Programvarukonstruktion

Jürgen Börstler
jubo@cs.umu.se
http://www.cs.umu.se/~jubo

http://www.cs.umu.se/kurser/TDBB12/

Course Organisation

- Lecture part
  - Traditional lectures
  - Guest lectures
  - Group exercises
  - Assignments (3)
  - Written examination ("tenta")
- Project part
  - Teamwork
    - GUI design
    - Prototype development
  - Team meetings
  - Oral presentation of results
  - Written reports
- Examination: Assignments + "Tenta" + Project

Contents

- ➩ Introduction
- ➩ Requirements Engineering
- ➩ UI Design
- ➩ Project Management
- ➩ Software Design
- ➩ Detailed Design and Coding
- ➩ Quality Assurance

PVK--HT99 3

➩ Introduction

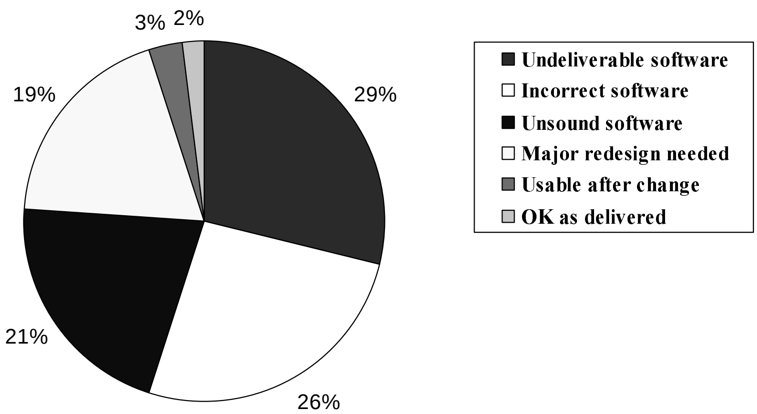
- ➩ What is Software Engineering
- ➩ History of Software Engineering
- ➩ Software Development Processes
- ➩ Software Quality
- ➩ Approaches to Improve Quality

PVK--HT99 4

Software Problems

- Software becomes more and more complex
- Software permeates our daily life
- Software failures may harm our lives
- Software does not meet expectations
- Software projects exceed budgets and schedules
- ...

The Software Crisis is not Over

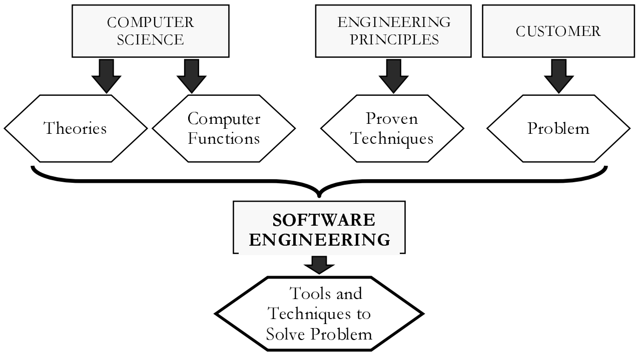


Study from ...

What is Software Engineering?

"The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."

Definition proposed by Fritz Bauer at the NATO conference '68 in Garmisch [NRB 76]



PVK--HT99 7

But ...

" ... we all tell each other and ourselves that software engineering techniques should be improved considerably, because there is a crisis. But there are a few boundary conditions which apparently have to be satisfied. I will list them for you:

1. We may not change our thinking habits.
2. We may not change our programming tools.
3. We may not change our hardware.
4. We may not change our tasks.
5. We may not change our organizational set-up in which the work has to be done.

Now under these five immutable boundary conditions, we have to try to improve matters. This is utterly ridiculous. ..."

Comment by Edsger Dijkstra at the NATO conference '69 in Garmisch [BuRa 70]

PVK--HT99 8

A Little History (1)

1940
Construction of the first computers
Hardware dominates

1950
Single batch programs

1960
More complex programs
Software crisis
- More application domains
- Not enough educated programmers
- No common programming languages
- No special project management
- No quality assurance
- Quality = efficiency
- ...

1968
NATO conference in Garmisch gives “birth” to Software Engineering
- Engineering approach to software production
- Systematic production of large software systems of high quality

A Little History (2)

1970
Life-cycle models
Structured programming and design
General purpose languages
Requirements definition languages
Modularization

1980
Many new methods, languages, and tools
First programming environments
Project management support

1985
Object-orientation, GUIs
Reuse, Re-engineering, ...
Process modeling
Distributed computing

1990
OOA/D, Software architecture, CORBA
Patterns, Internet computing, CMM, PSP

1995
Java, UML
T-PSP, ...

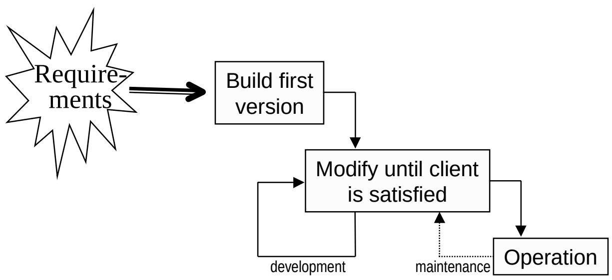
Elements of Software Engineering

- Methods
  Technical “how tos” to support software development tasks
- Languages
  Notations to support methods
- Tools
  (Semi-) automated support for (the usage of) methods and languages
- Processes
  Coordination and management of software development tasks supported by methods, languages, and tools

➨ Economically produce quality software

PVK--HT99 11

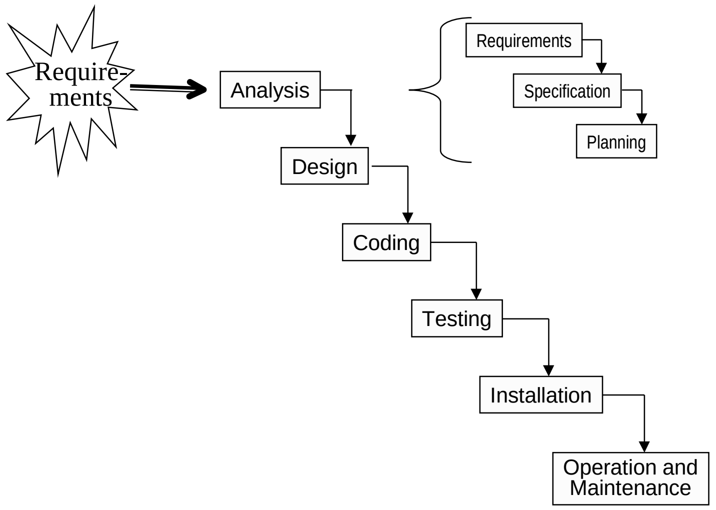
Software Development Processes



➨ Does this scale up?

