Università degli Studi dell'Insubria

Challenges in Dedicated Software Engineering: Software Engineering for Management

Luigi Lavazza

Dipartimento di Informatica e Comunicazione

Università degli Studi dell'Insubria

Varese, Italy

Motivations

Software size measurement continues to be a significant practical problem in software engineering. Without a solid baseline of size neither estimation and planning nor control for large scale software projects is possible. As a result, estimation errors are reported to be essential causes of poor management

incipit from

C. Gencel and O. Demirors. Functional Size Measurement Revisited. *ACM Transactions on Software Engineering and Methodology*, 17(3), 2008.

Motivations

Being able of successfully developing software does not require only to know development techniques, methodologies, tools, etc.

You must be able to produce software of the required quality in convenient time and at reasonable costs.

From the presentation of my course on Software project management.

Motivations

Cost and schedule overruns occur quite frequently in the software industry and are a primary cause of project failures.

Only 9% of projects in large companies were successful. At 16.2% and 28% respectively, medium and small companies were somewhat more successful. A whopping 61.5% of all large company projects were challenged compared to 46.7% for medium companies and 50.4% for small companies. The most projects, 37.1%, were impaired and subsequently cancelled in medium companies, compared to 29.5% in large companies and 21.6% in small companies.

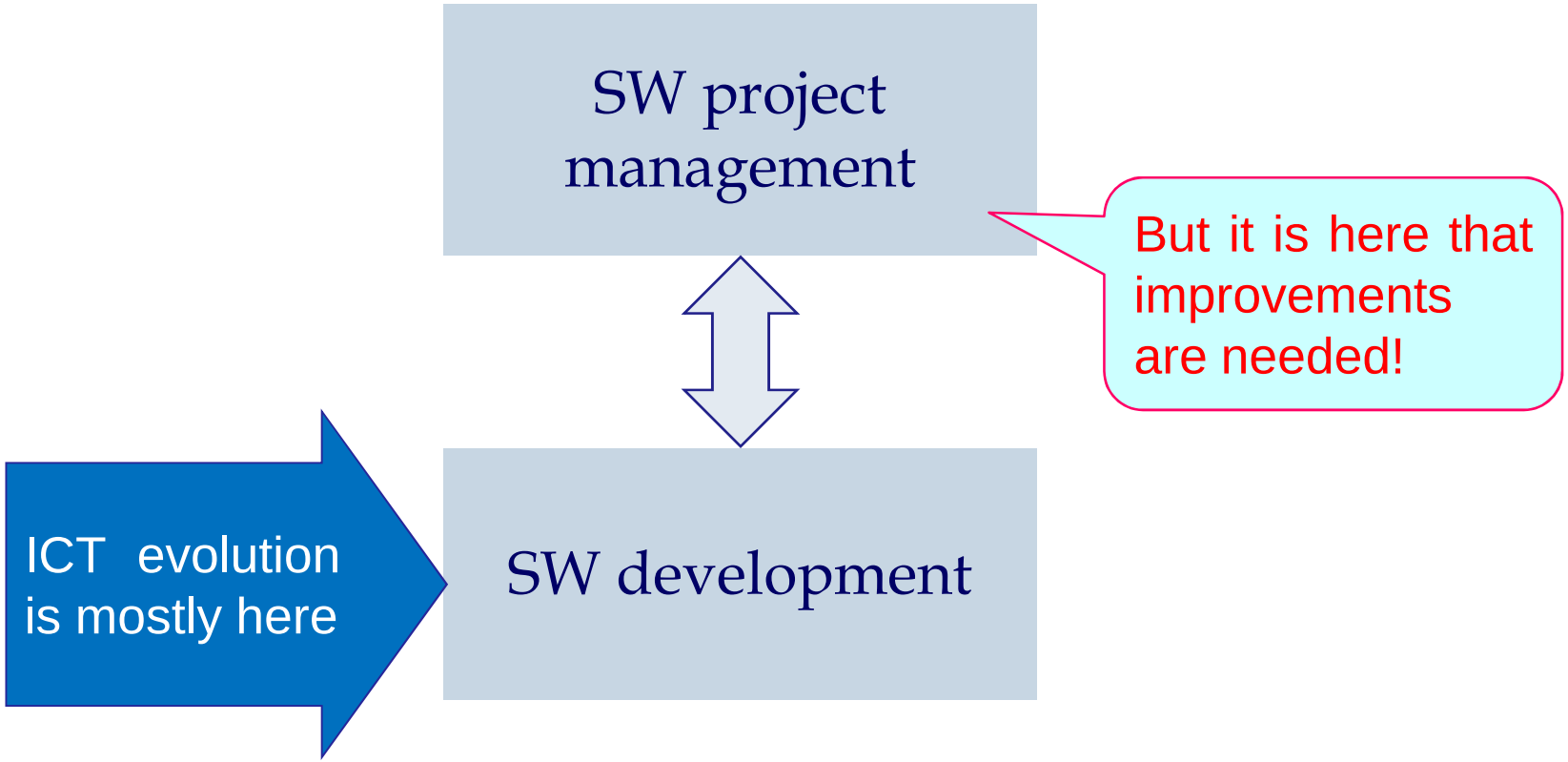
The Standish Group. Chaos Summary 2009.

