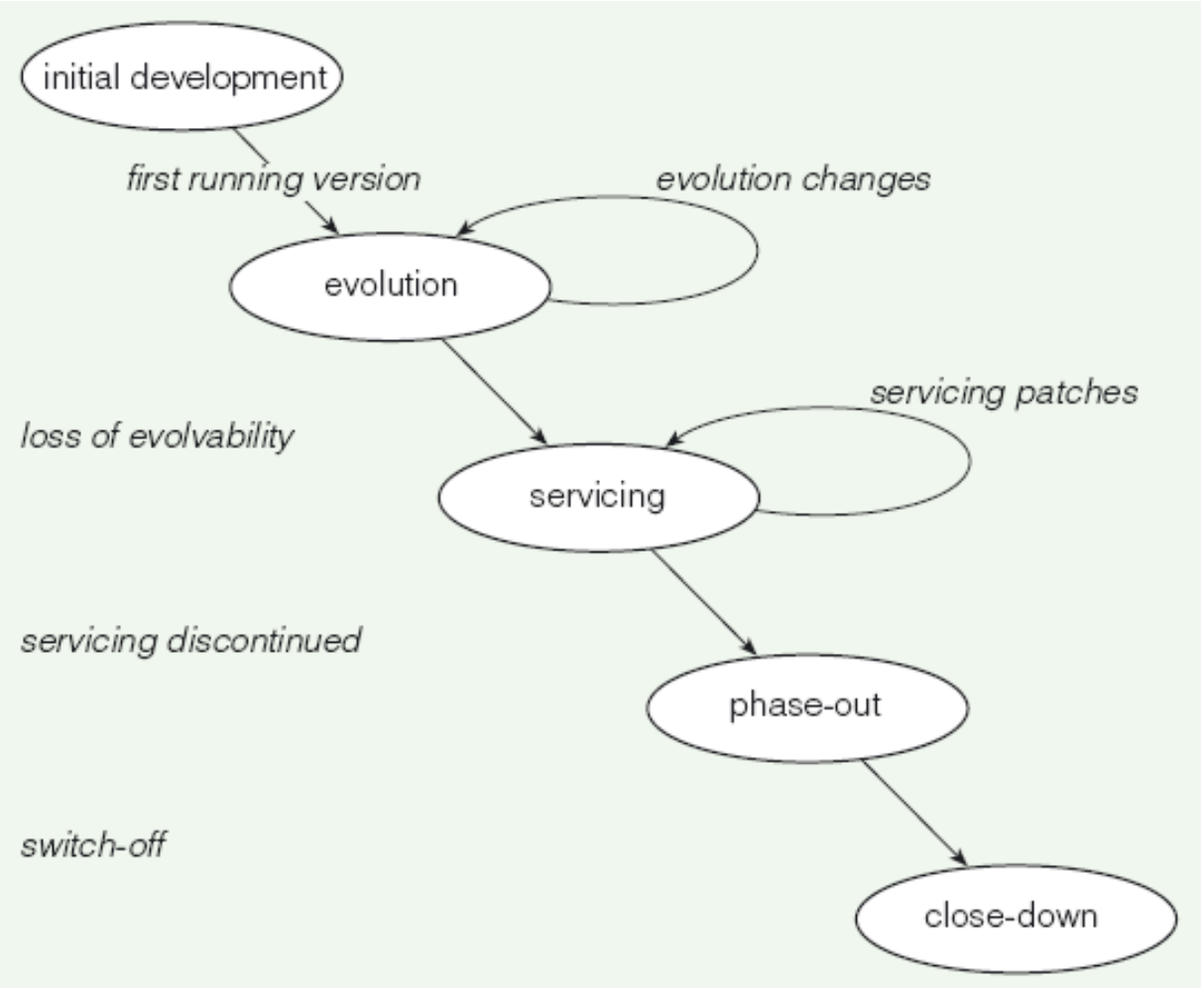
Quality & Evolution: some relationships

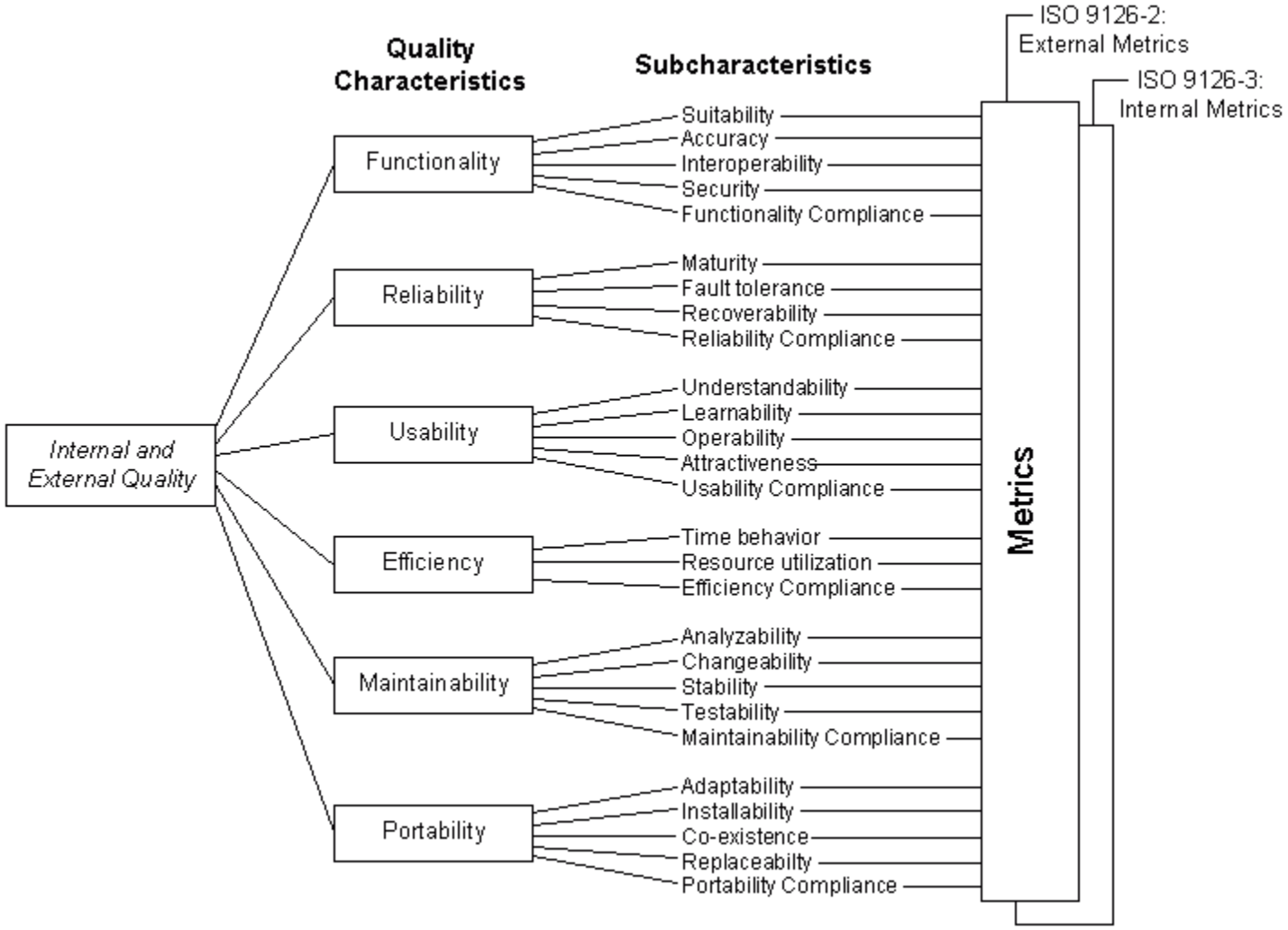
Michel Wermelinger
Computing Department
The Open University, UK
http://michel.wermelinger.ws