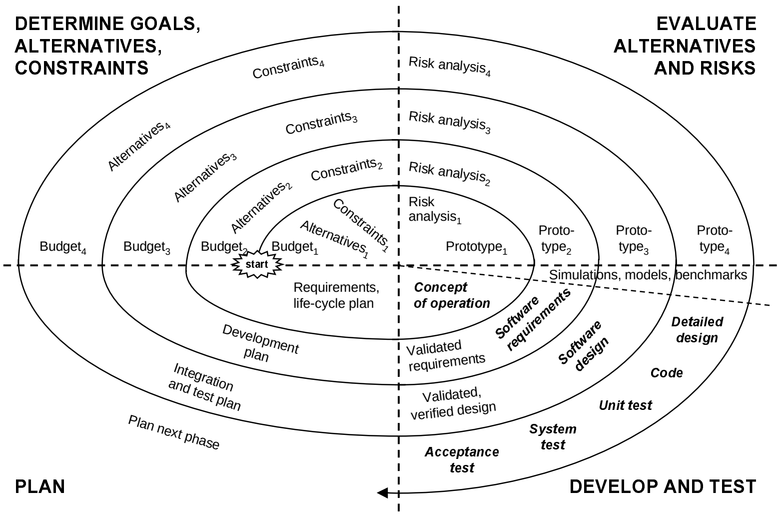
PVK--HT99 12

The Waterfall Model ('70)



PVK--HT99 13

The Spiral Model ('88)

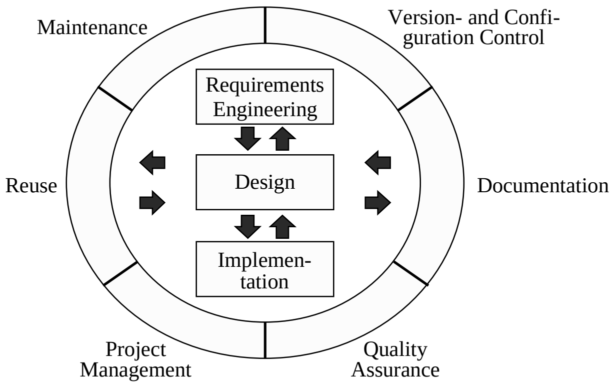


PVK--HT99 14

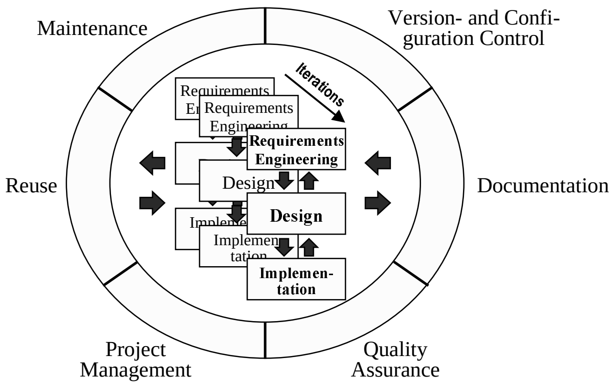
Waterfall vs. Spiral Model

	Waterfall Model	Spiral Model
Model Complexity	Simple, linear sequence of phases	Complex, iterative model; many integrated tasks
Management	Document driven	Risk driven
Quality Control	Natural milestone after each phase	Continuous evaluation, integrated into the model
Customer interaction	No	Prototypes are built and evaluated by customers in every iteration
Risk	High (late feedback)	Low (risk analysis is integrated in the model)
Usability	Small and/or low risk projects	Large projects

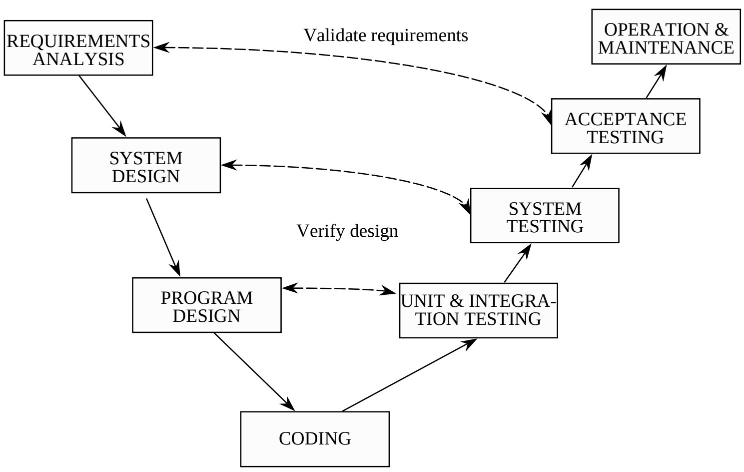
A Generic Process



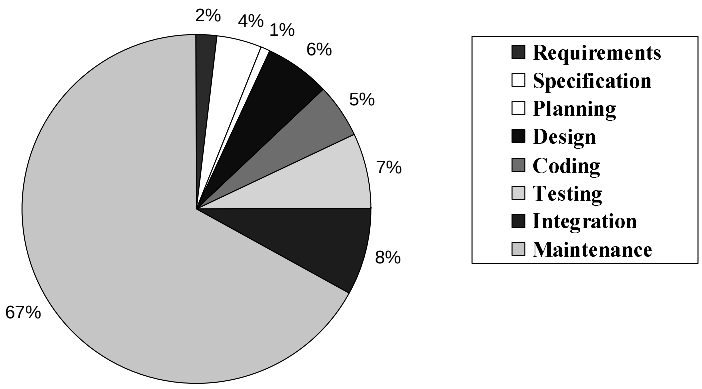
Incremental and Iterative Processes



V Model

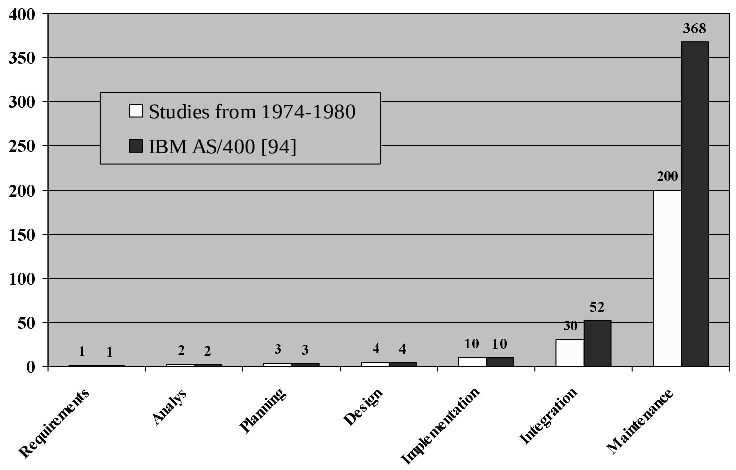


Relative Costs of Development Phases



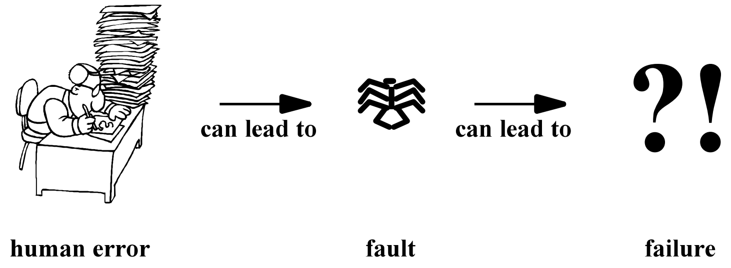
Compiled data from 1976-1981, see [Schach 97].

