

# Software Process Improvement: What gets measured gets done

Petter Øgland, Department of  
Informatics, University of Oslo

*IRIS 31, Aug 13. 2008*

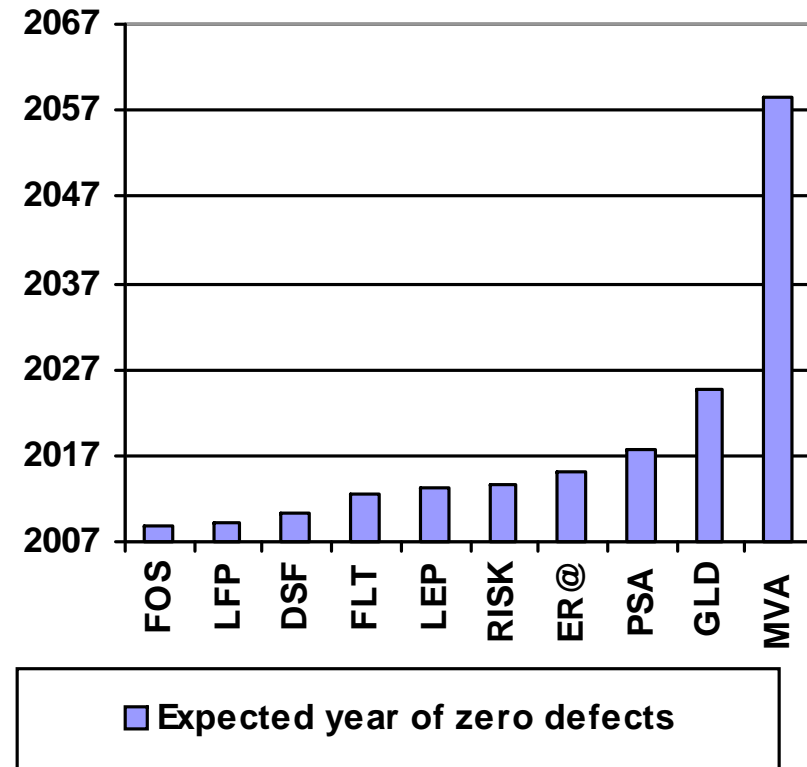
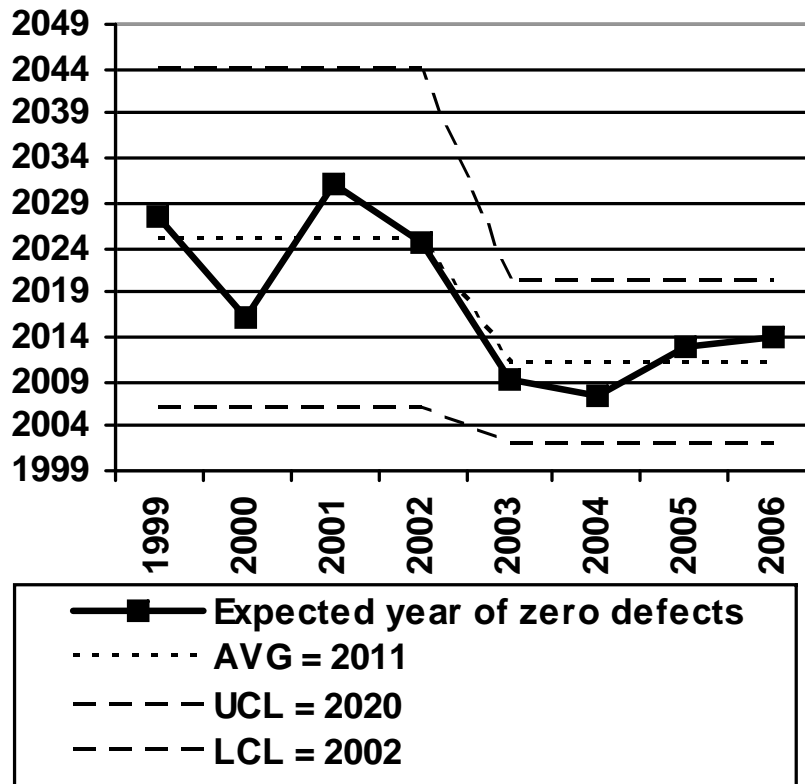
# Case background (motivation for QMS design)