Product Evolution



Bennett & Rajlich. Software maintenance and evolution: a roadmap. ICSE 2000

Quality


Quality Characteristics
Subcharacteristics
ISO 9126-2: External Metrics
ISO 9126-3: Internal Metrics
Internal and External Quality
Functionality
Suitability
Accuracy
Interoperability
Security
Functionality Compliance
Reliability
Maturity
Fault tolerance
Recoverability
Reliability Compliance
Usability
Understandability
Learnability
Operability
Attractiveness
Usability Compliance
Efficiency
Time behavior
Resource utilization
Efficiency Compliance
Maintainability
Analyzability
Changeability
Stability
Testability
Maintainability Compliance
Portability
Adaptability
Installability
Co-existence
Replaceabilty
Portability Compliance
Metrics

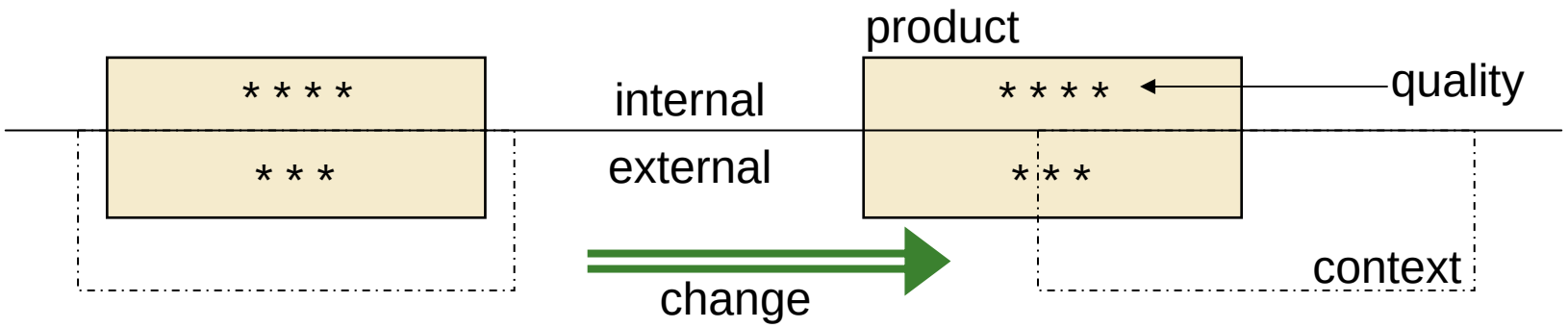
Principle of Software Uncertainty

- The real-world outcome of any E-type software execution is inherently uncertain with the precise area of uncertainty also not knowable

Madhavji, Ramil and Perry (eds.) Software Evolution and Feedback. Wiley 2006

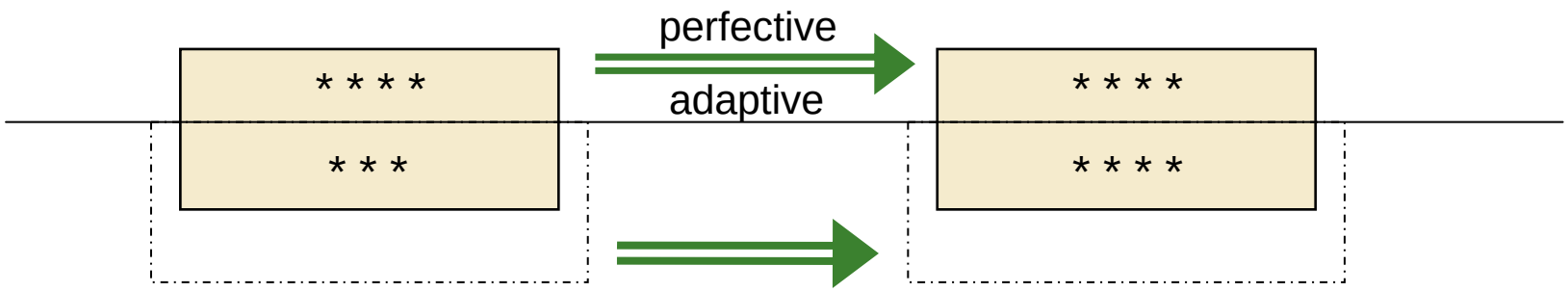
Declining Quality

- Unless rigorously adapted and evolved to take into account changes in the operational environment, the quality of an E-type system will appear to be declining
  - 7th law of software evolution



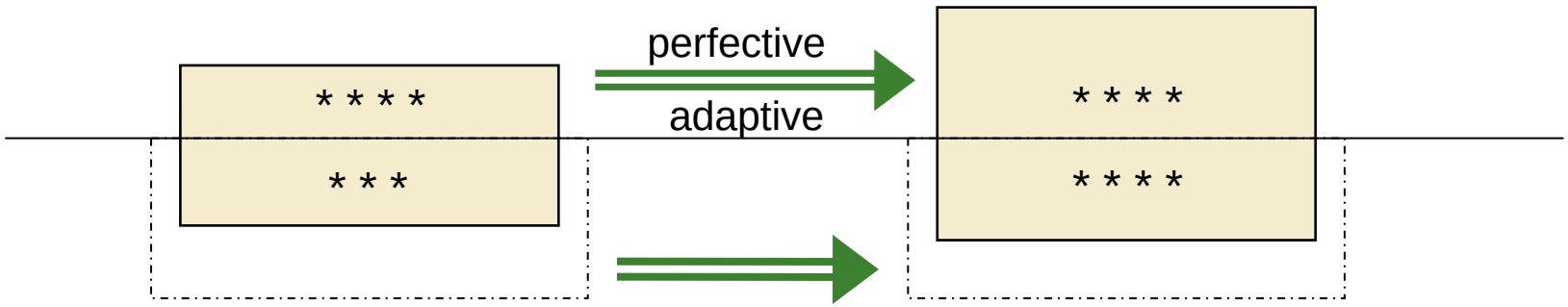
Continuing change

- An E-type system must be continually adapted or else it becomes progressively less satisfactory in use
  - 1st law of software evolution

