Umeå University
Department of Computing Science

Programvaruteknik vt15

Configuration Management, Maintenance and Support

Jonas Andersson

http://www8.cs.umu.se/kurser/5DV151/VT15/

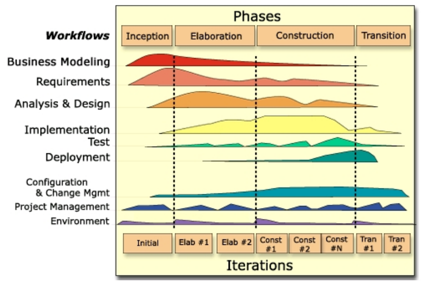
Last time

- Implementation
  - Good Implementation – Clean Code
  - Complete + Correct
  - Traceable
  - Performant
  - Readable + Maintainable
- Quality Assurance
  - Produce High Quality Software
    - Quality
    - Ready to use

Today

- Software Configuration Management
- Maintenance

The bigger picture



Software Configuration Management

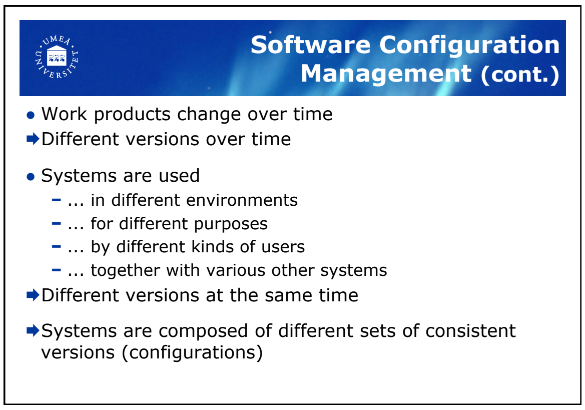
Process of managing artifacts

The purpose of SCM is to establish and maintain the integrity of artifacts / work products.

Activitites:
- Understand which artifacts to manage
- Define framework; naming model, storage/access
- Decide which tools to use
- Ensure process adoption

Software Configuration Management

- Example Artifacts/work products
  - Plans
  - Process descriptions
  - Requirements
  - Specifications
  - Designs
  - Drawings
  - Code
  - Test Cases
  - Documentation
  - Tools
  - Setup scripts
  - Deployment scripts
  - etc…



Software Configuration Management (cont.)

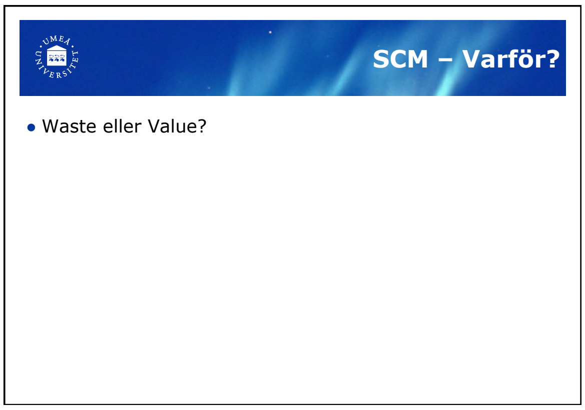
- Work products change over time

➡Different versions over time

- Systems are used
  - … in different environments
  - … for different purposes
  - … by different kinds of users
  - … together with various other systems

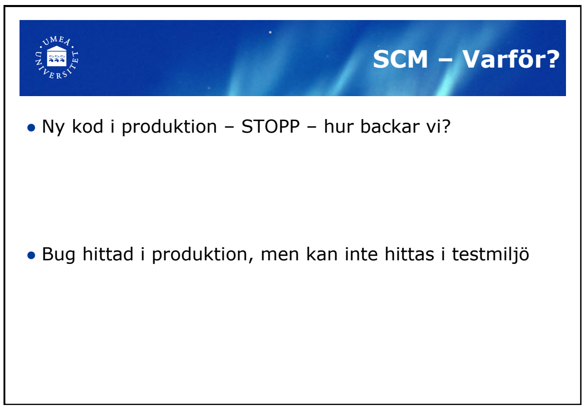
➡Different versions at the same time

➡Systems are composed of different sets of consistent versions (configurations)



SCM – Varför?

- Waste eller Value?



SCM – Varför?

- Ny kod i produktion – STOPP – hur backar vi?

- Bug hittad i produktion, men kan inte hittas i testmiljö

SCM – Varför?

