# Perspectives on Software Engineering cntd. Requirements Engineering

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
February 28, 2008

# Goals of this Lecture

To finish perspectives on software engineering

- Stages (1 half)
- Management Issues (1 half)
- Programmer Issues (2 half)

To discuss requirements engineering

To perform tutorial on perspectives on Software Engineering

- D403a, D601a

To discuss SCRUM

# What you should know after

Able to discuss some perspectives on programmer issues in SE
Able to reason on requirements engineering process and some techniques
Able to apply SCRUM

# Testing – Certification

No syntactic error
No compilation Error
There exist data for which program gives correct answers
For typical sets of data program gives correct answers
For difficult sets of data program gives correct answers
For all possible sets of data valid with respect to specification the program gives correct answers
For all possible sets of valid data and all likely conditions of errorneous input the program gives correct answers
For all possible input, the program gives correct answers

# Program Verification


FIGURE 7. Assertions $A_i$ and $A_j$ surround each statement of a program.


FIGURE 8. Predicates $A_1$ and $A_n$ specify input-output behavior of a program.

# Go-to statement considered harmful


DIJKSTRA, E. "GOTO statement considered harmful," *Commun. ACM* 11, 3 (March 1968), 147-148.

## Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing
*CR* Categories: 4.22, 5.23, 5.24

EDITOR:

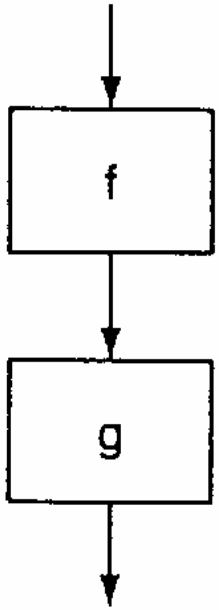
For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all "higher level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

My first remark is that, although the programmer's activity

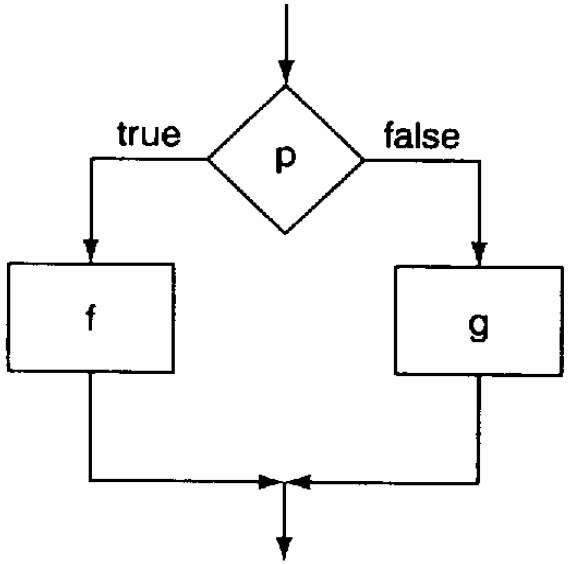
dynamic
call of t
we can
textual
dynamic
Let u
or repea
superflu
recursiv
clude th
mented
the othe
makes u
processe
the repe
describe
a repeti
namic i
corresp