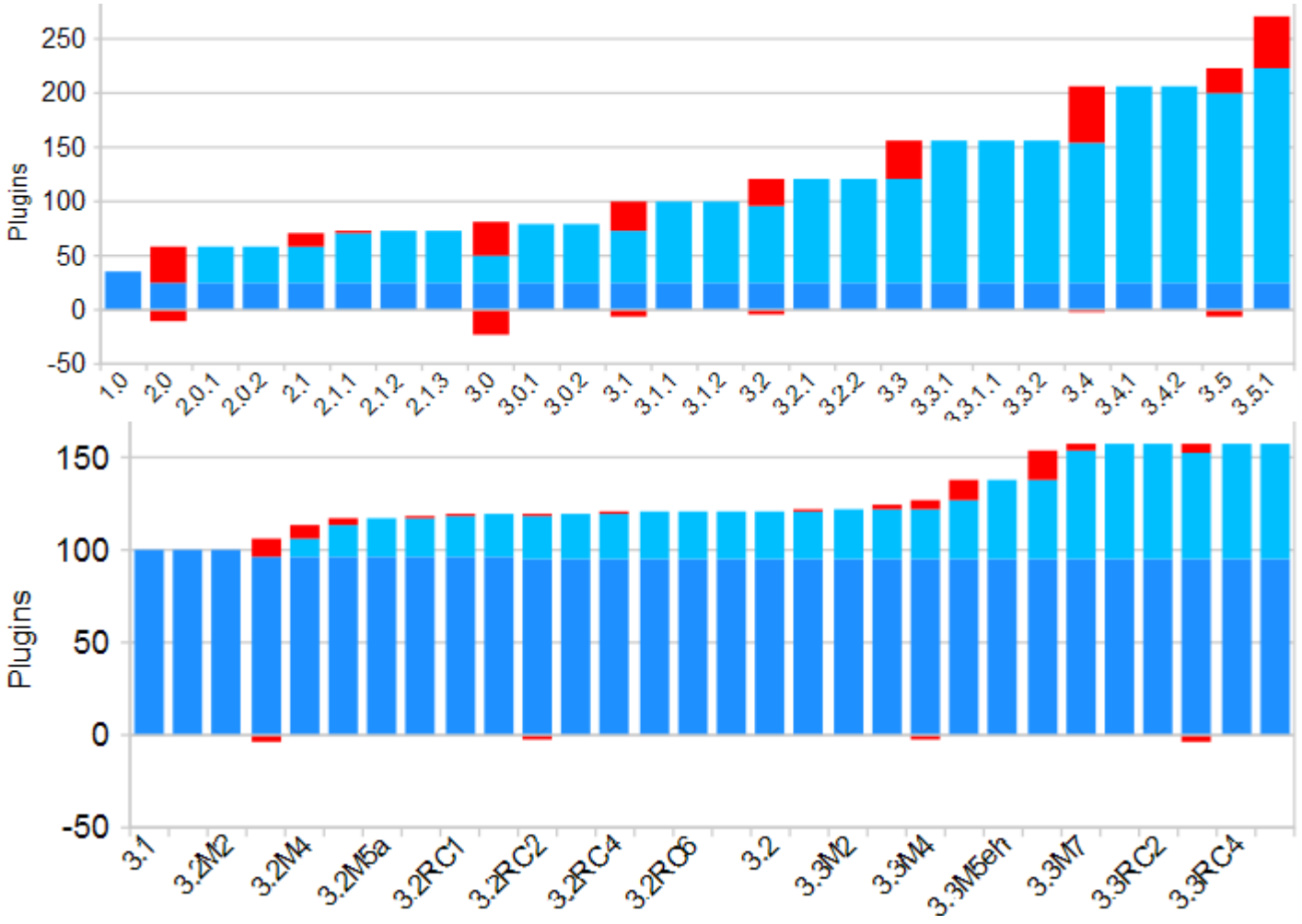


Continuing growth

- The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over the system lifetime
  - 6th law of software evolution



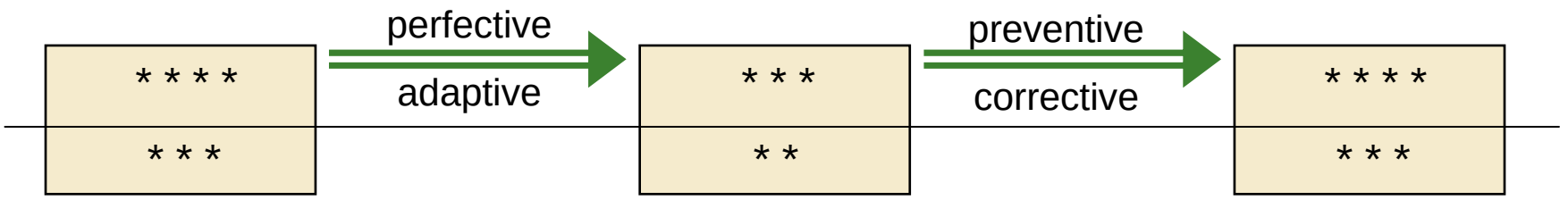
Punctuated Equilibrium



Wermelinger et al. Design principles in architectural evolution: a case study. ICSM 2008

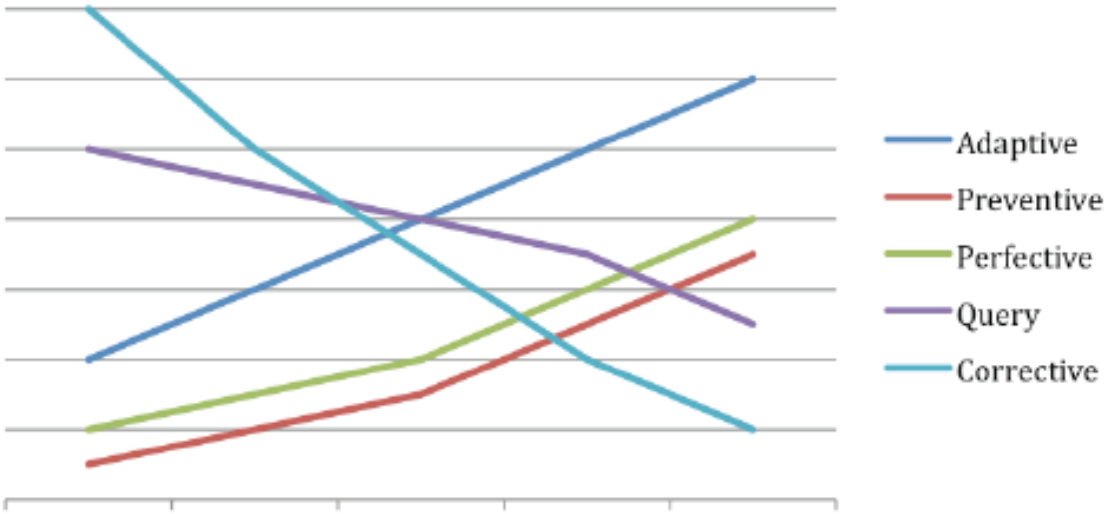
Increasing complexity

- As an E-type system is changed its complexity increases and becomes more difficult to evolve unless work is done to maintain or reduce the complexity
  - 2nd law of software evolution