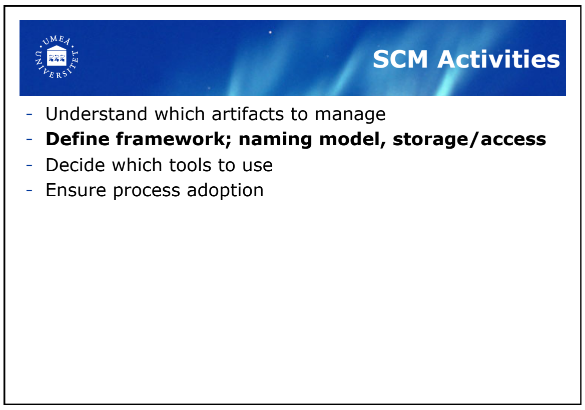
- En fysisk maskin i produktion dör, hur snabbt får vi upp ny?
  - "Tryck på knappen"?
  - En veckas manuellt jobb?
- Hur lång tid tar det att sätta upp utvecklingsmiljö på egen maskin?

SCM – Varför?

- Du jobbar i samma kodbas som annan utvecklare. Varje gång hen har checkat in ny kod så förstörs formattering.
- En ny designer läser in skisser i ny version av Super Designer. Plötsligt kan ingen annan läsa in skisserna…

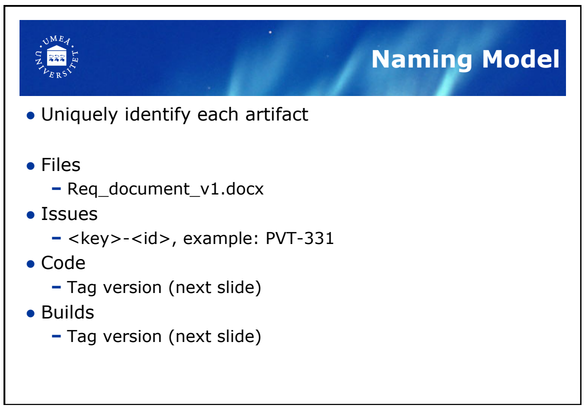
Which artifacts?

- Behöver vi total spårbarhet, audit trails, för varje artefakt? Dvs att kunna se exakt vad vem har gjort för varje skapad artefakt?
- Konservativt företag, försvarsindustri etc: Ja!
- Litet företag eller stor agil organisation: Nej!



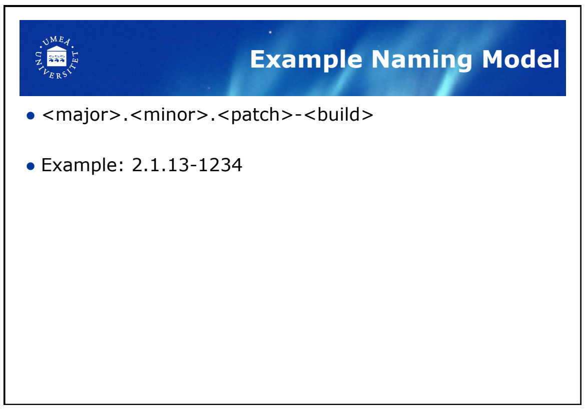
SCM Activities

- Understand which artifacts to manage
- **Define framework; naming model, storage/access**
- Decide which tools to use
- Ensure process adoption



Naming Model

- Uniquely identify each artifact
- Files
  - Req_document_v1.docx
- Issues
  - <key>-<id>, example: PVT-331
- Code
  - Tag version (next slide)
- Builds
  - Tag version (next slide)



Example Naming Model

- <major>.<minor>.<patch>-<build>
- Example: 2.1.13-1234

Storage and Access Model

- Naive: Separate files for each version
- Version handling by numbering schemes

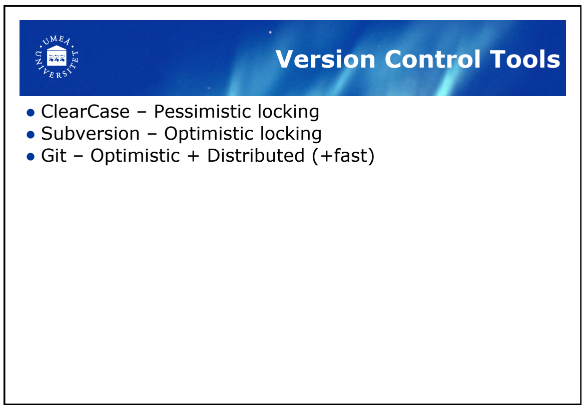
- Takes much storage
- Lots of manual work