Relative Costs of an Error

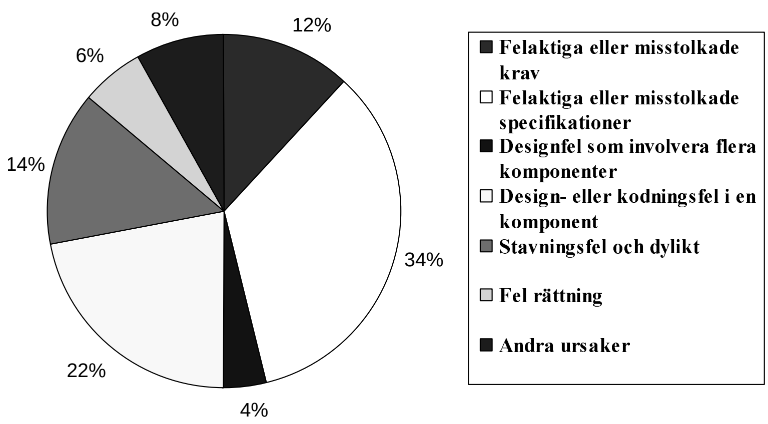


See [Schach 97].

Fault vs Failure



Causes of Errors



Study from 1978, see [GoRu 95].

➩Introduction

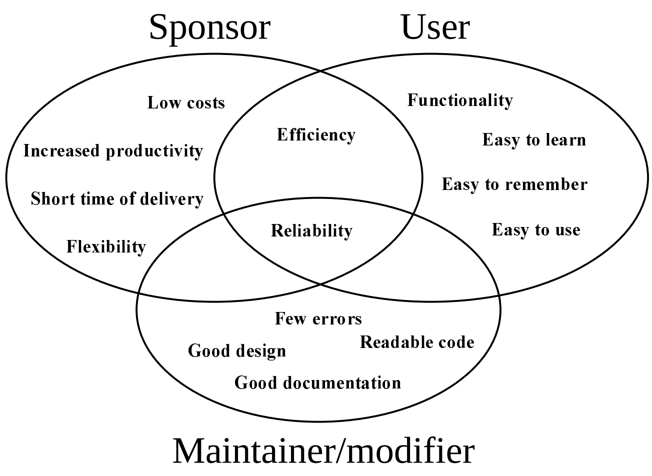
- ➩What is Software Engineering ✔
- ➩History of Software Engineering ✔
- ➩Software Development Processes ✔
- ➩**Software Quality**
- ➩**Approaches to Improve Quality**

Quality is ...

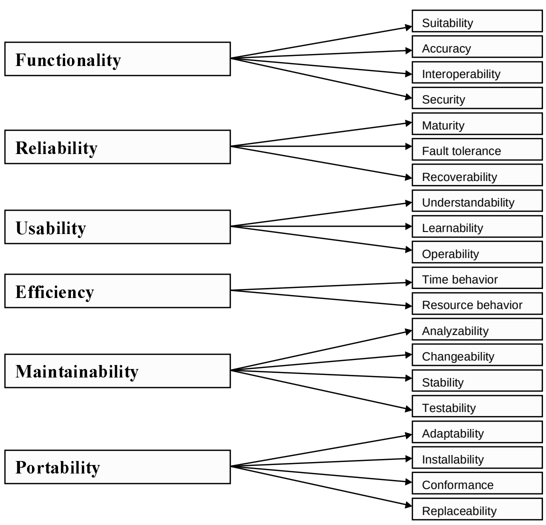
- … I know it when I see it
- … it suits the client/user
- … it conforms to the specification
- … it has some inherent quality
- … it depends on the price

 Copyright © 1997-1999, jubo@cs.umu.se

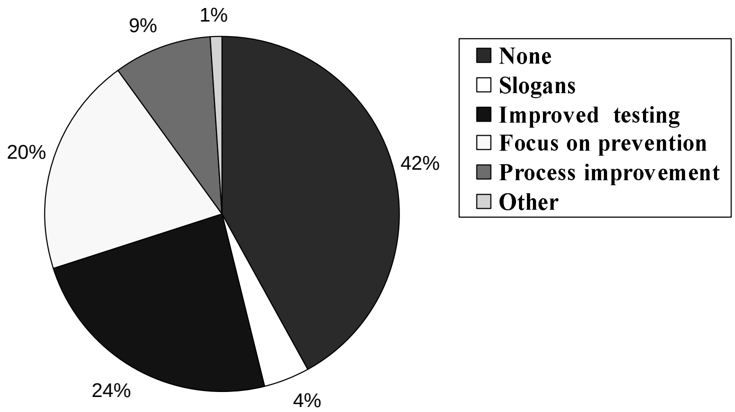
And Quality is …



Quality Factors (ISO 9126)

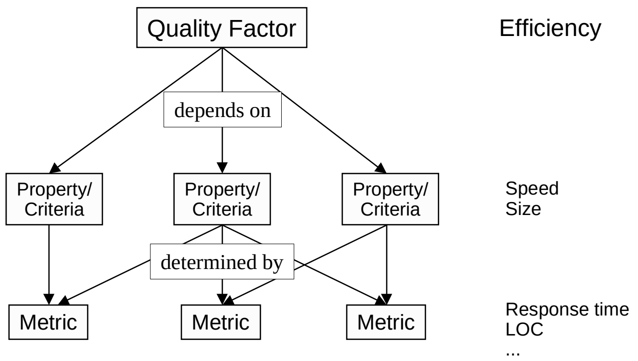


How Companies Pursue Software Quality



A survey from the CASE Research Corporation (1990), see [Yourdon 92].

How To Measure Quality?



- Metrics are (objective) measurements to determine (subjective) quality factors

Some Example Metrics

- To measure efficiency
  - Time behaviour
    - Transactions per second
    - Response time
    - Screen refresh time
  - Resource behaviour
    - KBytes of executables
    - LOC
    - Number of processors
- To measure usability
  - Training time
  - Number of help frames
- To measure reliability
  - MTTF (Mean Time To Failure)
  - Availability
- To measure robustness
  - Time to restart after a failure
  - Probability of data corruption on failure
- To measure portability
  - Number of target systems
  - Percentage of target dependent statements

➨ Measurement is necessary

PVK--HT99 29

Purpose of Measurement

- Analysis: Determine current quality
- Prediction: Predict future quality
- Not only code can be measured, but also
  - Products
    - Documentation
    - Design
  - Processes
    - Analysis phase
    - Test phase
  - Resources
    - Personnel
    - Budget

PVK--HT99 30

Approaches to Improve Quality

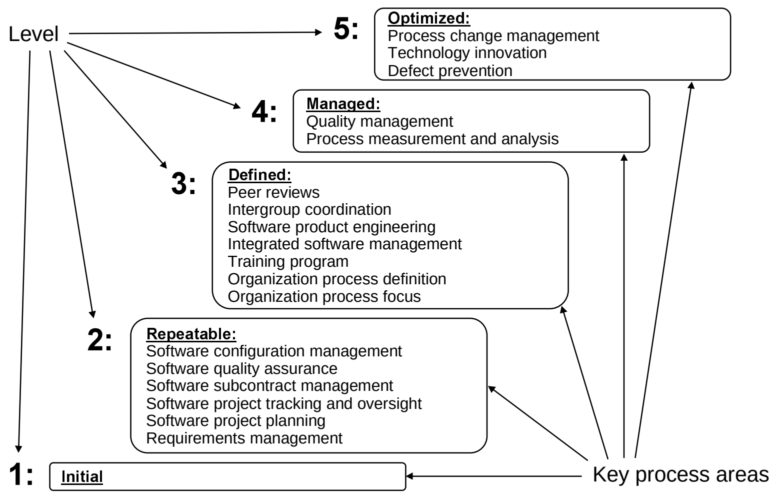
- CMM
- PSP
- Inspections
- Standards (ISO9000, ...)
- Cleanroom engineering
- Statistical quality control
- ...

