

Sistemi Distribuiti

Corso di Laurea in Ingegneria

Prof. Paolo Nesi

PARTI: 3 – Ciclo di Vita, management, assessment

Department of Systems and Informatics

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

nesi@ingfi1.ing.unifi.it, nesi@dsi.unifi.it nesi@computer.org


www: <http://www.dsi.unifi.it/~nesi>



Ciclo di Sviluppo

- Cicli di Sviluppo
- Confronto ed Evoluzione, procedurale, OO
- Metodologie e ciclo di vita
- Gestione di Progetti
- Assessment di prodotto e di processo
- qualita' di processo e di prodotto
- Profilo di prodotto
- Processo con miglioramento continuo
- Qualita' di prodotto
- OO e ISO9126



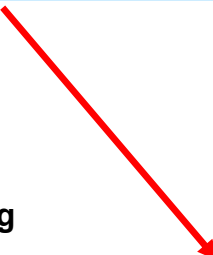





Development Life Cycle

- Waterfall
- Spiral
- Fountain
- Flipper, pinball
- Extreme programming
- No methodology !!!

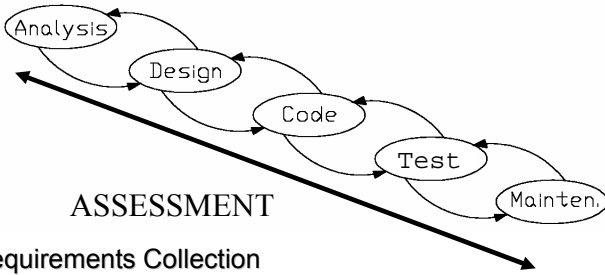
Development life-cycle model has to be technology independent





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004 3




Waterfall Life-Cycle



- Requirements Collection
- Requirement analysis
- Analysis, abstract design (composition/decomposition)
- Detailed Design (communication, HW details, behavior)
- Coding
- Testing (formal verification, simulation, etc.)
- Delivering → Maintenance





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004 4




From Traditional to Object Oriented Projects

- Uniform Model along the Project Development
- Object Oriented methodologies for Analysis and Design are supported by Evolutionary Life-Cycles: Spiral, Fountain, ..
- Middle out model: both bottom-up and top-down
- Prototype Oriented
- Different roles for PM, SSM, programmers
- Different management methodologies
- Different testing techniques
- Different assessment methodologies

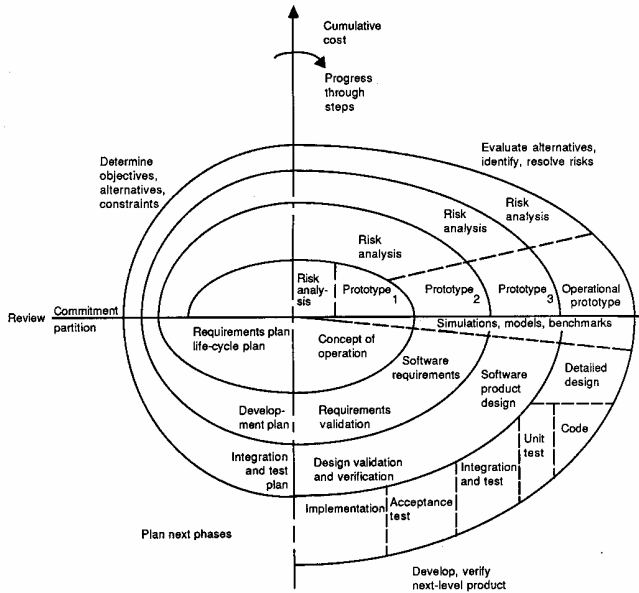



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004



5



Spiral Life-Cycle (Boehm 1986)



The diagram illustrates the Spiral Life-Cycle as a series of concentric, overlapping loops. Each loop represents an iteration. The vertical axis is labeled 'Cumulative cost' and 'Progress through steps'. The horizontal axis is labeled 'Review' and 'Commitment partition'. The spiral starts with 'Determine objectives, alternatives, constraints' and 'Requirements plan life-cycle plan'. Subsequent iterations include 'Risk analysis', 'Concept of operation', 'Software requirements', 'Software product design', 'Detailed design', 'Code', 'Unit test', 'Integration and test', 'Acceptance test', and 'Implementation'. The spiral ends with 'Operational prototype' and 'Develop, verify next-level product'. The diagram also shows 'Simulations, models, benchmarks' and 'Evaluate alternatives, identify, resolve risks' at various points in the cycle.