Storage and Access Model

Store in central Repository
Add check-in/check-out mechanism
- Work on local copy
- Add changes via linear deltas

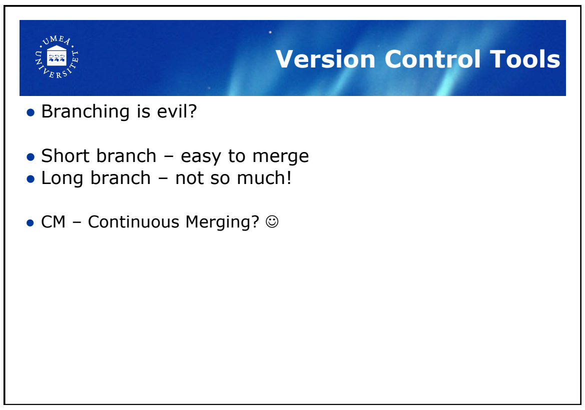
Tools for Storage and Access Model

- Version Control:
  - History
  - File comparing
  - Modification tracking
  - Control of development branches
  - Efficient storage and retrieval
  - Access control
  - Merging versions
  - Pessimistic vs Optimistic locking
  - …



Version Control Tools

- ClearCase – Pessimistic locking
- Subversion – Optimistic locking
- Git – Optimistic + Distributed (+fast)

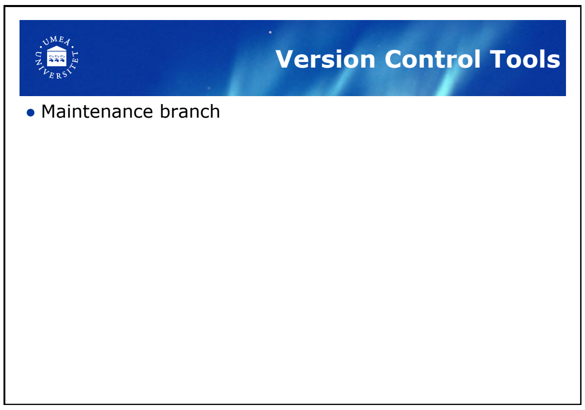


Version Control Tools

- Branching is evil?

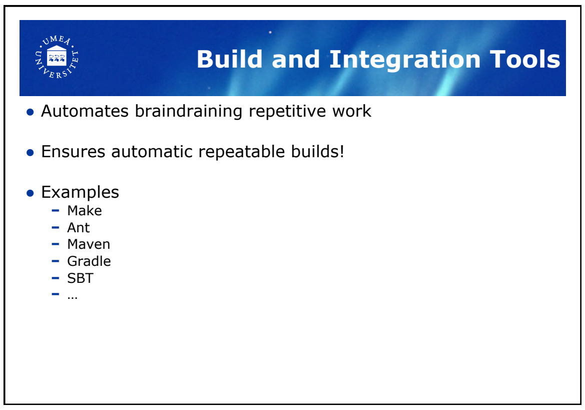
- Short branch – easy to merge
- Long branch – not so much!

