal process is an *abstract* process
  - Recall that abstraction = = one-to-many
  - Many possible ways of getting the work done satisfy the ideal
    - E.g., different sequencing of activities
    - Another reason for looking at products (process independent)
  - Abstracting allows us to document and understand what a set of related processes have in common
- Use the term "modeling" to describe the capability to construct specifications of abstract processes
  - Modeling languages provide constructs for describing the parts of a process (roles, activities, products, sequencing, etc.)
  - Formalizing the syntax permits tool support
  - Formalizing the semantics permits analysis

Process as Product

Contrasting Views of Process

- Process improvement view (e.g., CMM)
  - We have a process that we want to improve
  - Must understand it, document it, measure it, then change it
  - Result is a more mature process for developing software products
- Process as product view (SSE)
  - Not all developments are alike so not all processes are alike
    - That's why we have many process models (e.g., XP, SCRUM, spiral, waterfall)
  - Each development effort needs a process appropriate to its goals and constraints (examples?)
    - Sometimes we can tailor an existing process
    - Sometimes we need a new kind of process
  - Implication: *Processes are artifacts we produce too!*

Implications for Developers

=> Processes must be treated as first-class products

1. We need to be process developers as well as product developers
2. We need skills in creating, specifying, and analyzing processes
3. We need a *process development process*
4. Process development needs the same organizational support as product development
   - Resources
   - Management
   - Requirements, etc.
5. We may need tool support for tedious, error-prone tasks (specify, analyze, change, tailor, track, etc.)

Summary

- To implement strategic software development, need to gain and maintain control over development of software processes.
- Current approaches to process improvement are inadequate
- Writing abstract specifications of processes provides a vehicle to design, verify, represent and communicate process development decisions
- Implies that processes should be treated as first-class products (like code or other core assets)

Next Week

- Assignment
  - Blackboard discussion
  - Set up for exercise
    - Instructions will be on web page
    - Need SVN client (e.g., TortoiseSVN)
- Lecture
  - From web, links on calendar page
  - Skip the part on the Eclipse process tool unless you are just interested

Questions?