- 1997: Tragic death of programmer, near fatal business disaster as nobody understands the COBOL software
- 1998: General agreement that there is need for standardized software to prevent similar events in the future

# QMS design

- Participatory Design (CPSR, 2008)
- Double-loop learning (Argyris & Schön, 1978)
- Provide and distribute numerical feedback (Sommerville, 2001)

# QMS evaluation: Predicting “Year of zero defects”



# Why is the 10th project (“MVA”) behaving differently?

- Should we have anticipated the odd behaviour?
- Is there a weakness in current QMS design theory?
- Can we improve theory and redesign QMS to provide better results?

# QMS design theory (Sommerville, 2001)

*To avoid problems with software engineers not following standards, we should:*

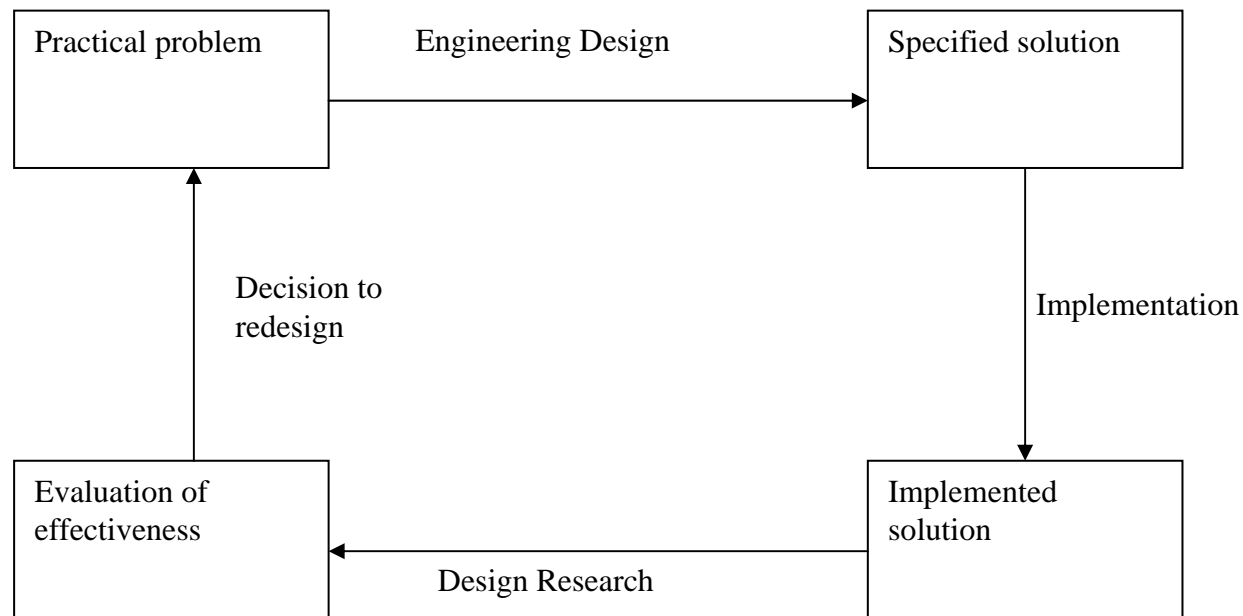
1. Involve engineers in design of standards
2. Review and modify standards
3. Provide software tools to support standards

# QMS design theory (measurements)

- Mason Haire: "What gets measured gets done" (Peters & Waterman, 1982)
- Senior managers have strongest belief in measurements, BUT software engineers accept measurements (Hall & Fenton, 1997)
- Although measurements shape behaviour, they may shape behaviour in both desirable and undesirable ways (Behn, 2003)