- CM – Continuous Merging? ☺



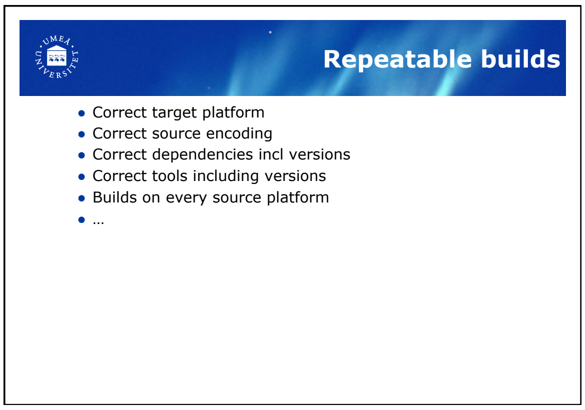
Version Control Tools

- Maintenance branch



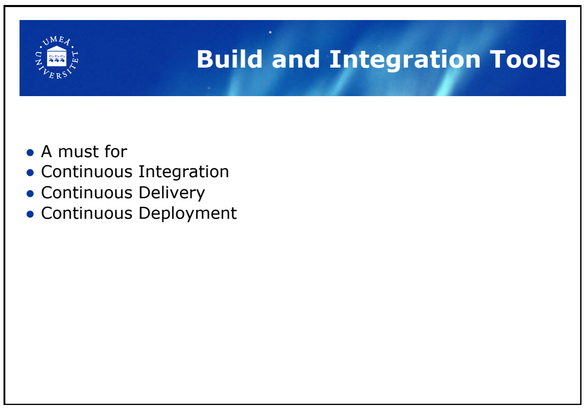
Build and Integration Tools

- Automates braindraining repetitive work
- Ensures automatic repeatable builds!
- Examples
  - Make
  - Ant
  - Maven
  - Gradle
  - SBT
  - …



Repeatable builds

- Correct target platform
- Correct source encoding
- Correct dependencies incl versions
- Correct tools including versions
- Builds on every source platform
- …



Build and Integration Tools

- A must for
- Continuous Integration
- Continuous Delivery
- Continuous Deployment

Maven

- Maven coordinates

  -
    ```
    <groupId>org.umu.pvt</groupId>
    <artifactId>project-war</artifactId>
    <version>2.4.7-SNAPSHOT</version>
    ```
  - `<packaging>jar</packaging>`