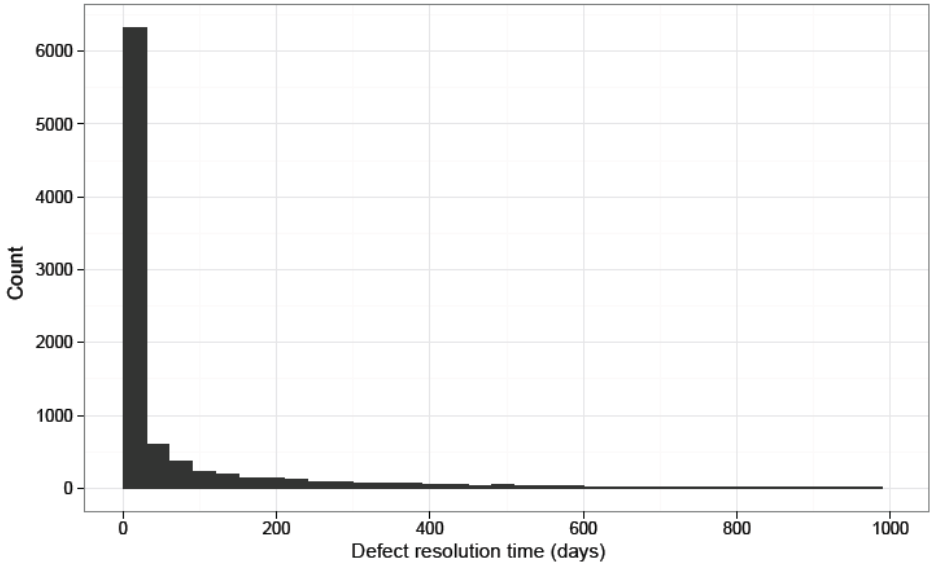
Maintenance

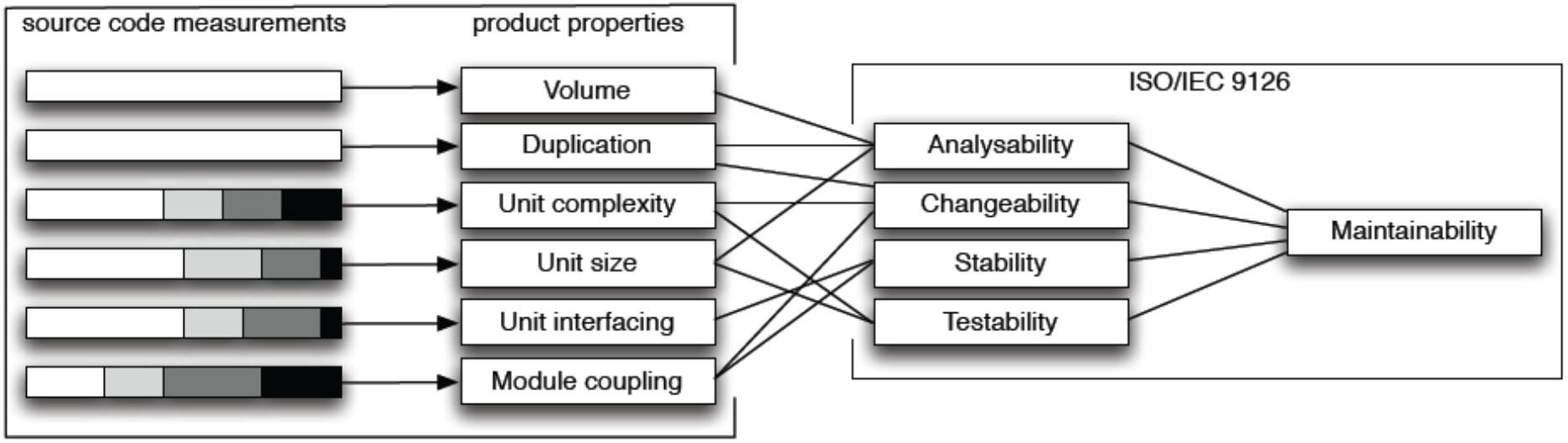
	Correction	Enhancement
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive



April. Studying Supply and Demand of Maintenance Services. QUATIC 2010

Maintainability



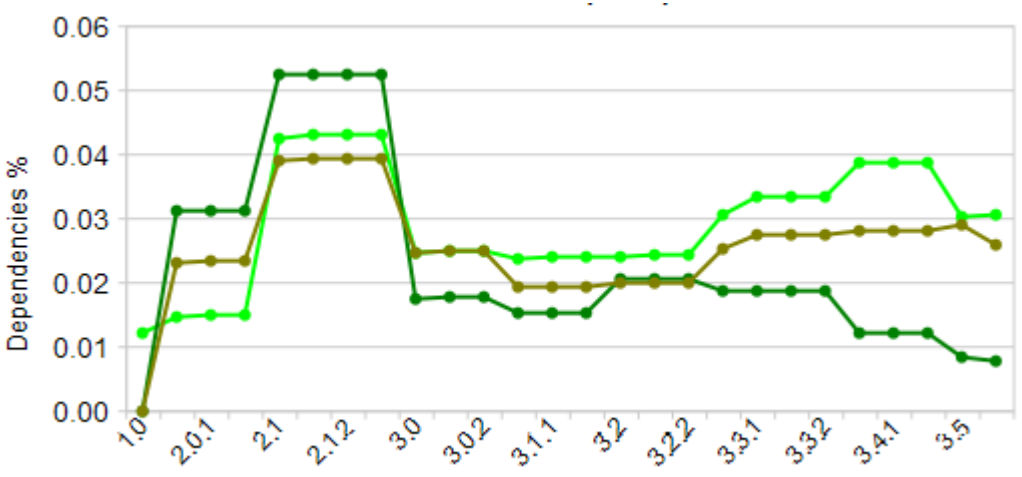


Luijter & Visser. Faster defect resolution with higher quality of software. SQM 2010