CMM

- **C**apability **M**aturity **M**odel
- Developed by SEI 1986 (for the DoD)
- Five maturity levels
  - Initial (ad-hoc process)
  - Repeatable (repeatable process)
  - Defined (well-defined, documented process)
  - Managed (predictable process)
  - Optimised (continuous process improvements)

➨ The DoD requires level 3 from all contractors

CMM Overview



PVK--HT99 33

CMM Results

CMM level	Development time	Person months	Faults detected during dev.	Faults delivered and installed	Total dev. costs in US$
1	29,8	593,5	1.348	61	5.440.000
2	18,5	143,0	328	12	1.311.000
3	15,2	79,5	182	7	728.000
4	12,5	42,8	97	5	392.000
5	9,0	16,0	37	1	146.000

Model predictions for the development of a 200.000 LOC data processing product (1993), see [Schach 97].

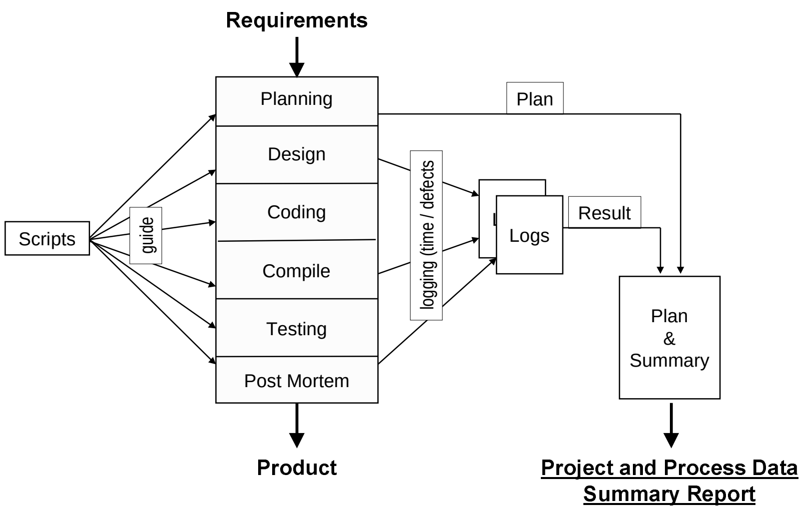
PVK--HT99 34

PSP

- A process for individual developers
  - Well-defined process steps (*scripts*)
  - Forms
  - Instruction for filling in the forms
  - Standards
- Framework for analysis
- Tool for individual process improvements
  - Developers find more errors
  - Developers improve their estimations
  - Developers improve productivity

➨ Improvements at “no” costs

PSP Overview



Exempel på ett script

Fas	Syfte	Handledning för PSP0 planering
	Inträdeskrav	Problembeskrivning Projektplan sammanfattningsformuläret Tidsloggning
1	Definiera programkrav	Framställa eller skaffa ett kravdokument för programmet Säkerställa att kravdokumentet är klar och motsägelsefri Lösa alla frågor angående kravdokumentet
2	Uppskatta resurser	Göra den bästa möjliga uppskattningen av tiden som krävs för att utveckla programmet
	Utträdeskrav	Dokumenterade krav Fullständig projektplan sammanfattning med tidsuppskattning för utvecklingstid Fullständiga tidsloggar

PVK--HT99 37

Tidslog

Namn: jubo
Datum: 990428
Program: Test_1

Datum	Tid Start	Tid Stop	Minuter Avbrott	Minuter Delta	Fas	Kommentar
990428	13:15	14:15	15	45	Planering	Kort fikarast
	14:15	15:15		60	Design	

PVK--HT99 38

Fellog

Namn: *jubo*
Datum: *990428*
Program: *Test_1*

Datum	Nummer	Feltyp	Införd	Borttagit	Fix-tid (min)	Fixat fel
990428	25	20	kodning	kompilering	1	

Beskrivning: Glömd krulleparantes.

Datum	Nummer	Feltyp	Införd	Borttagit	Fix-tid (min)	Fixat fel
990428	26	20	kompilering	kompilering	1	25

Beskrivning: Knappade in två krulleparantes vid felrättning.

PVK--HT99 39

Projektplan och sammanfattning

- Jämför planen med uppnådda resultat
  - Tid per fas
  - Kodstorlek
  - Kvalitet
    - Produkt
    - Process
  - Produktivitet
  - Effektivitet
- Statistik
- **Kolla exempel**

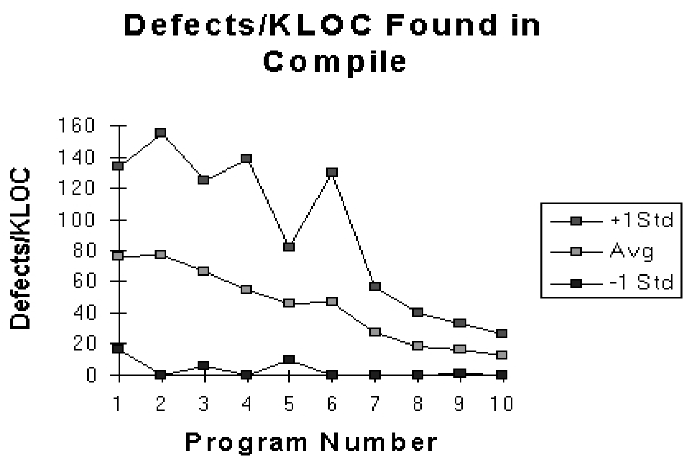
PVK--HT99 40

PSP Results

- Many published case studies
  - SEI
  - DoD m m (fig. 7!)
  - Embry-Riddle Aeronautical University
  - Hawaii University
  - Lund universitet (Claes Wohlin)
  - OOPS! Often very few students (not Hawaii/Lund)
- Industry relevant (mainly USA and Japan)
- Results are generally positive
  - Improved predictions
    - Development time
    - Program size
    - Quality of results
  - Improved quality
  - Improves development processes