-
  ```
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.1</version>
  </dependency>
  ```

- mvn clean install
  - Cleans up from earlier builds
  - Compiles all code
  - Runs all tests
  - Creates artifact, example project-war-2.4.7-SNAPSHOT.jar
  - And more!

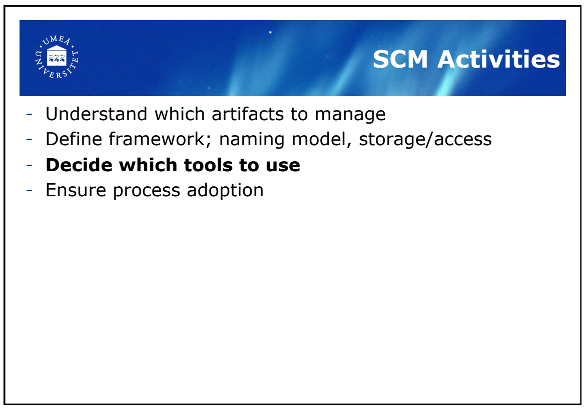
Build and Integration Tools

- The Build Server!
  - Bamboo (Atlassian)
  - Jenkins (OpenSource)
  - TeamCity (JetBrains)

- Essentially
  - Checks out code on change
  - Runs scripts (maven, bash, …)
  - Keeps history
  - Notifies failure/Success

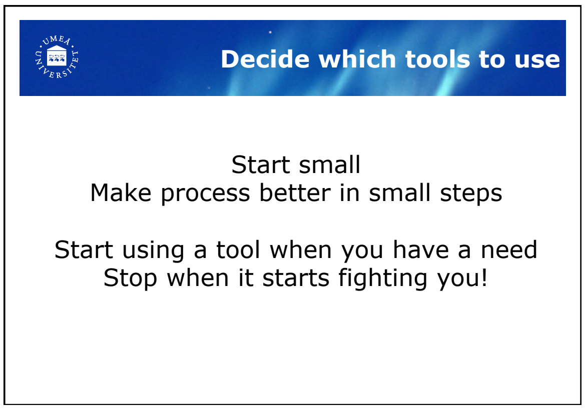
Infrastructure As Code

- System creation – Infrastructure As Code
  - Puppet
  - Chef
  - Vagrant
  - Docker



SCM Activities

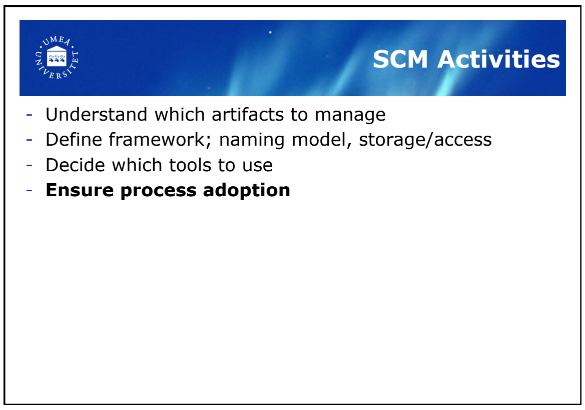
- Understand which artifacts to manage
- Define framework; naming model, storage/access
- **Decide which tools to use**
- Ensure process adoption



Decide which tools to use

Start small
Make process better in small steps

Start using a tool when you have a need
Stop when it starts fighting you!



SCM Activities

- Understand which artifacts to manage
- Define framework; naming model, storage/access
- Decide which tools to use
- **Ensure process adoption**

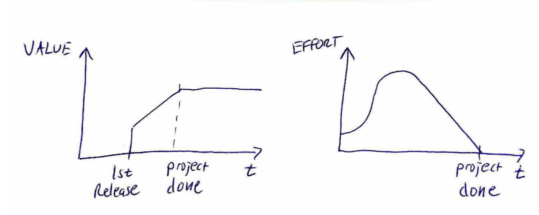
Ensure process adoption

- CM-Polis vs Cross-Pollination

The End

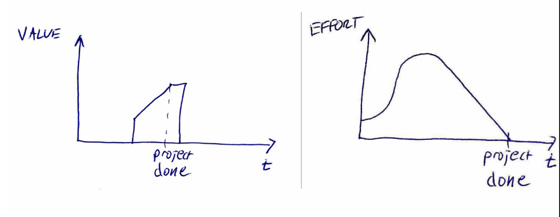
Maintenance

- Vad händer med produkten när projektet är över?



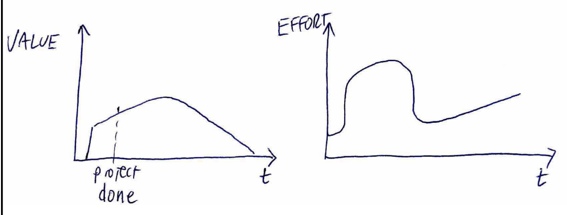
Maintenance

- Vad har hänt här?



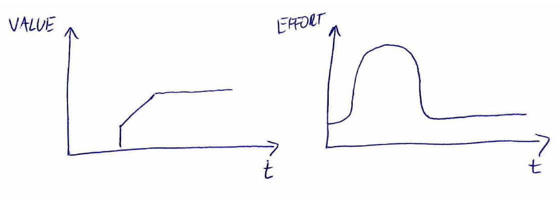
Maintenance

- Här då?



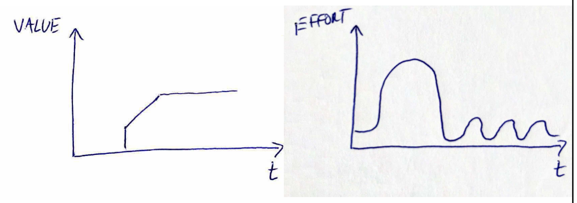
Maintenance

- Eller kanske såhär?



Maintenance

- Eller så här?



Maintenance

- När lägger vi ner produkten?
  (Product portfolio management)

Projektet är klart

- Har vi satt upp en supportgrupp och utbildat den?
- Har vi satt upp en driftgrupp och utbildat den?
- Vad händer med utvecklingsteamet?
- Vem har nu ansvaret?

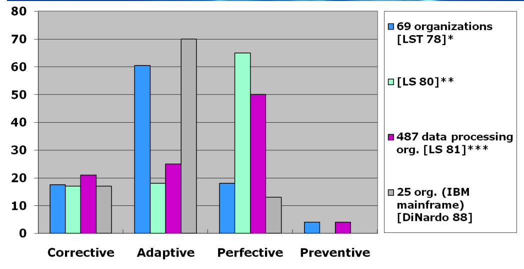
- Överlämningar = Waste!

- Långsiktigt → Tänk produkt, inte projekt!