6

Fountain Life-Cycle (Henderson-Sellers 1993)

- Flexible reaction to problems
- Lack of well-defined process of development
- The deep of each iteration is not previously defined
- ...

SOFTWARE POOL

Real-World System

Sistemi Distribuiti, ...

7

Pinball Life-Cycle (Ambler, 1994)

- Undefined cycle
- any steps is possible
- no time constrains are imposed
- no measurable
- ...

Implementation

Projection

Sistemi Distribuiti, ...



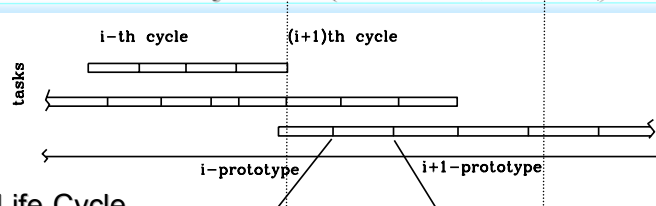
Extreme Programming (Beck, 1998)

- Pair Programming
- Test-first Design (5 minutes cycle, test cases)
- Refactoring (improvements for 5 minutes)
- Continuous Integration (CVS short time)
- Collective Ownership (CVS, RCS)
- Coding Standard (convention rules)
- 40 hours per week no more

Lack of controllability and rigorous assessment models



Macro and Micro Cycles (P. Nesi, 1998)



- Macro: Spiral Life-Cycle
- Each Cycle contains a micro
- Each micro is partially
 - ♣ Fountain, and
 - ♣ has parallel phases:
 - ➔ Assessment
 - ➔ Test
 - ➔ Documentation



Spiral Macro Cycle

- Well defined Functionalities for each spiral, early defined

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004

Task Relationships

- 3-4 Cycles (spiral) (micro-cycle relationships)
- micro-micro-cycle: 1-2 people, 5 minutes, etc..
- bi-Weekly meetings, CVS, etc.
- Monthly general meetings or when needed