Stable Dependencies Principle

- Dependencies should be in the direction of stability

Martin. Large-scale stability. C++ Report 1997

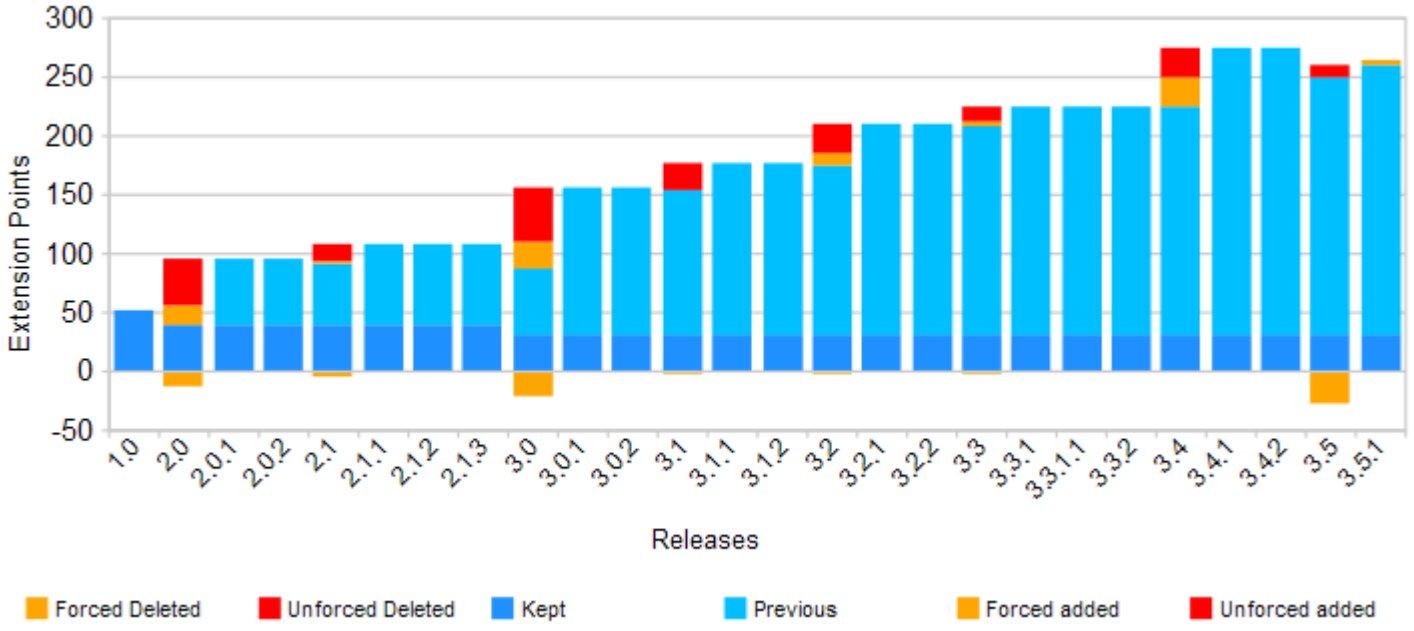


Wermelinger et al. Design principles in architectural evolution: a case study. ICSM 2008

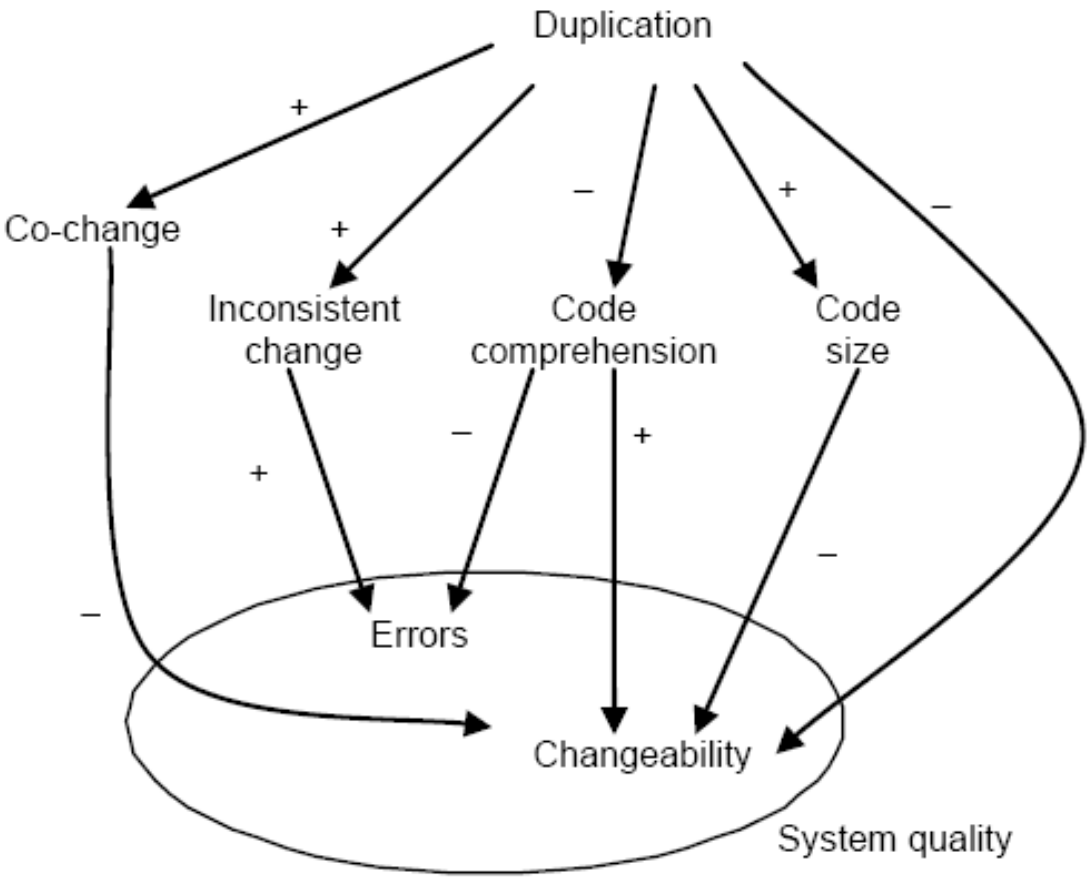
Open/Closed Principle

- Entities should be open for extension but closed for modification.