PVK--HT99 41

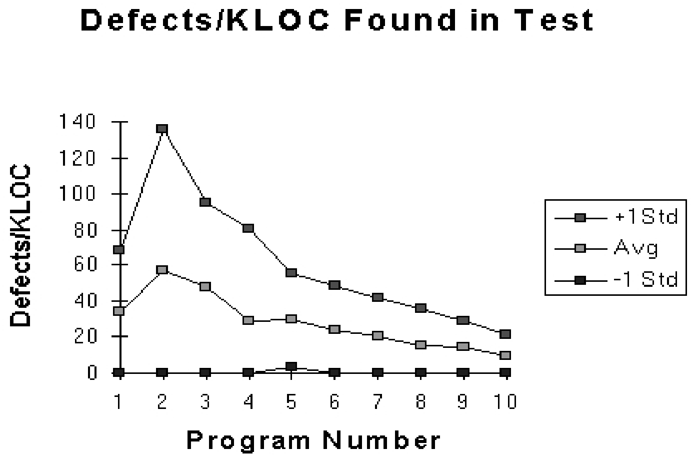
Defect Detection in Compile



See SEI´s web pages at http://www.sei.cmu.edu/psp/Results.htm

PVK--HT99 42

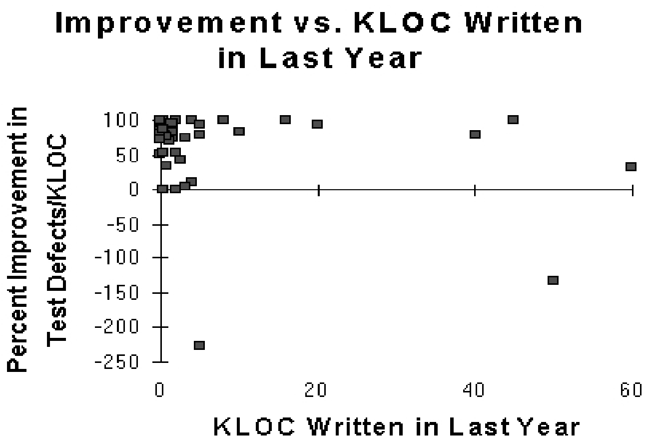
Defect Detection in Test



See SEI´s web pages at *http://www.sei.cmu.edu/psp/Results.htm*

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 43

Improvements in Defect Detection



See SEI´s web pages at *http://www.sei.cmu.edu/psp/Results.htm*

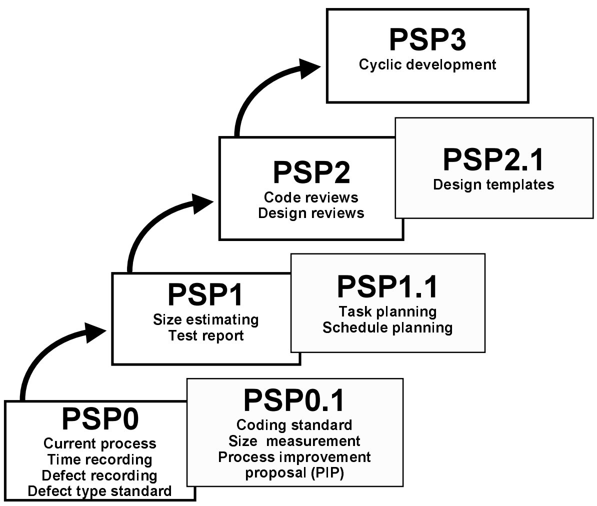
PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 44

Problem med PSP

- Stel process
  - Måste allt vara klart innan första compile?
  - Måsta alla fel loggas?
  - Hur hanteras iterativ utveckling/prototyping?
- Utvecklare slutar använda PSP
  - Uppföljning?
  - Koppling till CMM?
- Datainsamling problematisk
  - Manuell datainsamling är opålitlig (Hawaii studie)
  - Automatisk datainsamling innebär etiska risker
- Inga verktyg som stödjer PSP fullt ut

PVK--HT99 45

PSP Levels



PVK--HT99 46

PSP0 Documents

- PSP0 Process Script
- PSP0 Planning Script
- PSP0 Development Script
- PSP0 Post-mortem Script
- PSP0 Project Plan Summary and Instructions
- Time Recording Log and Instructions
- Defect Recording Log and Instructions
- Defect Type Standard

PSP3 Documents

- PSP3 Process Script
- PSP3 Planning Script
- PSP3 High-level Design Script
- PSP3 High-level Design Review Script
- PSP3 Development Script
- PSP3 Post-mortem Script
- PSP3 Project Plan Summary and Instructions
- Operational Scenario T&I[1]
- Functional Specification T&I
- State Specification T&I
- Logic Specification T&I
- PSP3 Design Review Checklist
- Code Review Checklist
- Task Planning T&I
- Schedule Planning T&I
- PROBE[2] Estimating Script
- Test Report T&I
- Size Estimating T&I
- PIP[3] and Instructions
- Coding Standard
- Time Recording Log & I
- Defect Recording Log & I
- Defect Type Standard

[1] T&I: Template and Instructions
[2] PROBE: Proxy-based Estimation
[3] PIP: Process Improvement Proposal

CMM and PSP

5: <u>Optimized:</u>
- *Process change management*
- *Technology innovation*
- *Defect prevention*

4: <u>Managed:</u>
- *Quality management*
- *Process measurement and analysis*

3: <u>Defined:</u>
- *Peer reviews*
- Intergroup coordination
- *Software product engineering*
- *Integrated software management*
- Training program
- *Organization process definition*
- *Organization process focus*

2: <u>Repeatable:</u>
- Software configuration management
- Software quality assurance
- Software subcontract management
- *Software project tracking and oversight*
- *Software project planning*
- Requirements management

1: <u>Initial</u>

PSP key process areas

Contents

➩Requirements Engineering

- ➩RE Activities
- ➩Requirements Documentation
- ➩Properties of Requirements
- ➩RE Methods and Notations
- ➩Examples

RE Activities

- ◆ Acquire and identify requirements
  - ❑ Study the system / organisation
  - ❑ Study available documents
  - ❑ Ask users / domain experts
    - ❍ Questionnaires
    - ❍ Interviews
- ◆ Analyse and evaluate requirements
  - ❑ Domain analysis
  - ❑ Prototyping
  - ❑ JAD / JAW
  - ❑ Scenario modelling
- ◆ Document requirements
- ◆ Review and validate requirements

Purpose of the Requirements Document

- Describe external system behavior
  - Functional requirements ← guide analysis
  - User interface
  - Acceptable responses to undesired events
- Describe constraints
  - Non-functional requirements ← guide design
  - Acceptance criteria

➨ Implementation independent reference

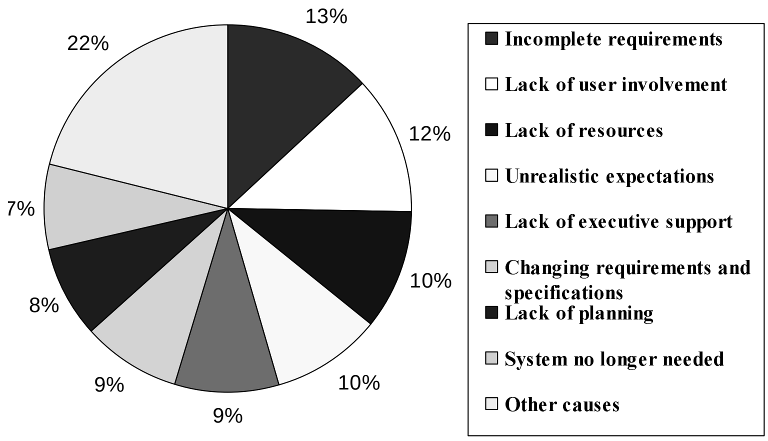
➨ Specifies the WHAT and not the HOW

➨ Part of the contract between customer and developer

Non-Functional Requirements

- Process requirements
  - Delivery
  - Implementation
  - Standards
- External requirements
  - Legislative
  - Costs
  - Inter-operability
- Product requirements
  - Usability
  - Efficiency
    - Performance
    - Space
  - Reliability
  - Portability

Why Do Projects Fail?