Types of Maintenance

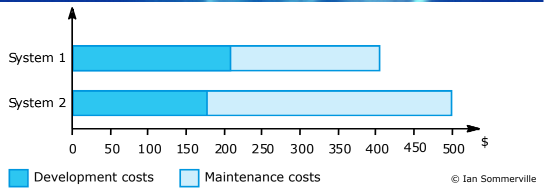
- **Corrective** maintenance; *to repair software faults*
  - Correct deficiencies to meet its (original) requirements
- **Adaptive** maintenance; *to add to or modify the system's functionality*
  - Satisfy new requirements
  - Operate in a different context
- **Perfective and preventive** maintenance
  - Quality improvements without changing the functionality
  - Improving maintainability

Types of Maintenance

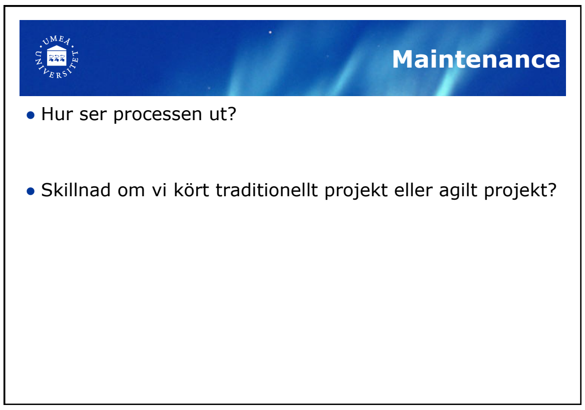


Data from SE textbooks: * [Schach 97], ** [Sommerville 96], *** [Pfleeger 98].

Maintenance is Expensive?

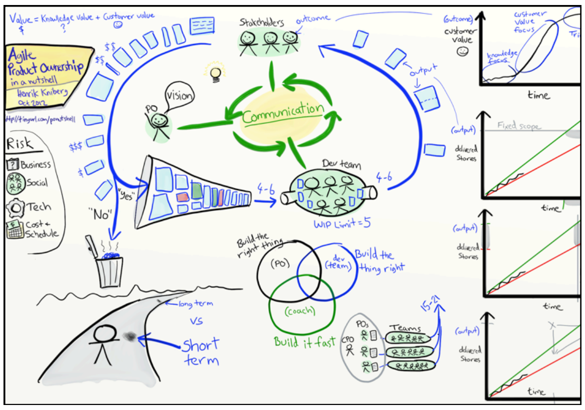


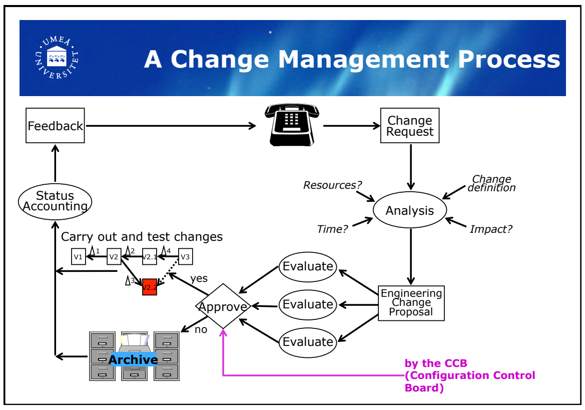
- Maintenance is often much more expensive than development
- The most time-consuming activity is program comprehension (up to 60 % of the total effort)



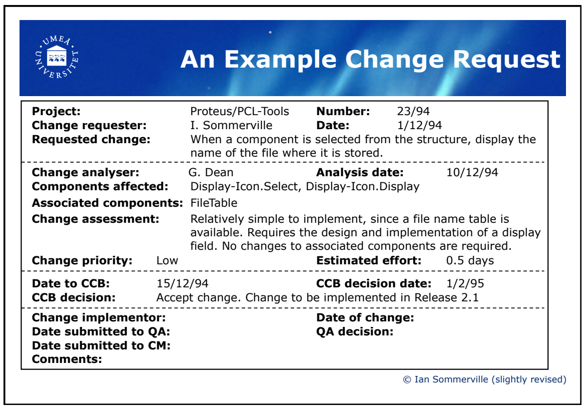
Maintenance

- Hur ser processen ut?
- Skillnad om vi kört traditionellt projekt eller agilt projekt?



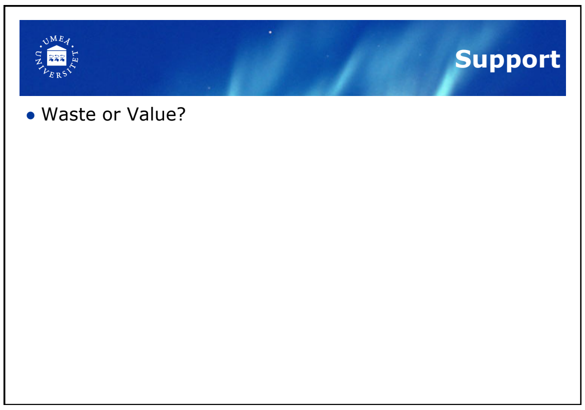


A Change Management Process



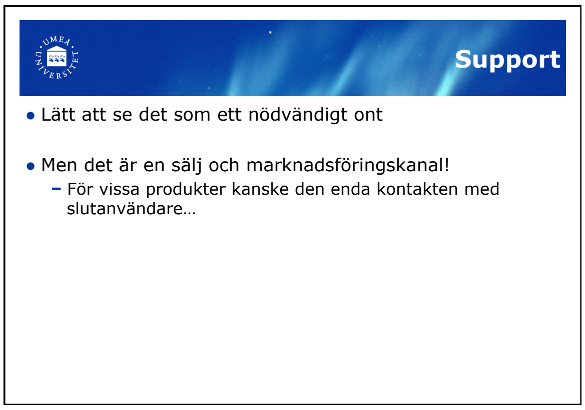
An Example Change Request

Project:	Proteus/PCL-Tools	**Number:**	23/94
Change requester:	I. Sommerville	**Date:**	1/12/94
Requested change:	When a component is selected from the structure, display the name of the file where it is stored.		
Change analyser:	G. Dean	**Analysis date:**	10/12/94
Components affected:	Display-Icon.Select, Display-Icon.Display		
Associated components:	FileTable		
Change assessment:	Relatively simple to implement, since a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.		
Change priority:	Low	**Estimated effort:**	0.5 days
Date to CCB:	15/12/94	**CCB decision date:**	1/2/95
CCB decision:	Accept change. Change to be implemented in Release 2.1		
Change implementor:		**Date of change:**	
Date submitted to QA:		**QA decision:**	
Date submitted to CM:			
Comments:			



Support

- Waste or Value?



Support

- Lätt att se det som ett nödvändigt ont
- Men det är en sälj och marknadsföringskanal!