# Structured Programming

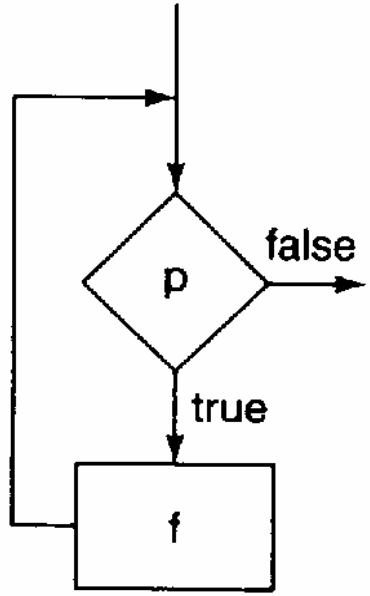
Sequence




Selection




Iteration




**Minimize use of “goto” statements**

**Use sequence, selection, and iteration building blocks**

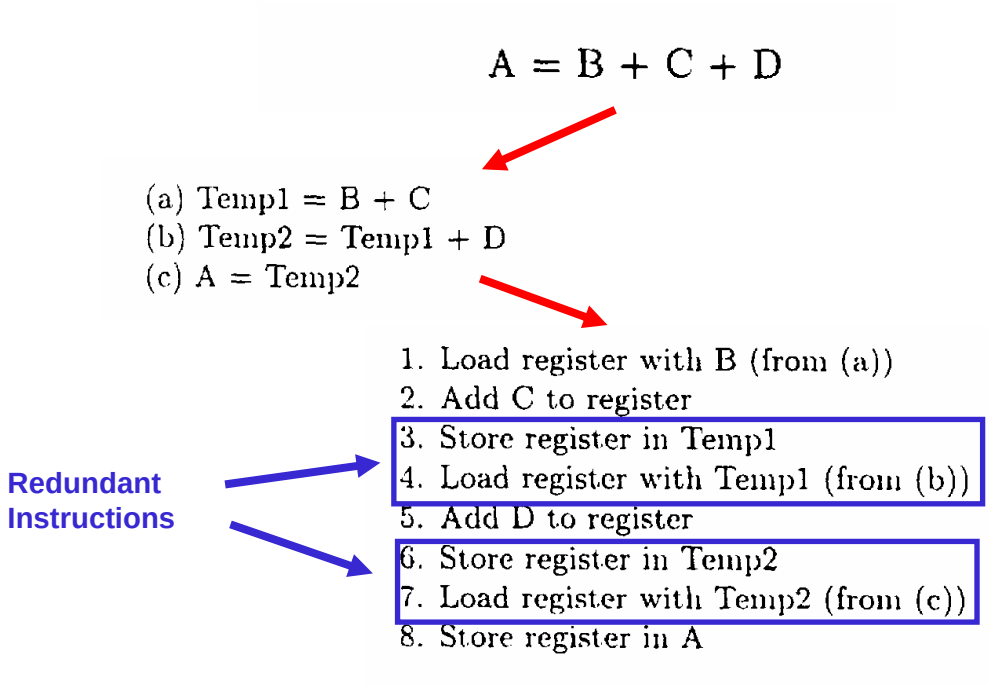
# Optimization I

A = B + C + D

(a) Temp1 = B + C
(b) Temp2 = Temp1 + D
(c) A = Temp2

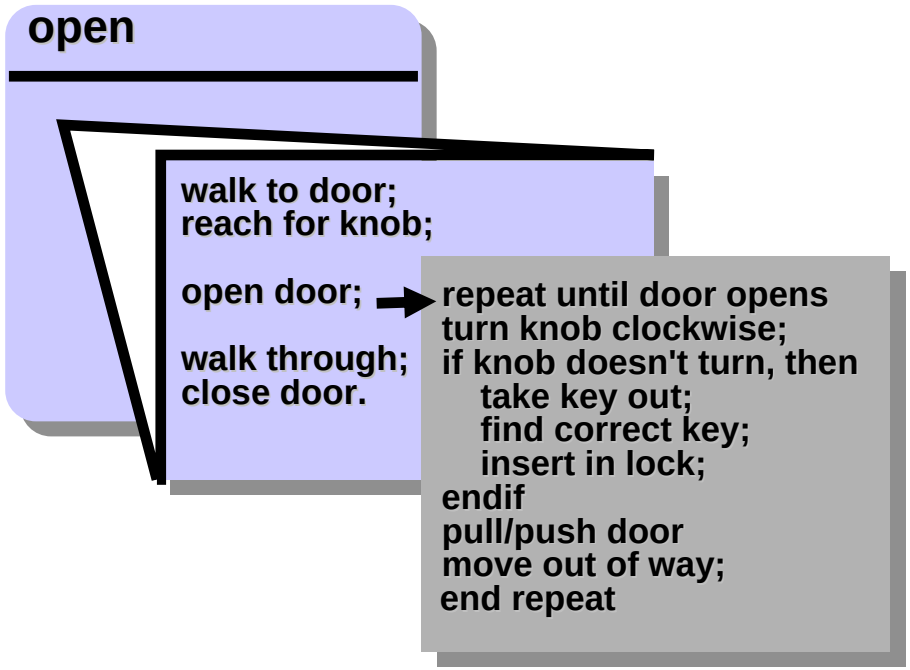
1. Load register with B (from (a))
2. Add C to register
3. Store register in Temp1
4. Load register with Temp1 (from (b))
5. Add D to register
6. Store register in Temp2
7. Load register with Temp2 (from (c))
8. Store register in A

# Optimization 2



# System Design

Top Down Development
Stubs
Iterative Enhancement
Throw away first version




Stepwise refinement

# Boehm's Seven Principles

Manage using a sequential life cycle plan
Perform continuous validation
Maintain disciplined product control
Use enhanced top-down structured programming
Maintain clear accountability
Use better and fewer people
Maintain commitment to improve process

# Goals of this Lecture

To finish perspectives on software engineering

- Stages (1 half)
- Management Issues (1 half)
- Programmer Issues (2 half)

To discuss requirements engineering

To perform tutorial on perspectives on Software Engineering

- D403a, D601a

To discuss SCRUM

# Requirements Engineering

”Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It **is** also concerned with the relationship of **these** factors to precise specifications of software behavior, and to their evolution over time and across software **families."**