Study by the Standish Group involving 350 companies from 1994/95, see [Pfleeger 98].

PVK--HT99 55

Documenting Requirements is Important

"Projects without clear goals will not achieve their goals clearly."

Gilb's principle of fuzzy targets

- Clear Goals are
  - Understandable by users and developers
  - Correct
  - Consistent
  - Complete
  - Realistic
  - Testable
  - Traceable
  - Prioritised

?

➨ Quality factors for requirements documents

PVK--HT99 56

Requirements Writing Style

- Do not use vague terms or verbs like “some,” “obviously,” “usually,” “often,” “it follows that,” …
- Make sure that uncompleted lists are understood completely (e.g. “etc.,” “and so on,” “…,” ...)
- Make sure that ranges are clearly understood, e.g. what means “in the range of 1 to 100”
- Ask for clear definitions of terms like “always,” “never,” “almost,” etc.
- Use pictures and examples to aid in understanding
- Explain all of your terminology
- Use “shall,” “must,” “should,” constistently

PVK--HT99 57

User Requirements and Software Requirements Documents

Service Information	**Service Information**
a Abstract	a Abstract
b TOC	b TOC
c Document status and history	c Document status and history
1 Introduction	**1 Introduction**
1.1 Purpose	1.1 Purpose
1.2 Scope	1.2 Scope
1.3 Glossary	1.3 Glossary
1.4 References	1.4 References
1.5 Overview	1.5 Overview
2 General Description	**2 General Description**
2.1 Product perspective	2.1 Relation to other projects
2.2 General capabilities	2.2 Function and purpose
2.3 General constraints	2.3 Environmental considerations
2.4 User characteristics	2.4 Relation to other systems
2.5 Operational environment	2.5 General constraints
2.6 Assumptions and dependencies	2.6 Model description
3 Specific requirements	**3 Specific requirements**
3.1 Capability requirements	3.1 Functional requirements
3.2 Constraint requirements	3.2 Non-functional requirements
	4 Requirements traceability matrix *Relates URD and SRD*

Slightly adapted from ESA's Software Engineering Standards PSS-05-0 (see [ESA 96])

PVK--HT99 58

⇨Requirements Engineering

- ⇨ RE Activities ✔
- ⇨ Requirements Documentation ✔
- ⇨ Properties of Requirements ✔
- ⇨ **RE Methods and Notations**
- ⇨ **Examples**

Classical Approaches to RE

- Problems:
  - Coping with size
    - ➜Structured approach
    - ➜Stepwise refinement
    - ➜Hierarchical organisation
  - Coping with change
    - ➜Logic model
    - ➜Maintainable results
  - Coping with documentation
    - ➜Simple notation
    - ➜Graphical elements
- Solutions:
  - **SA** (75/76)
    - ➜Function-oriented
  - **ER** (end 70s)
    - ➜Data-oriented
  - **SA/RT** (85/87)
    - ➜Control-oriented
  - Integrated approaches
    - ➜SA/ER/RT (end 80s)
    - ➜OO/RT (early/mid 90s)
    - ➜???

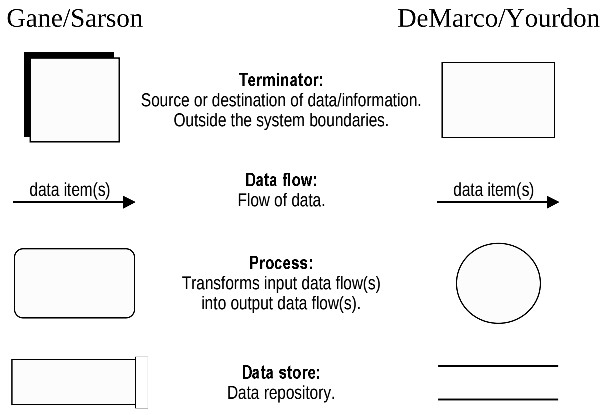
➨ Systematic approaches to requirements analysis and definition

Structured Analysis (SA)

- Developed 1975/76
  - DeMarco/Yourdon
  - Gane/Sarson
- System = Process transforming input into output
- Hierarchical, logical system model
  - Processes
  - Data flows
  - Data stores
  - Terminators
- Notation:
  - Data flow diagrams (DFDs)
  - Data dictionary (DD)
  - Process specifications (PSpecs)

PVK--HT99 61

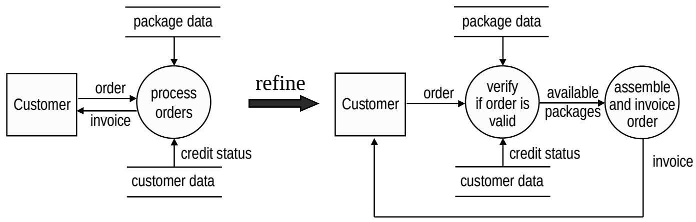
Data Flow Diagrams



PVK--HT99 62

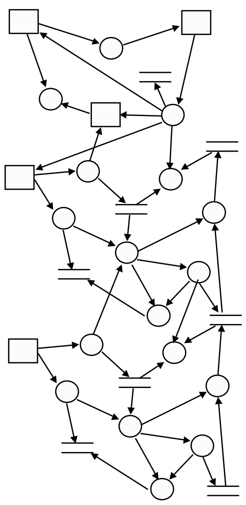
DFD Development

- Start with a *context diagram*
- Successively refine processes
- Describe all data in the data dictionary
- Describe all atomic processes by PSpecs
- Example: Order processing

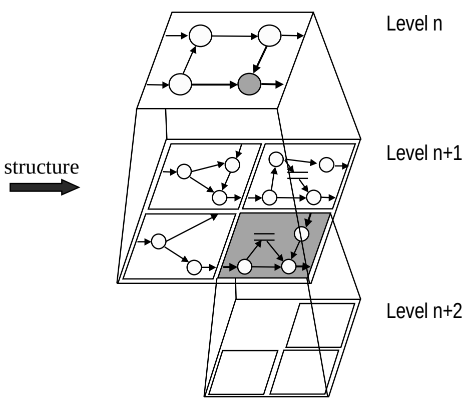


PVK--HT99 63

DFDs--Managing Complexity



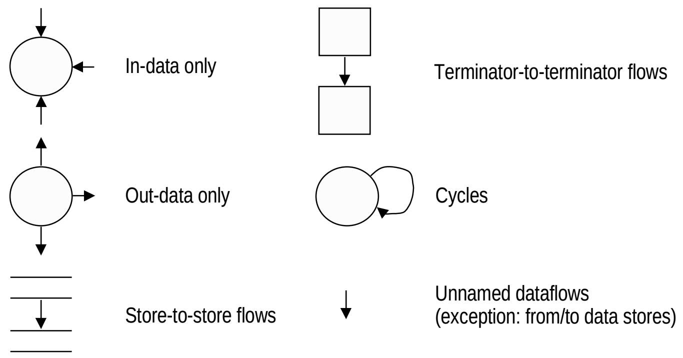
DFD hierarchy
+ numbering/naming rules
+ balancing rules