  - För vissa produkter kanske den enda kontakten med slutanvändare…

Supportkanaler

- Telefon
- Mail
- Formulär
- Forum
- Sociala medier
- App Store / Play reviews
- Etc…

Klassificering

- Emergency?
- Bug?
- Usability problem?
- Missing functionality?

Supportstrategi

- Webb: FAQ, Community, etc – ”Gratis”
- 1st line
  - Icke tekniker mer eller mindre insatt i produkten
  - Försöker svara på så mycket som möjligt; fel, workarounds, etc
  - Filtrer så att 2nd line kan fokusera
- 2nd line
  - Teknisk support; utvecklare e.d.

Support

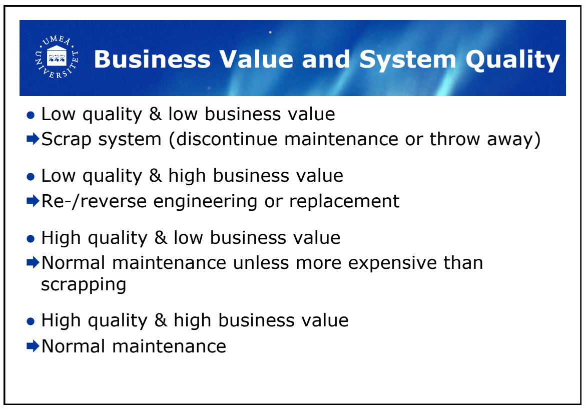
- Lärorikt – alla skulle skulle sitta i 1st line någon gång
  - Utvecklare
  - Projektledare
  - Testare
  - Beställare
  - …

Summary

- Configuration Management
  - A process for managing artifacts
  - Version control, naming model, tools
  - Automate!
- Maintenance
  - Focus on the product
  - Plan for it in good time – minimize handoffs
  - Maybe not so different from development
- Support

Lehman's Laws of Software Evolution

Law	Description
Continuing change	A program must change or become progressively less useful.
Increasing complexity	As a program changes, its structure becomes more complex; extra resources are required.
Large program evolution	System attributes, (e.g., size, time between releases) is ~invariant for each system release.
Organizational stability	A program's rate of development is ~constant.
Conservation of familiarity	The incremental change in each release is ~constant.
Continuing growth	The functionality has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems appear to be declining unless adapted to changing environments.
Feedback system	Evolutionary processes involve feedback systems for product improvement.



Business Value and System Quality

- Low quality & low business value

➡Scrap system (discontinue maintenance or throw away)

- Low quality & high business value

➡Re-/reverse engineering or replacement

- High quality & low business value

➡Normal maintenance unless more expensive than scrapping

- High quality & high business value

➡Normal maintenance