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004



Development Methodologies

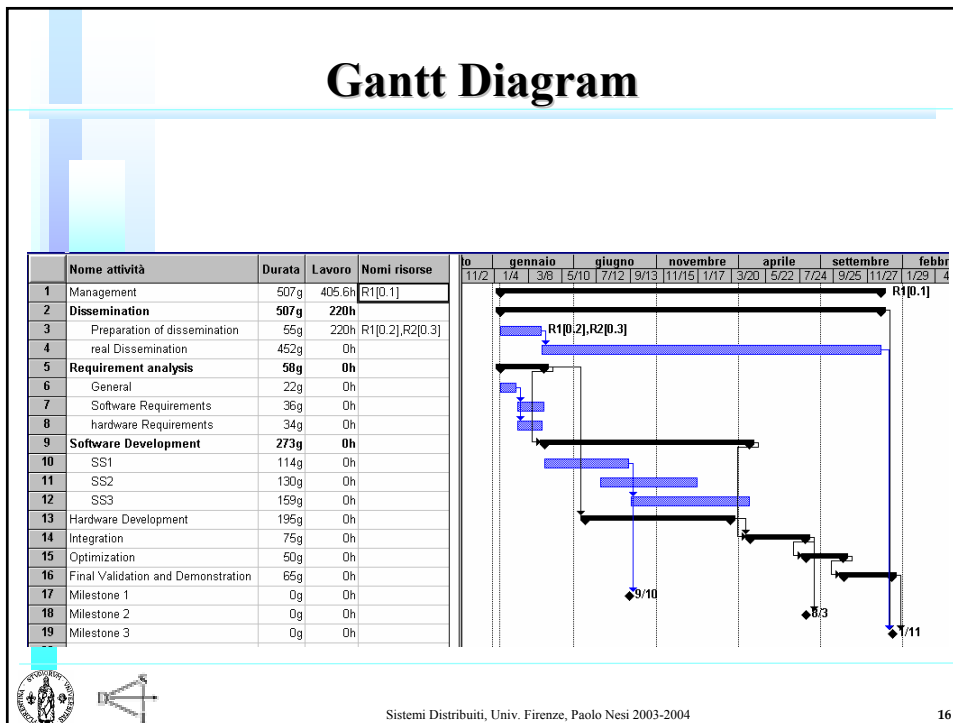
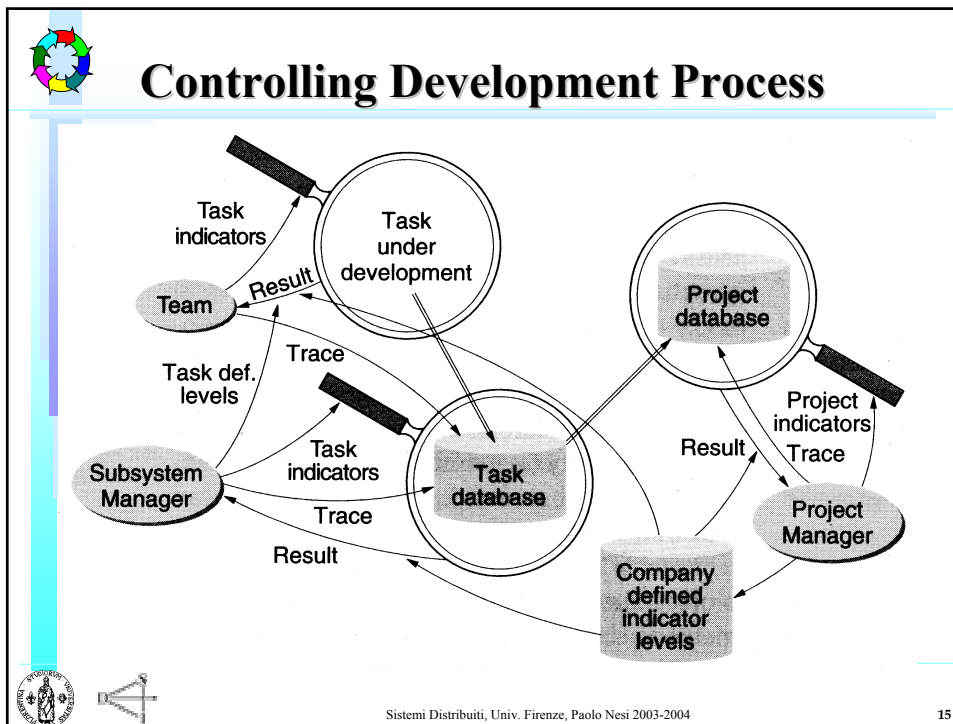
- Evolution of development process
- strongly dependent on the development paradigm
 - ♣ OO, design patterns, etc.
- No time for consolidating development models
 - ♣ technology changes too fast
- High investments are needed for adopting methodologies in deep
- Rapid evolution of technologies make them obsolete in short time



Management Methodology

- Take decision about process & products aspects:
 - ♣ development life cycle
 - ♣ development methodologies and technologies
 - ♣ allocation and resources and costs
 - ♣ business model and risk analysis
 - ♣ Heterogeneity (lang, model, arc.) management
 - ♣ project and phases duration (Gantt, Pert, etc.)
 - start up, allocation/staffing, etc.
 - development, testing, documentation,
 - assessment (process and product),
 - deployment, etc.
- Lack of suitable quality assessment and control methodologies







Process and Product Assessment

- **Process Assessment:**
 - ✦ Evaluation of the development process
 - ✦ Evaluation of the quality and the efficiency of the development process
 - ✦ definition of the compliant with the ISO 9000
 - ✦ CMM, etc..