Meyer. Object-Oriented Software Construction. Prentice Hall 1988

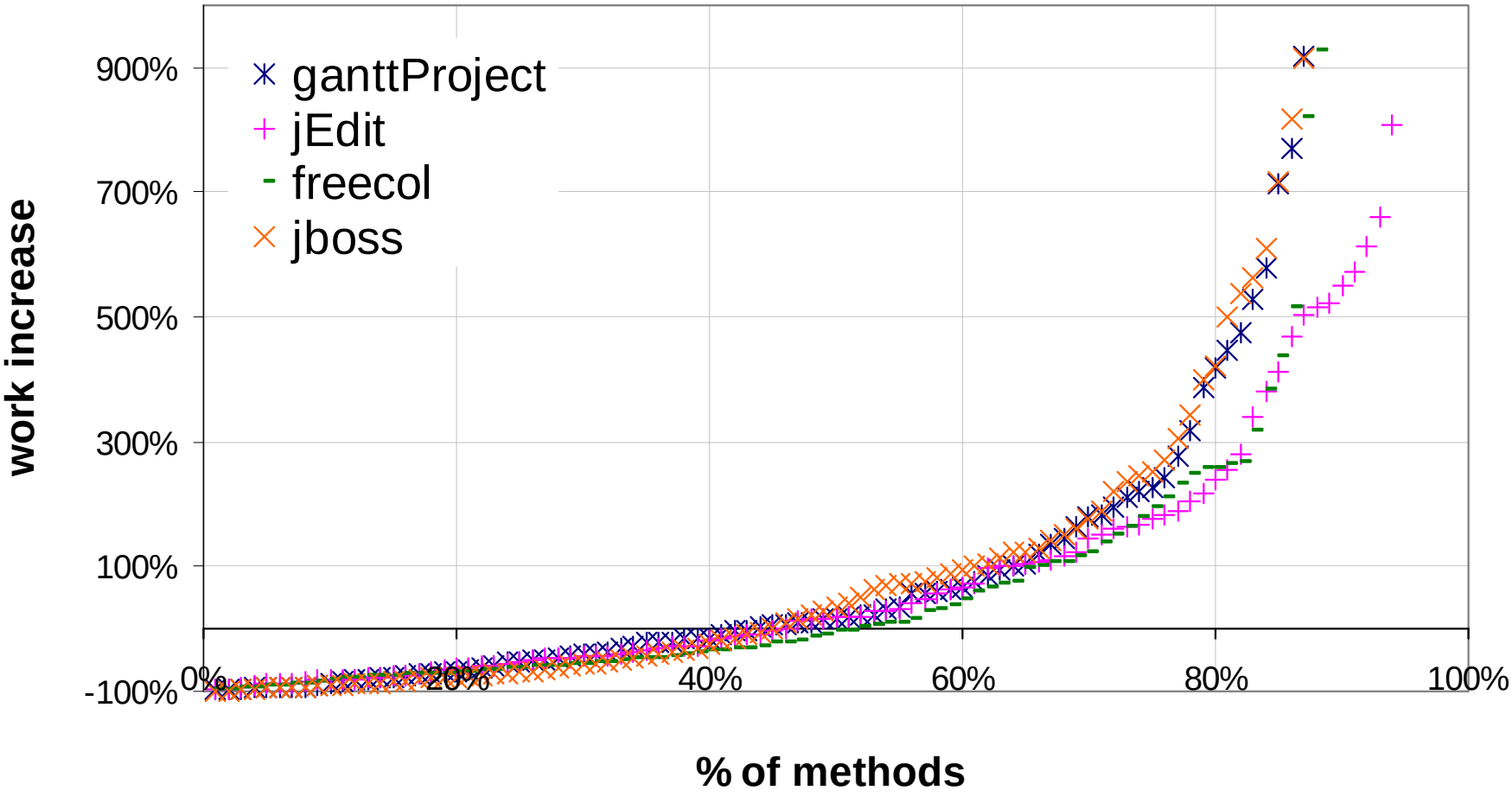


Cloning considered harmful



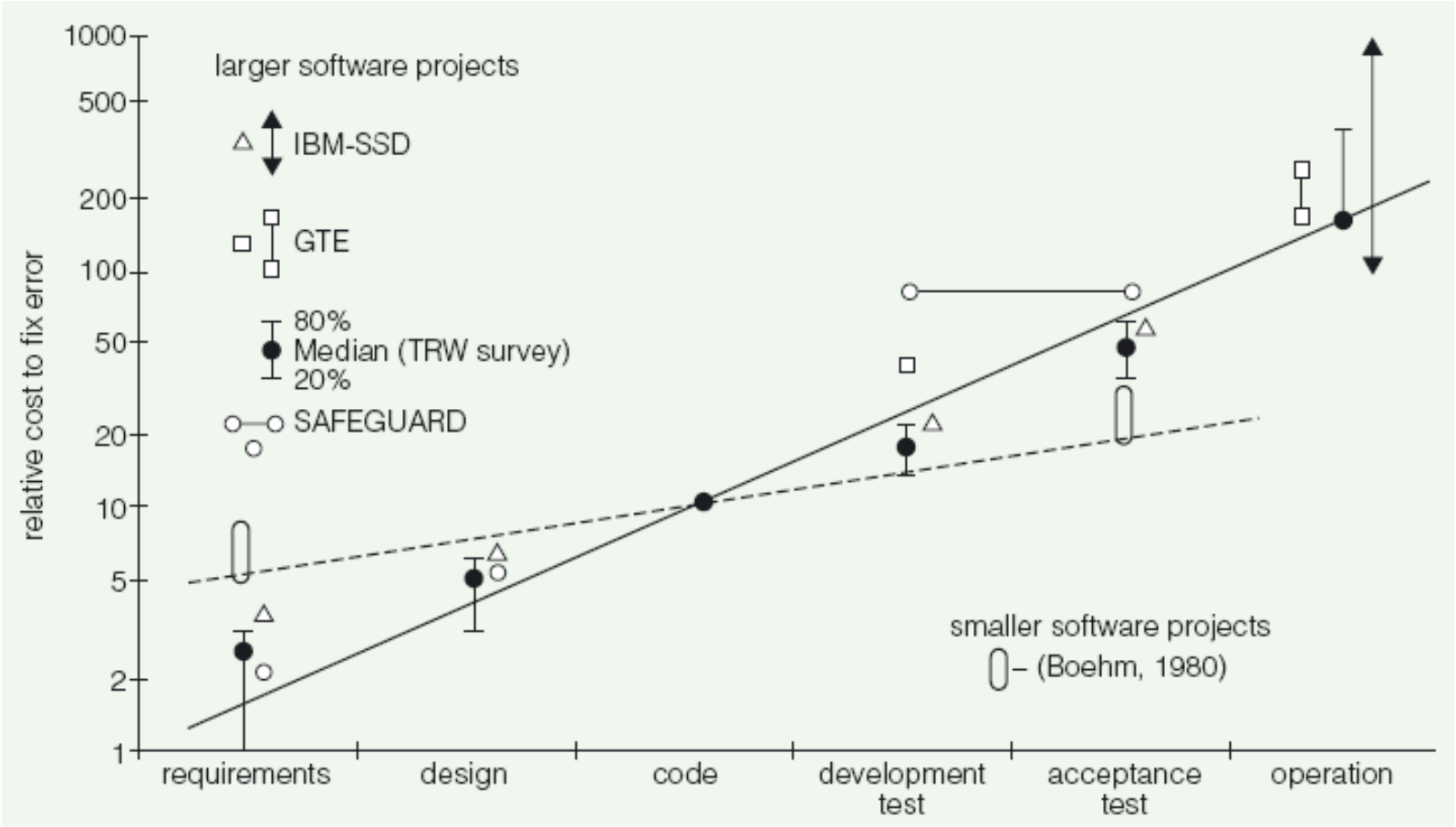
Hordijk et al. Harmfulness of Code Duplication - A Review of the Evidence, EASE 2009

Maintainability of clones



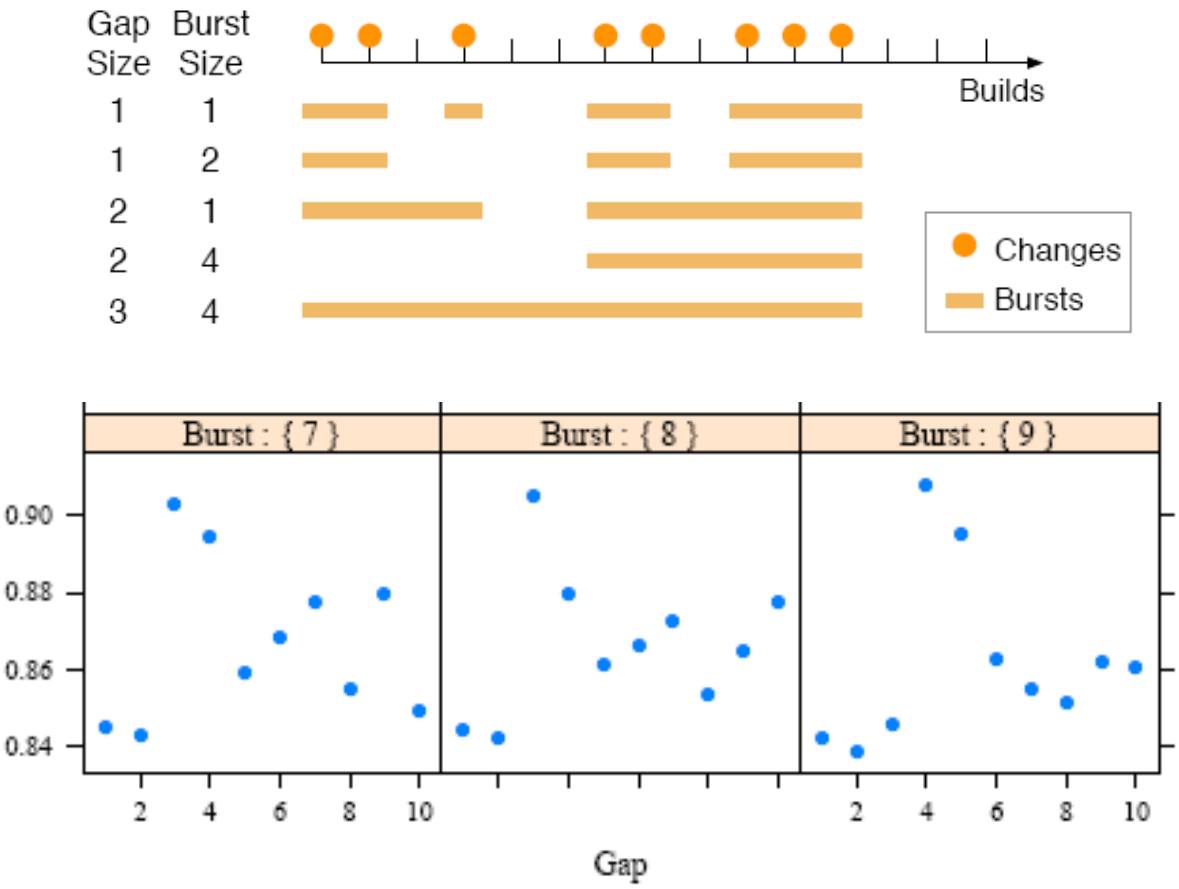
Lozano & Wermelinger. Assessing the effect of clones on changeability. ICSM 2008