# Method: Design Research (March & Smith, 1995; Simon, 1996)

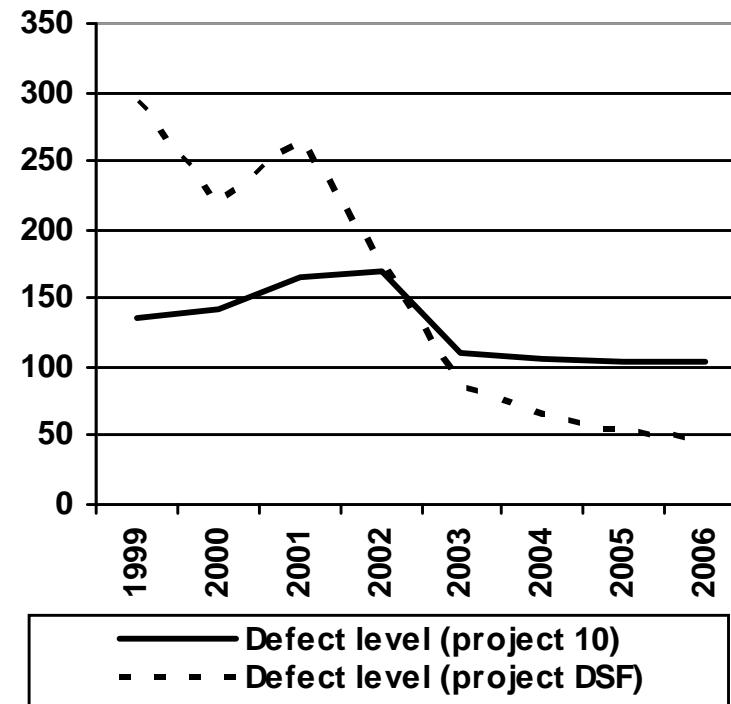
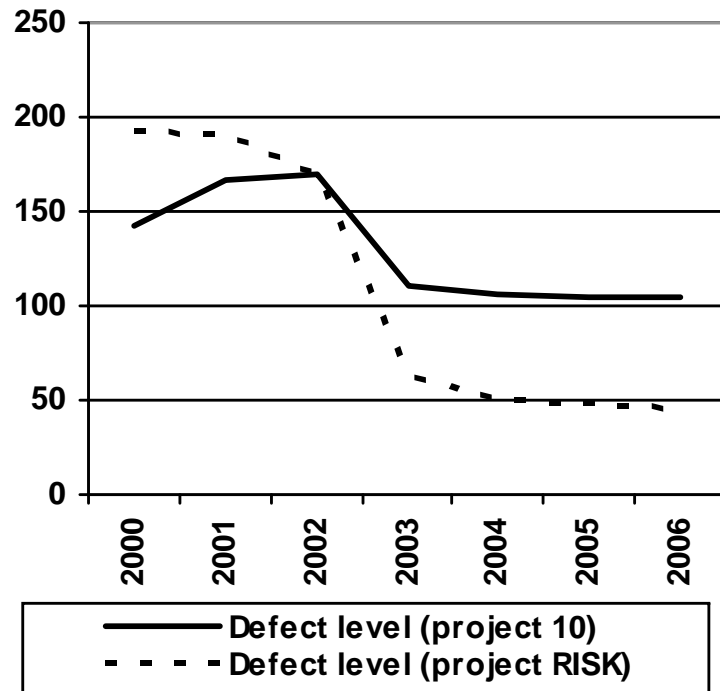
- Question = how to design QMS?
- Answer in terms of design algorithm