PVK--HT99 64

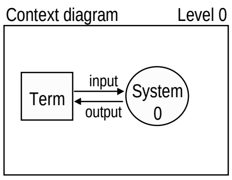
DFDs--“Forbidden” Structures

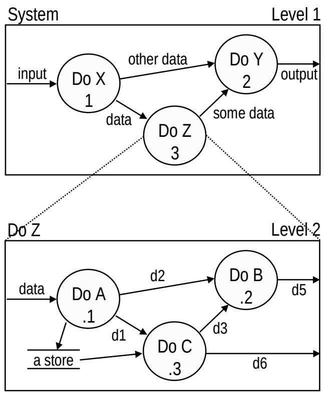
- The SA notation is not formally defined
- Rules are defined by experiences and tool features



PVK--HT99 65

DFD Naming and Balancing





All names must be unambigous.

Numbering scheme helps to find processes in the hierarchy
➜ Do C: 3.3

In data dictionary: some data = d5 + d6
(alternatively: some data = d5 | d6)

PVK--HT99 66

PSpecs and DD

- The format of PSpecs is not restricted
  - Free text
  - Pseudocode
- PSpecs must be defined for all atomic processes
- The format of the DD is semi-formal
- Example:

telephone number = [local extension | outside number] ← selection (or)
local extension = 2 + { number }^{3}
outside number = 0 + [local number | long distance number] ← composition (and)
local number = prefix + access number
long distance number = (1) + area code + local number ← optional
prefix = [123 | 124 | 125]
access number = { number }^{4} ← repetition
number = * any number between 0 and 9 * ← a comment

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 67

SA--Summary

- Advantages
  - Simple notation
  - Supports hierarchical decomposition
  - Easy to understand
  - Good communication medium
  - Supports consistency checks
- Disadvantages
  - Not well defined
  - No common guidelines
  - Many dialects
  - Incomplete
    - Very poor data descriptions
    - No description of control flows

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 68

SA/RT

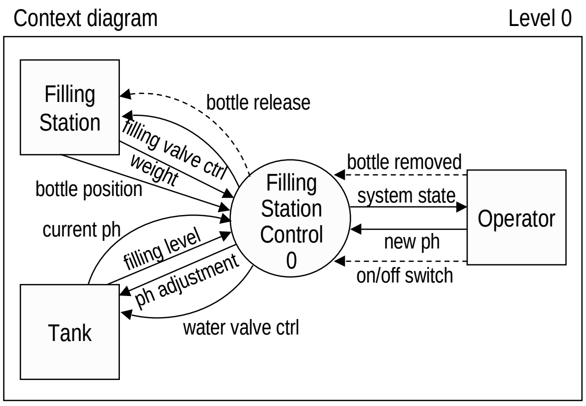
- Extension of SA to describe control flow
  - Activation/deactivation of processes
  - Modelling of events (signals)
  - States and state transitions
- Ward/Mellor (1985), Hatley/Pirbhai (1987)
- Additional notation (by Hatley/Pirbhai)
  - Control flow diagrams (CFDs) ← Extended DFDs
  - Process activation tables (PATs)
  - State-transition diagrams (STDs)

Idea: Each DFD contains one central control process that consumes and produces all control flows.

PVK--HT99 69

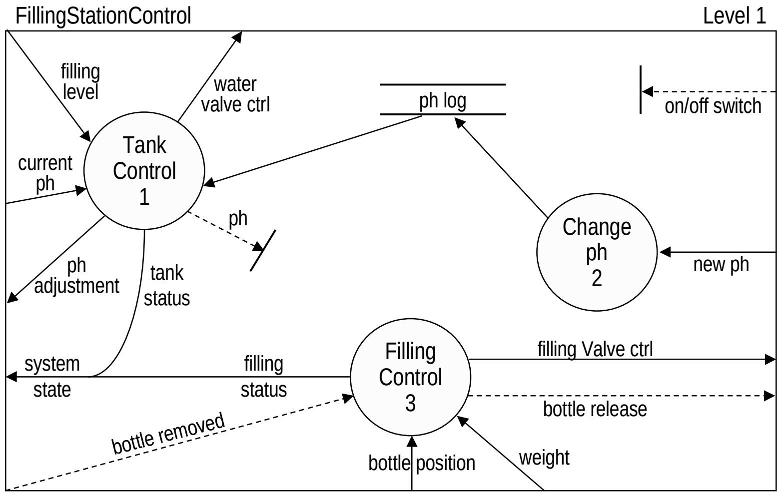
SA/RT--An Example

- Bottle filling station



PVK--HT99 70

Refinement of FillingStationControl



PVK--HT99 71

Process Activation Table (PAT)

- Shows how processes are activated/deactivated depending on the control flows
- To model complex dependencies that depend on internal system states STDs are needed

FillingStationControl

Control input		Process activation		
on/off switch	ph	Tank Control 1	Change ph 2	Filling Control 3
on	OK	1	0	1
on	out of bounds	1	1	1
off	--	0	0	0

PVK--HT99 72

SA/RT--Summary

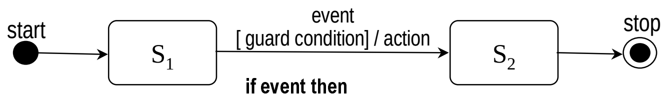
- Advantages
  - Straight forward extension of SA
  - Supports hierarchical decomposition
  - Broad applicability
  - Quite well defined (STDs)
  - Tool support
- Disadvantages
  - Very poor data descriptions

➨ Found its way to OO approaches

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 73

State Transition Diagram (STD)

- Most often used independently of SA/RT

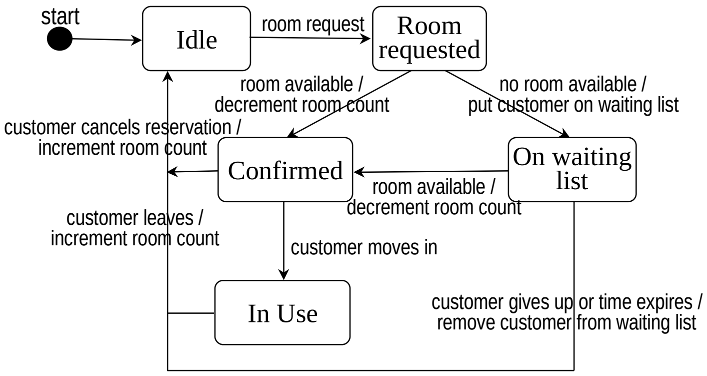



```
if event then
  if condition then
    trigger transition;
    execute action
  else
    nothing happens;
```

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 74

STD Example

- Room reservation system



PVK--HT99 | | 75

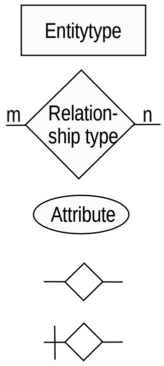
Data Modelling

- The entity-relationship (ER) model was developed by Chen (late 70s) to support data(base) modeling
- Focuses only on the static structure of data
- Notation
  - Entity-relationship diagrams (ERDs)
  - Attribute dictionary

PVK--HT99 | | 76

ERD Notation

◆ According to Chen + common extensions



Set of real or abstract things about which data is stored

Set relations between entities with cardinalities m and n.