Evolution of quality cost



after Fairley, Software Engineering Concepts, McGraw-Hill 1985

Evolution for quality prediction

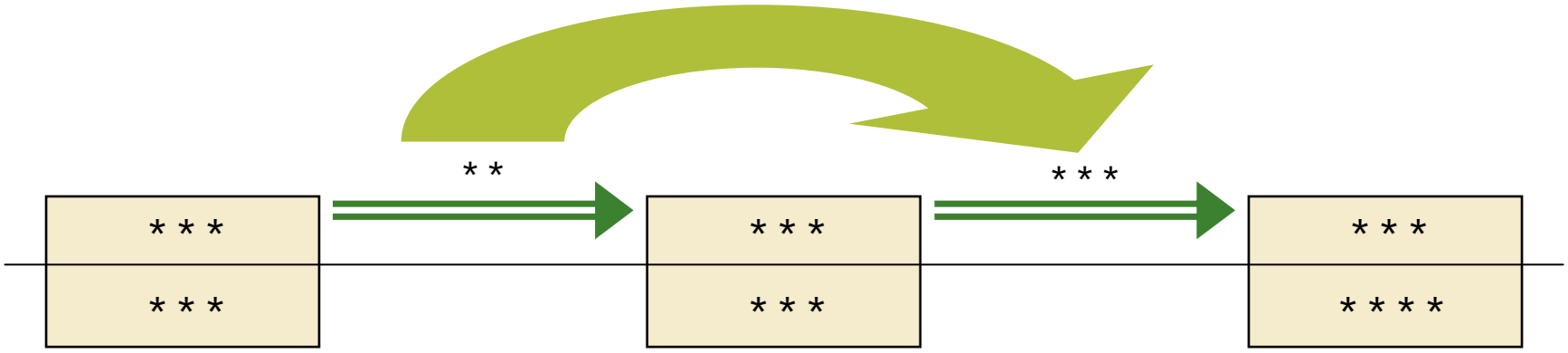


Nagappan et al. Change Bursts as Defect Predictors. ISSRE 2010

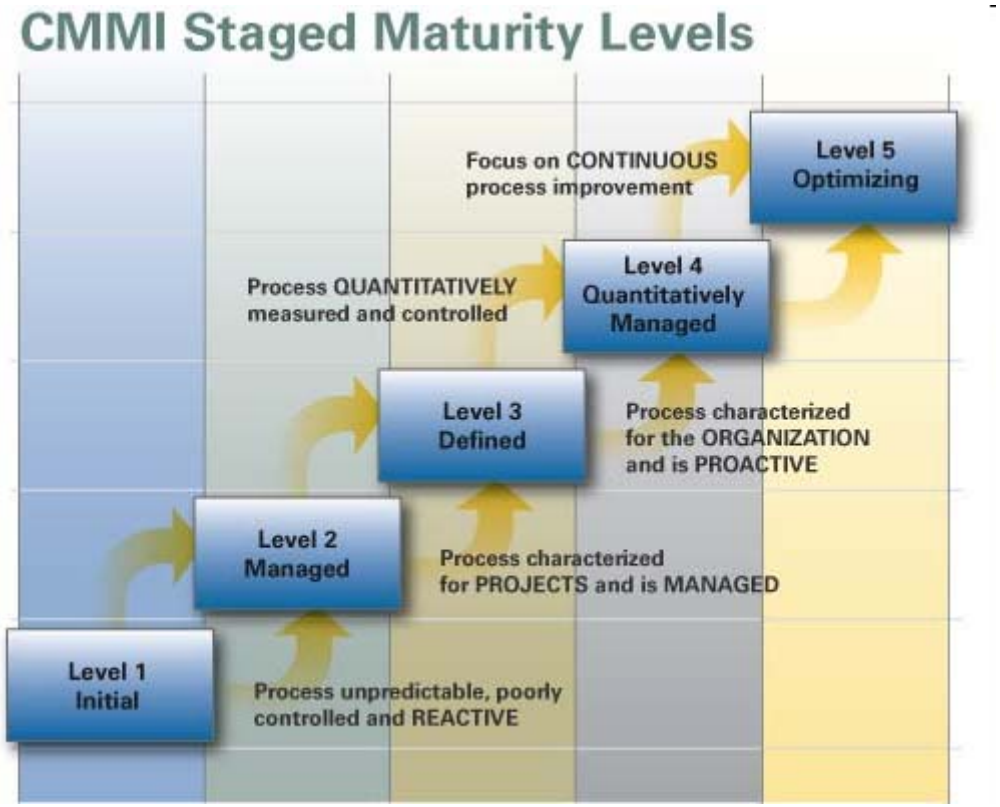
Process evolution

- Processes and applications are both executed, they both address requirements that need to be understood, both benefit from being modeled by a variety of sorts of models, **both must evolve guided by measurement**, and so forth.