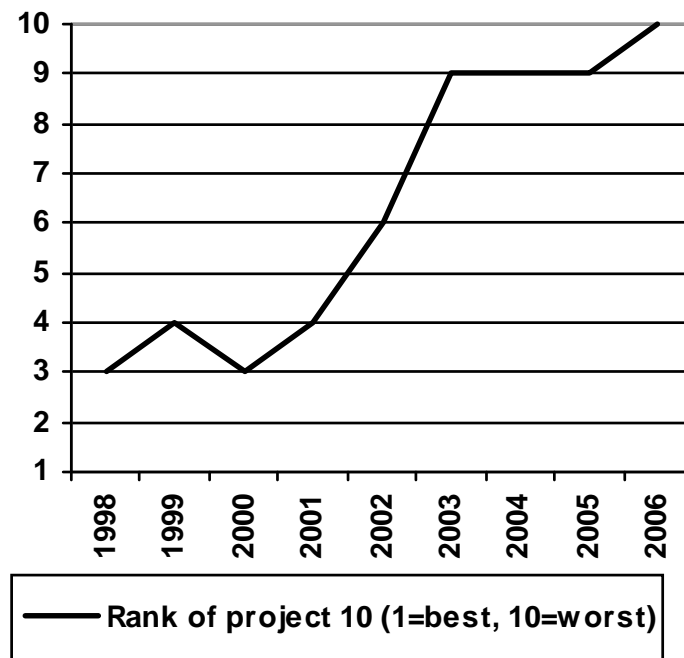
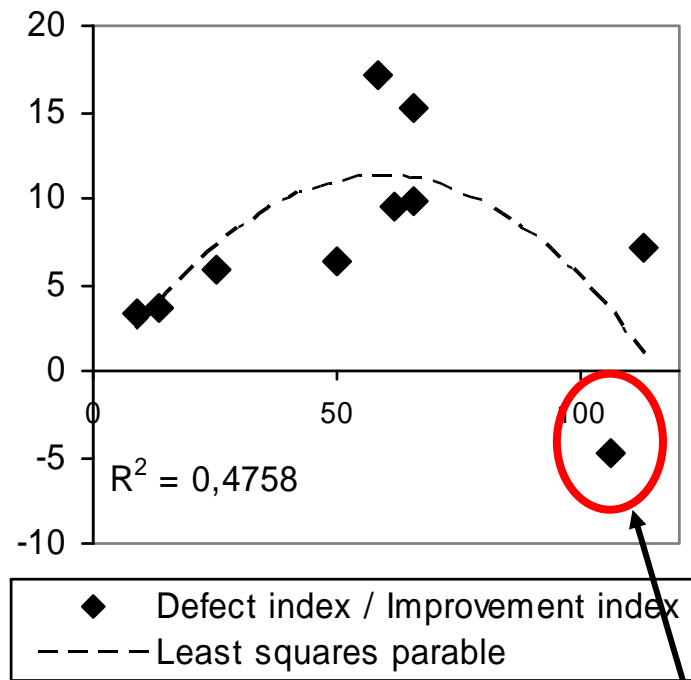
# Results from qualitative research (management explanations)

- The MVA software life-cycle model is different
- Being measured and widely published as "worst-in-class", the numerical results (benchmarking) for MVA are demotivating
- Due to high level of job security in the public sector, the MVA programmers do as they please and are impossible to manage

# The role of the software-development model



# The Pygmalion effect



MVA: High level of defects,  
low level of improvement

# “Social construction” of management commitment

- 9 to 5 software engineering (unmotivated)
- Management power games (unmotivated)

=> “Fundamental premises” such as quality culture and management commitment (e.g. Offen & Jeffrey, 1997) are not present, but the QMS design appears to compensate in 90% of the cases

# What gets measured gets done? (Input for new design algorithm)

- WGMGD is a statement about groups: Self-managed teams similar to SCRUM (Schwaber, 2004)
- The impact on measurements on individuals? Difficult to predict.
- QMS design insights seems to be: *Change conceptual model in order to avoid isolated islands of individuals among the teams*