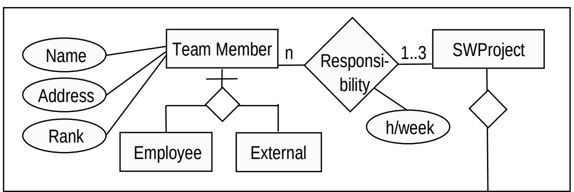
Information that is stored along with entities and relationships.

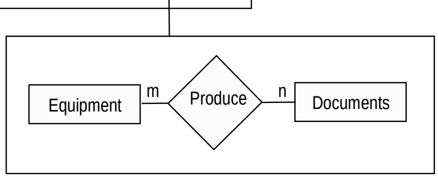
Composition of entities.

Classification between entity- and relationship types.

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 77

ERD--An Example





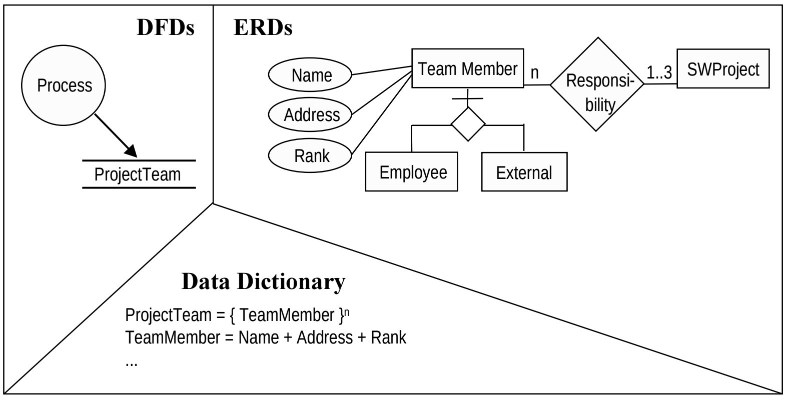
Attribute Dictionary

Attribute structures
TeamMember = Name, Address, Rank;
Employee = ...;
...

Attribute types
Name = STRING;
Address = STRING;
...

PVK--HT99 Copyright © 1997-1999, jubo@cs.umu.se 78

SA/ER Integration



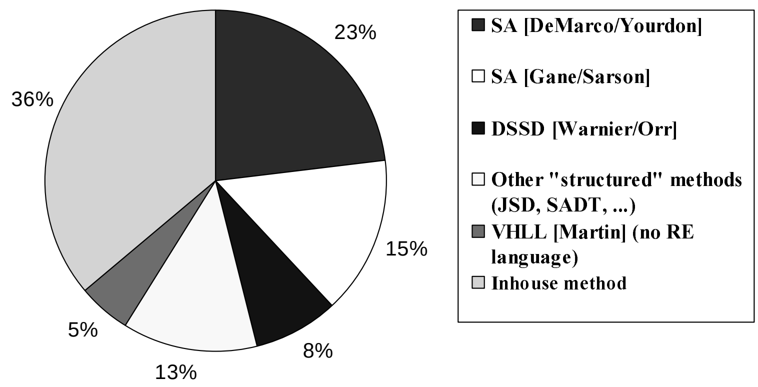
PVK--HT99 79

ERM--Summary

- Advantages
  - Simple notation
  - Supports hierarchical and structural decomposition
  - Easy to understand
  - Good communication medium
  - Well understood
  - Widely used
  - Good tool support
- ➨ Well-suited for DB design
- Disadvantages
  - No behaviour descriptions
  - No control descriptions
- ➨ Almost useless for non-DB applications
- ➨ Extensions of ERM lead to OO approaches

PVK--HT99 80

Popularity of (Classical) Methods



Study from 1990, see [Yourdon 92].

Object Oriented Analysis (OOA)

Optional topic, see lectures.

Contents

PVK--HT99 83

➩UI Design

- ➩ UI Design Activities
- ➩ General Guidelines
- ➩ Toolkits

PVK--HT99 84

UI Design Activities

- Analyse the users
  - Are they used to computers
  - Have they used similar systems
  - Are there different user groups
  - Are they trained before using the system
  - Have they certain cultural or ethnic backgrounds
  - ..
- Build prototype(s)
- Evaluate prototype(s)
- Choose an UI style
  - Command-line interface
  - Natural language interface
  - Menu-oriented interface
  - Question-answer dialogues
  - Form fill-in menus
  - Direct manipulation interface
  - WIMP interface
    - Windows
    - Icons
    - Menus
    - Pointing devices
  - Some combination

UI Design Issues

- Response time
- Error handling
  - Responses to undesired inputs
  - Error messages
- Help facilities
- Customisation facilities
- Common look and feel
- Guidelines
- Standards
- Human memory load
  - # Commands
  - # Arguments
  - # Windows
  - # System states
  - ...
- Colour coding
- Icon selection
- ...

General UI Design Guidelines

- Be consistent
- Offer meaningful feedback
- Ask for verification of any non-trivial destructive action
- Permit easy reversal of most actions
- Reduce memory load
- Seek efficiency (minimise user input, mouse motions, astonishment, ...)
- Forgive mistakes
- Categorise and organise activities
- Provide context-sensitive help
- Use simple action verbs or short verb phrases to name commands

Guidelines for Displaying Information

- Display only the currently relevant information
- Use appropriate presentation forms (for example graphs instead of clumsy tables)
- Use consistent labels, standard abbreviations and predictable colours
- Maintain visual context
- Produce meaningful error messages
- Format text to aid in understanding
- Compartmentalise different types of information
- Use “analogue” displays
- Use screen geography effectively

Guidelines for Data Input

- Minimise the number of input actions
- Maintain consistency between information display and data input
- Interaction should be flexible (keyboard, mouse)
- Allow the user to customise input
- Deactivate inappropriate commands
- Let the user control the order of activities
- Provide help
- Eliminate “mickey mouse” input

Specific Guidelines

- OPEN LOOK™
  - Three level certification by AT&T
- Motif
- Microsoft
- Java
- ...

GUI Toolkits

- Libraries with predefined “widgets” and event handling
  - MVC paradigm (Smalltalk)
  - Interviews (C++)
  - Java AWT/ Swing
  - ...
- Visual tools (WYSIWYG)
  - Visual Basic
  - Delphi
  - XYZ Builder
  - ...