- **Product Assessment:**
 - ✦ Process by which some features of the product are evaluated
 - ✦ Typical features are those derived from the product profile
 - ✦ Effort estimation and prediction
 - ✦ Quality prediction and estimation
 - ✦ <High Level> prediction and estimation

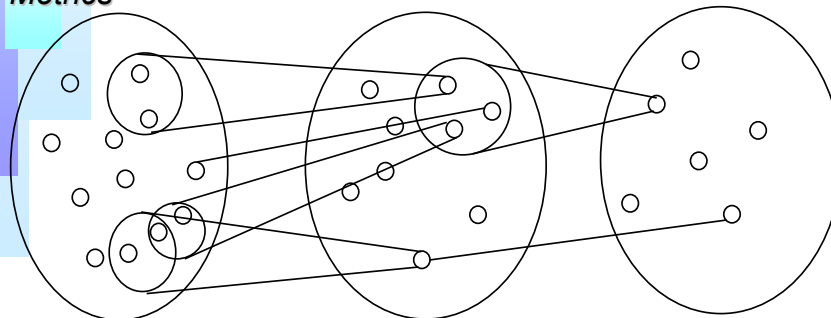


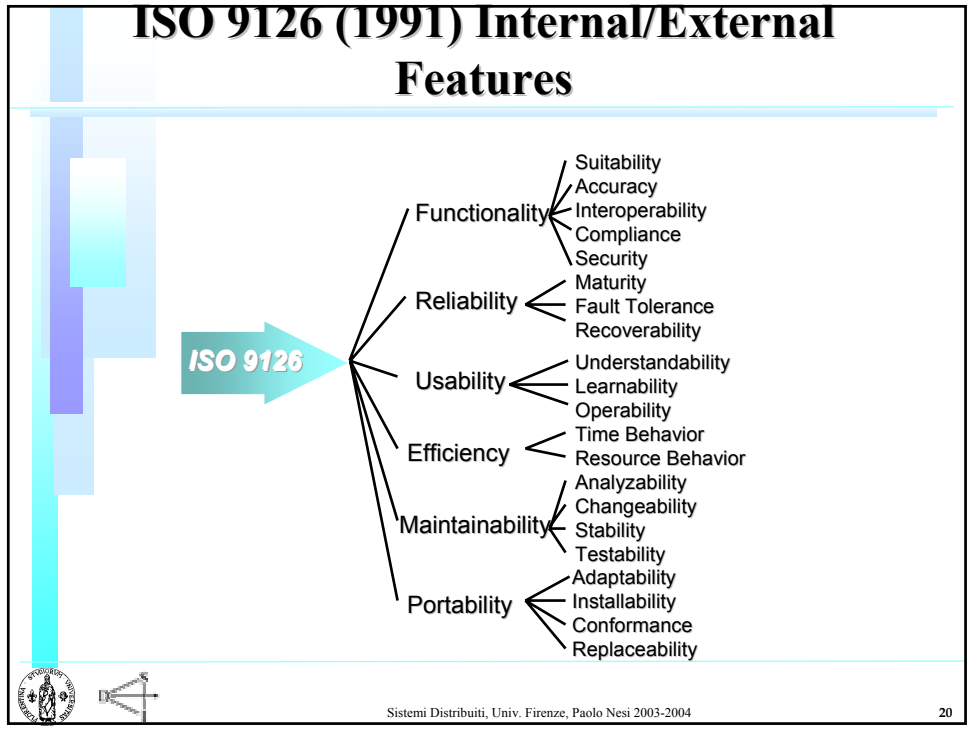
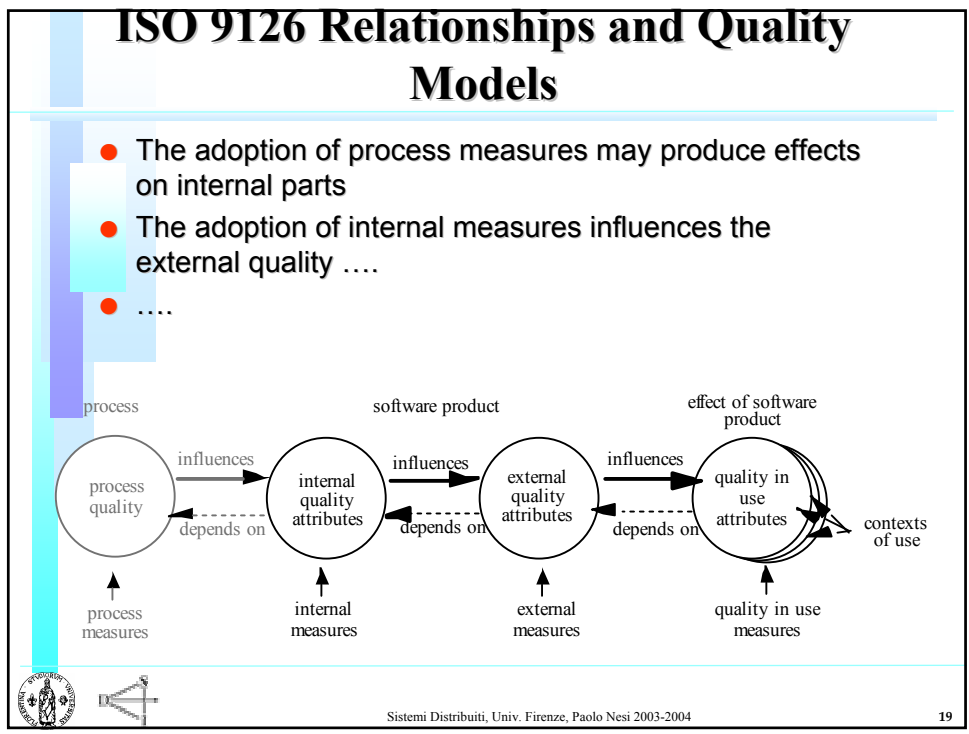
Metric Relationships