Osterweil. Software processes are software too, revisited. ICSE 1997

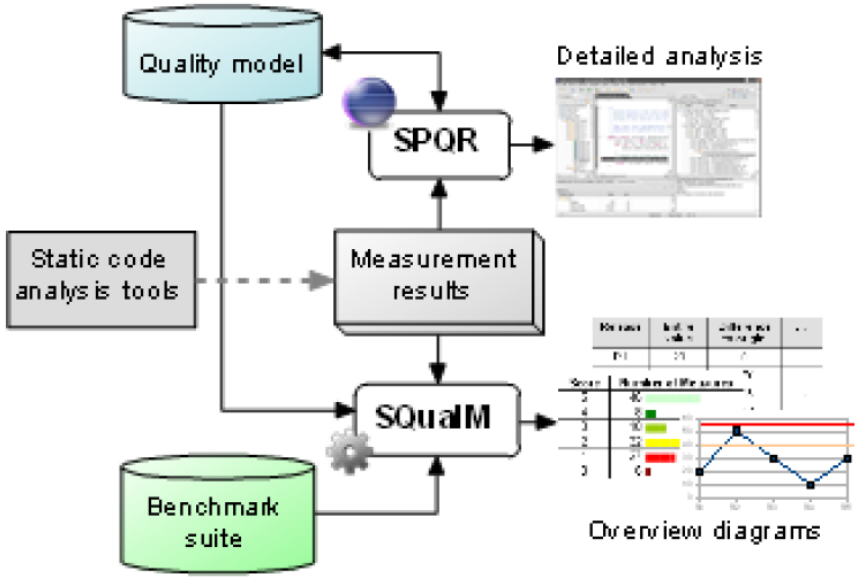


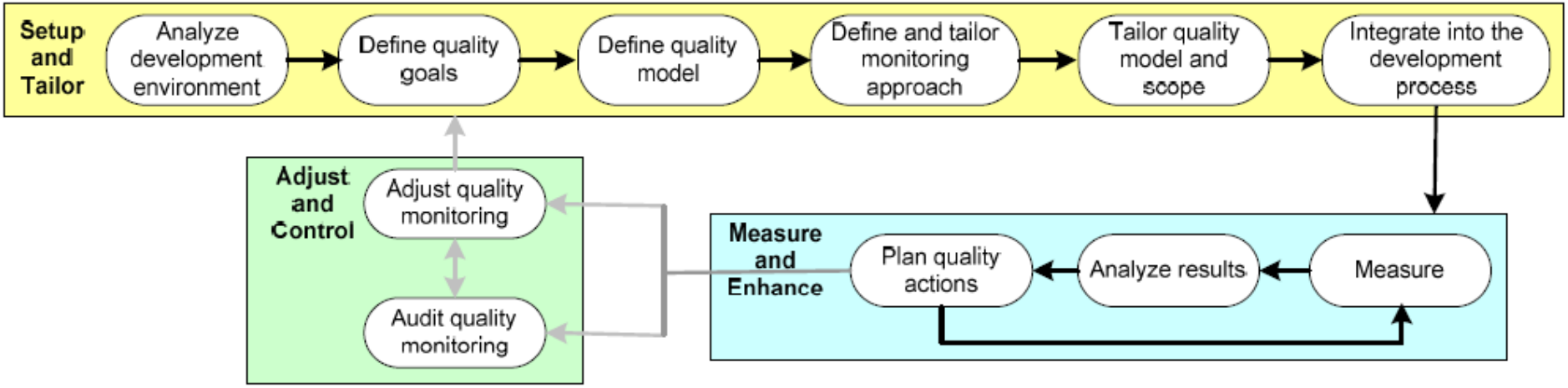
Process Improvement



from cmmilevels.com

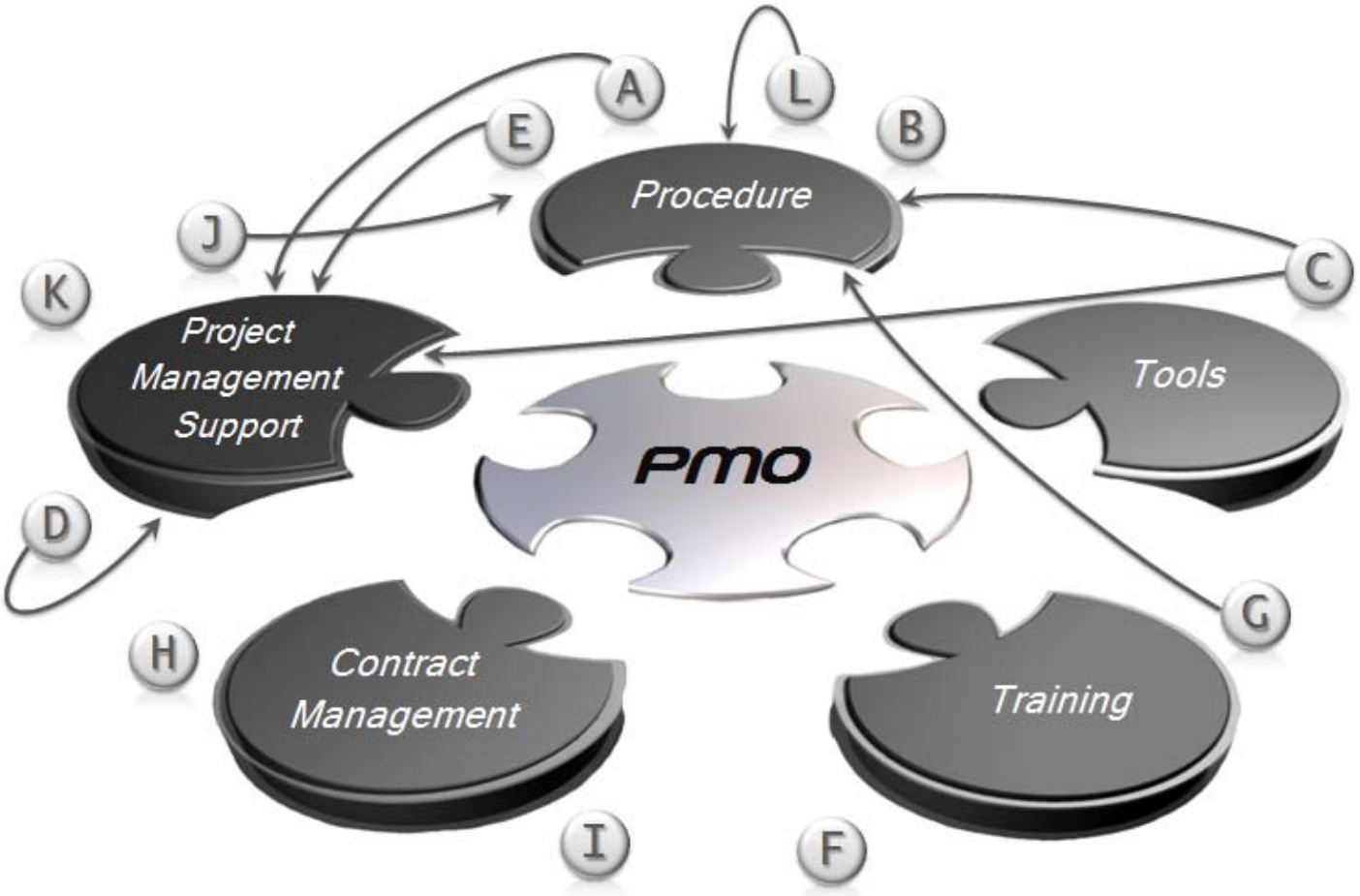
Quality management





Plösch et al. A method for continuous code quality management. QUATIC 2010

Process Quality Evolution



Bettin et al. A PMO Installation for IT Project Management. QUATIC 2010

Education

- Holistic view
  - Human / Social
  - Legal / Ethical
  - Economic

Conclusions

- E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems
  - 8th law of software evolution