Opinions about why projects are impaired and ultimately cancelled

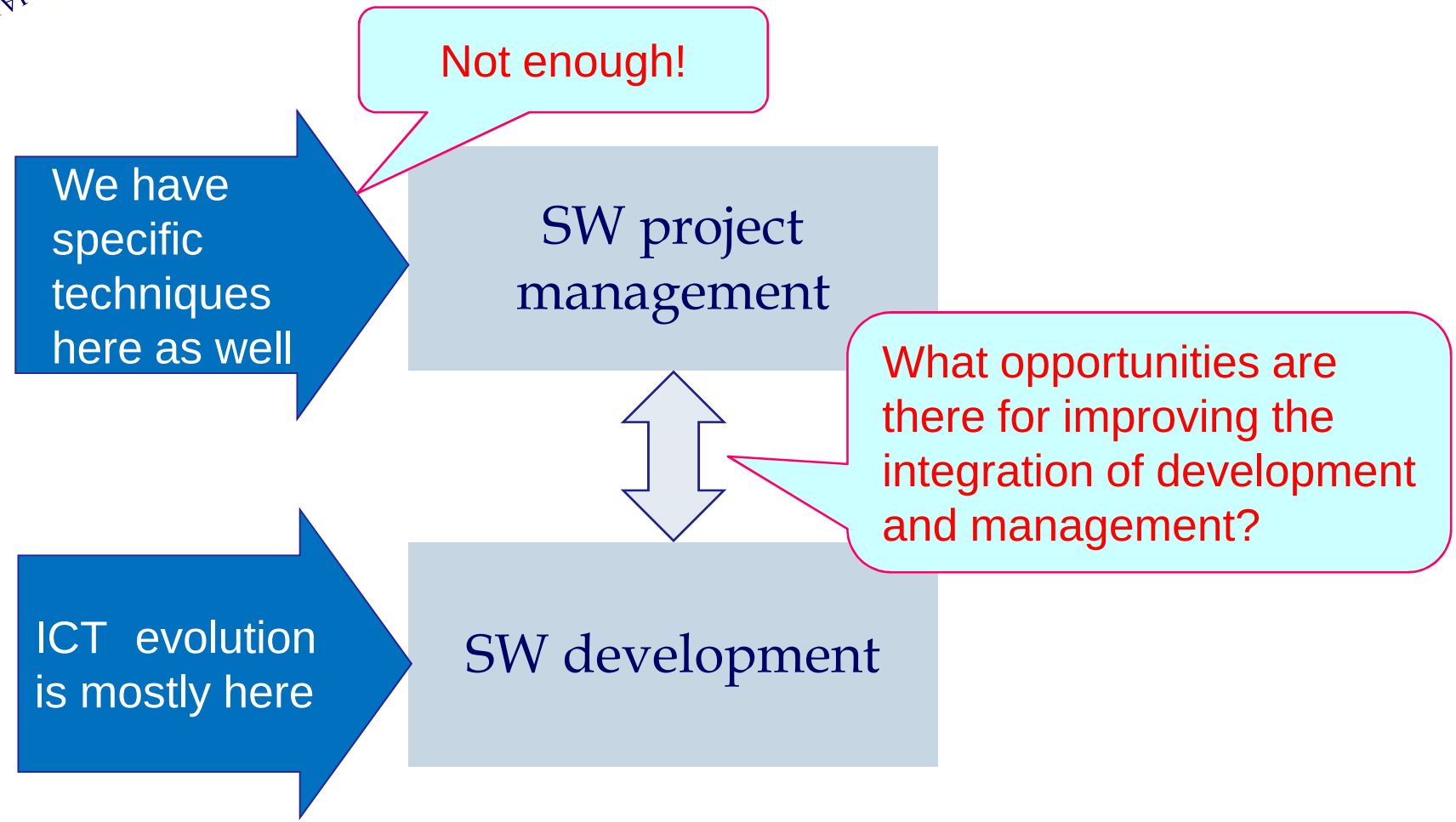
Project Impaired Factors	% of Responses
1. Incomplete Requirements	13.1%
2. Lack of User Involvement	12.4%
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specifications	8.7%
7. Lack of Planning	8.1%
8. Didn't Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