Technical/direct Metrics

Indirect Metrics

Features





Direct/Indirect Metrics

- direct metrics should produce a direct measure of parameters under consideration; for example, the number of the Lines of Code (LOC) for estimating the program length
- Indirect metrics are usually related to high-level characteristics; for example, the number of LOC is typically related to development effort
- the same measure can be considered as a direct and/or an indirect metric depending on its adoption.





Direct Metrics

- Size Metrics:
 - ♣ LOC: number of Lines of Code
 - ♣ number of language tokens
 - ♣ number of functions, operators, statements, modules,
 - ♣ number of comments, ...
 - ♣ number of ';' or 'CR' or 'LF' or 'begin'
- Complexity
 - ♣ Data Structure: number of variables, types
 - ♣ Logic: number of cycles, etc.
 - ♣ Computational: asymptotic, ..
 - ♣ Number of Nesting levels
 - ♣ Interface
 - ♣ Cognitive, psychological



Direct Metrics

- Functionals
 - ✦ number of functionalities
 - ✦ number of level of procedure call
 - ✦ number of use cases
 - ✦ number of sections in the manual
 - ✦ number of paragraph in the textual requirements
 - ✦ number of defects
 - ✦ ..
- In/Out
 - ✦ number of inputs, outputs
 - ✦ number of external variables
 - ✦ number of files
 - ✦ number of communication channels
 - ✦ number of buttons
 - ✦ ..





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004

23

Direct Metrics

- Cohesion/Coupling
 - ✦ number of calls for procedure
 - ✦ number of external variable of module
 - ✦ number of external variable of a function
 - ✦ number of calls out of the module
 - ✦ ..
- Object Oriented Structure
 - ✦ Inheritance hierarchy: balance, ramification, etc.
 - ✦ Distribution of metrics on the hierarchy
 - ✦ ..
 - ✦ ..



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2003-2004

24

Indirect metrics

- have to be validated for demonstrating their relationships with the corresponding high-level features.
 - (i) evaluating parameters of the metrics (e.g. weights and coefficients),
 - (ii) verifying the robustness of the identified model against several real cases.
- The model can be linear or not, and it must be identified by using both mathematical and statistical techniques



Indirect Metrics

- Indirect metrics have to consider the different scale:

$$M = w \langle \text{direct metric} \rangle$$

w has to correct the metric dimension.

- Metric should be normalized:

$$0 \leq M_i \leq 1$$

- Thus they can be compared with the values produced for other systems



Composite Metrics

- Most Indirect metrics are composite metrics.
- These are derived as the composition of other direct, indirect metrics with different scales:

$$CM = w1 M1 + w2 M2$$

- $w1, w2$ are used to correct the metric dimension and should be defined or estimated during validation.
- In this case a linear dependency has been supposed !
- Composite Metric should be normalized: $0 \leq CM \leq 1$ or referenced to the features under estimation scale.