# Software Requirements Processes

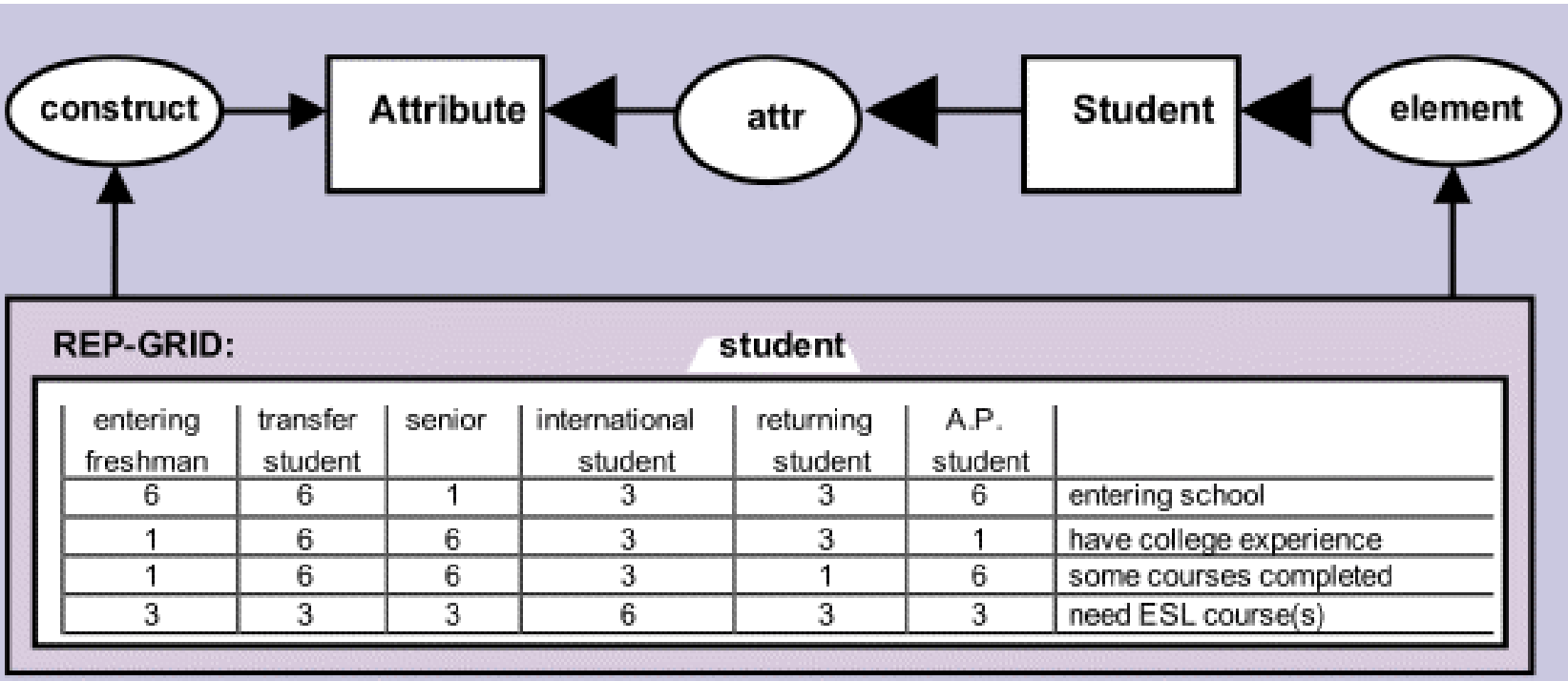
*eliciting* requirements,
*modelling* and *analysing* requirements,
*communicating* requirements,
*agreeing* requirements, and
*evolving* requirements.

# Context and GroundWork

Specific customer
Market driven
Different techniques for different type of software

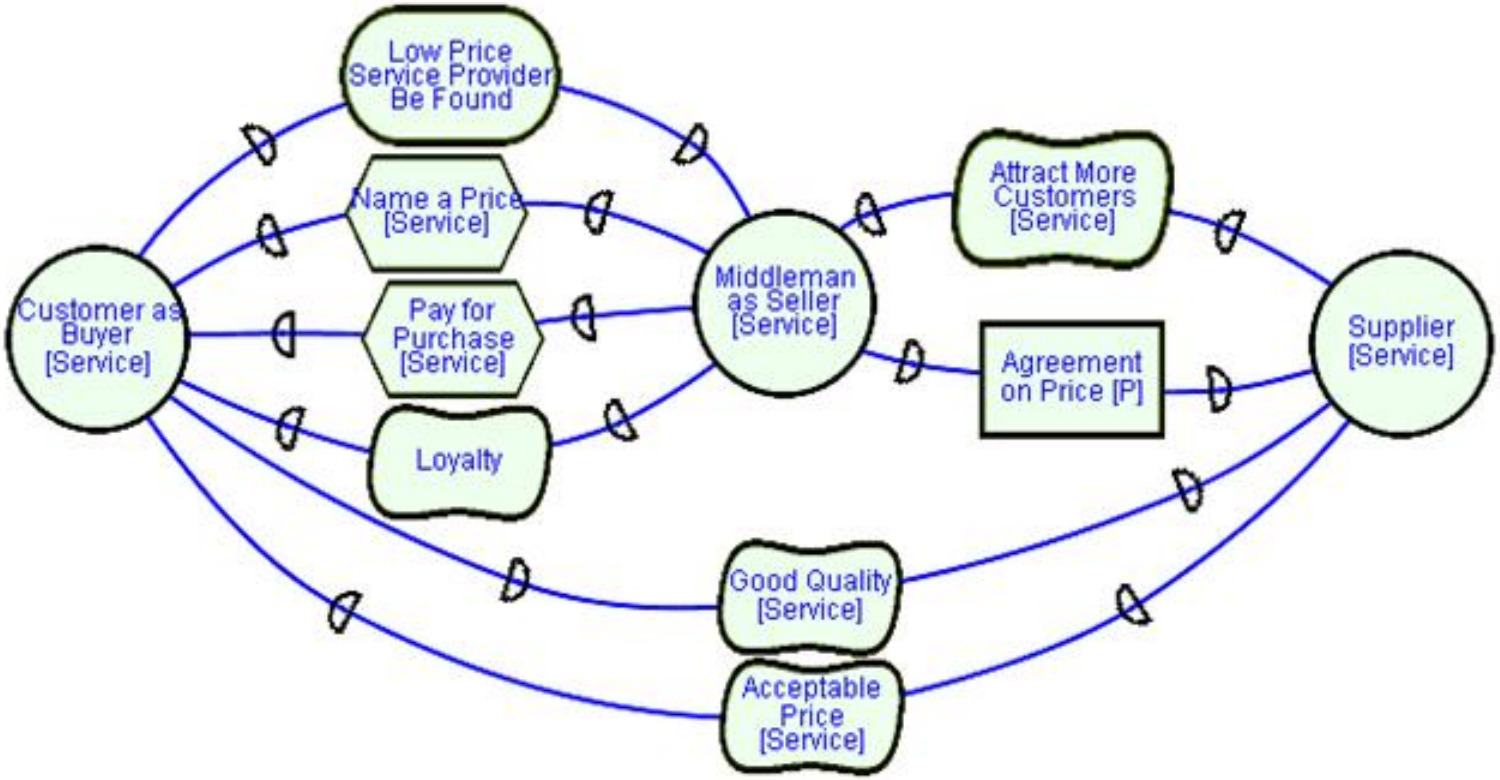
Preparation
Assessment on feasibility
High level conflict assessment
Identification of a suitable process for RE with suitable method and techniques

© boxesandarrows




| entering freshman | transfer student | senior | international student | returning student | A.P. student | |
|---|---|---|---|---|---|---|
| 6 | 6 | 1 | 3 | 3 | 6 | entering school |
| 1 | 6 | 6 | 3 | 3 | 1 | have college experience |
| 1 | 6 | 6 | 3 | 1 | 6 | some courses completed |
| 3 | 3 | 3 | 6 | 3 | 3 | need ESL course(s) |

**http://ksi.cpsc.ucalgary.ca/KAW/KAW98/delugach/**




**http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide**

# Requirements Elicitation

Elicitation and capturing
Boundaries
Stakeholders
Goals
Tasks
Use Cases

# Elicitation Techniques

Data access: questionaires, surveys, interviews, documentation, …

Group elicitation techniques: workshops, brainstormings

Prototyping

Model-driven techniques such as goal oriented or scenario oriented, OO techniques

Cognitive techniques: protocol analysis, laddering, card sorting, repository grids

Contextual techniques: participant observation

# Communicating requirements

To communicate to different stakeholders
Specification languages
Requirements management: traceble and readable by the others
Standards to document and structure
Traceability: read, navigate, query and change

# Agreeing the Requirements

Validation
Inspection and formal analysis: coherence
Prototyping and specification animation and scenarios:
correspondence to real world problems
What we know?
Social agreement?

# Evolving Requirements

Version control and configuration amangement
Impact of req. Change on other products of SEProcess
Consistency
Viewpoints framework

# Goals of this Lecture

To finish perspectives on software engineering

- Stages (1 half)
- Management Issues (1 half)
- Programmer Issues (2 half)

To discuss requirements engineering

To perform tutorial on perspectives on Software Engineering

- D403a, D601a

To discuss SCRUM