Hardly any technical motivation!

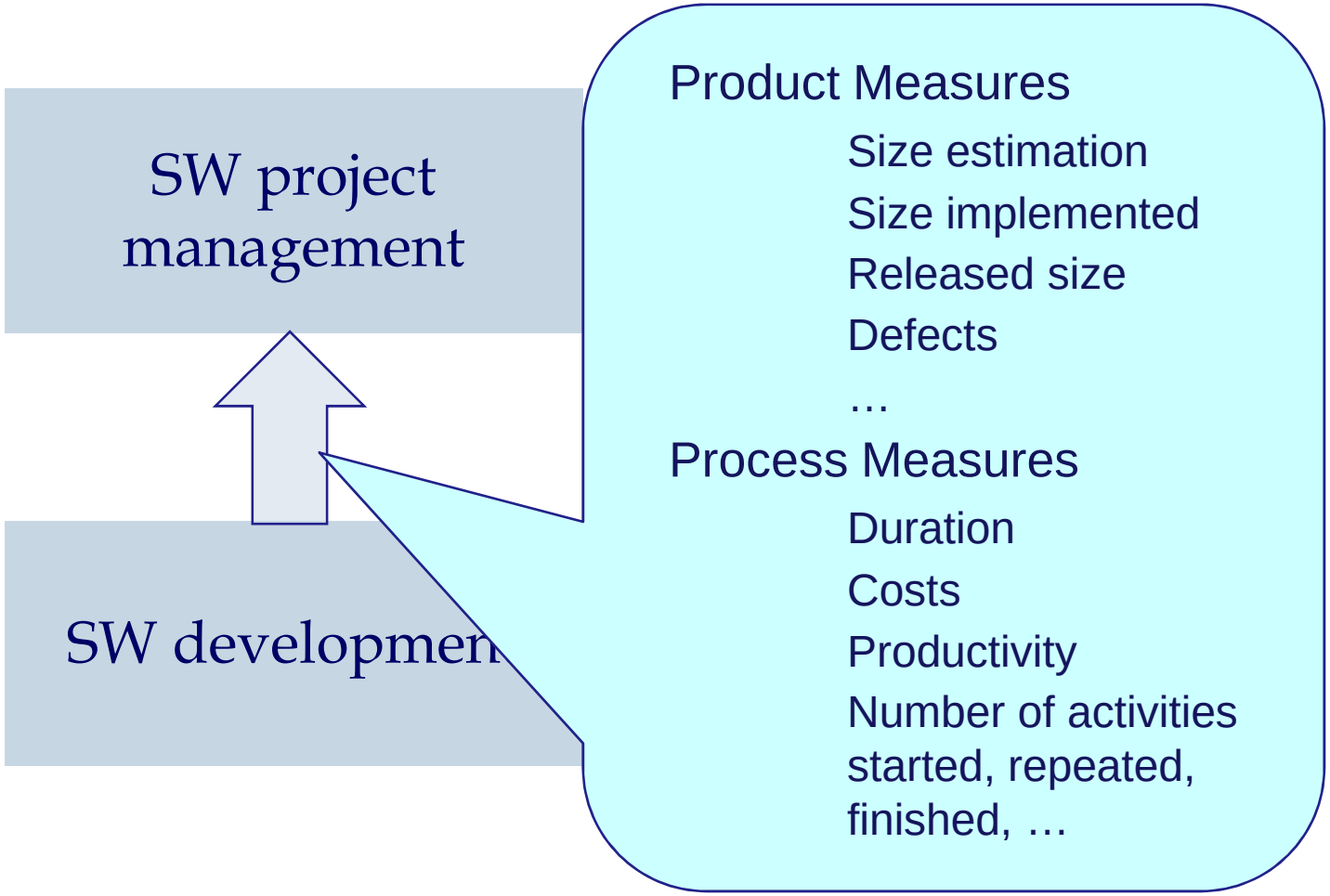
Situation


SW project management
But it is here that improvements are needed!
ICT evolution is mostly here
SW development

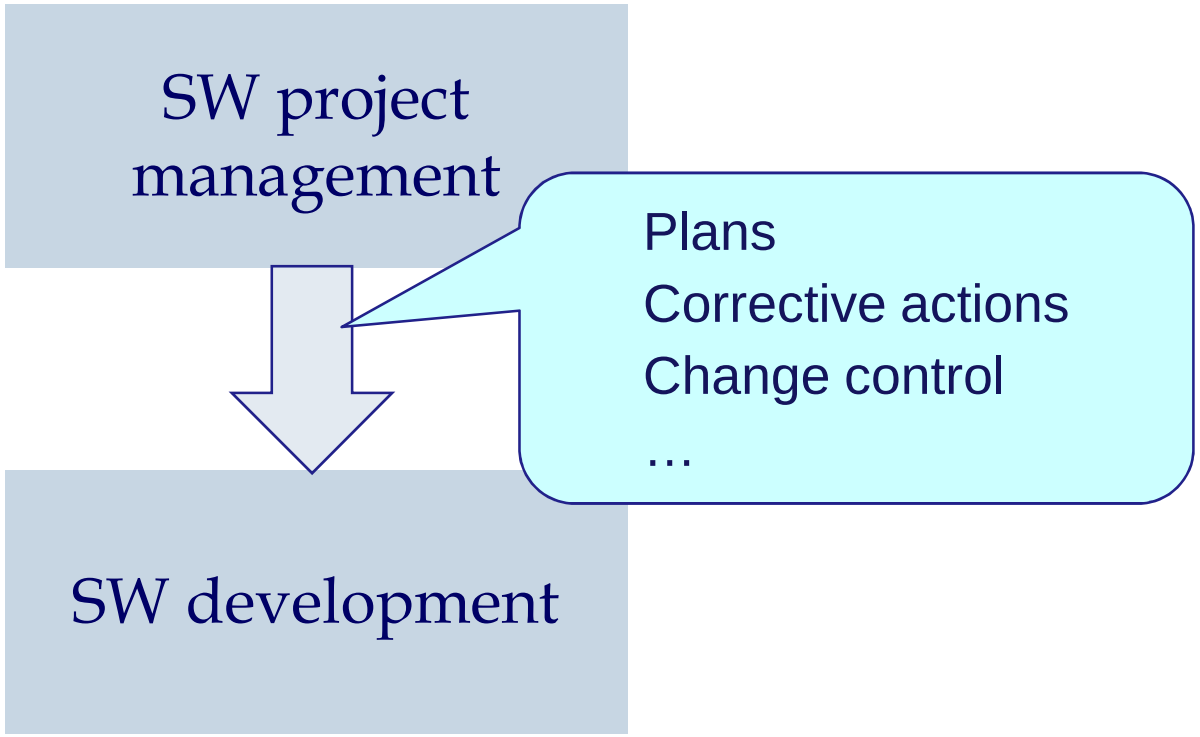
Situation


Not enough!
We have specific techniques here as well
SW project management
What opportunities are there for improving the integration of development and management?
ICT evolution is mostly here
SW development

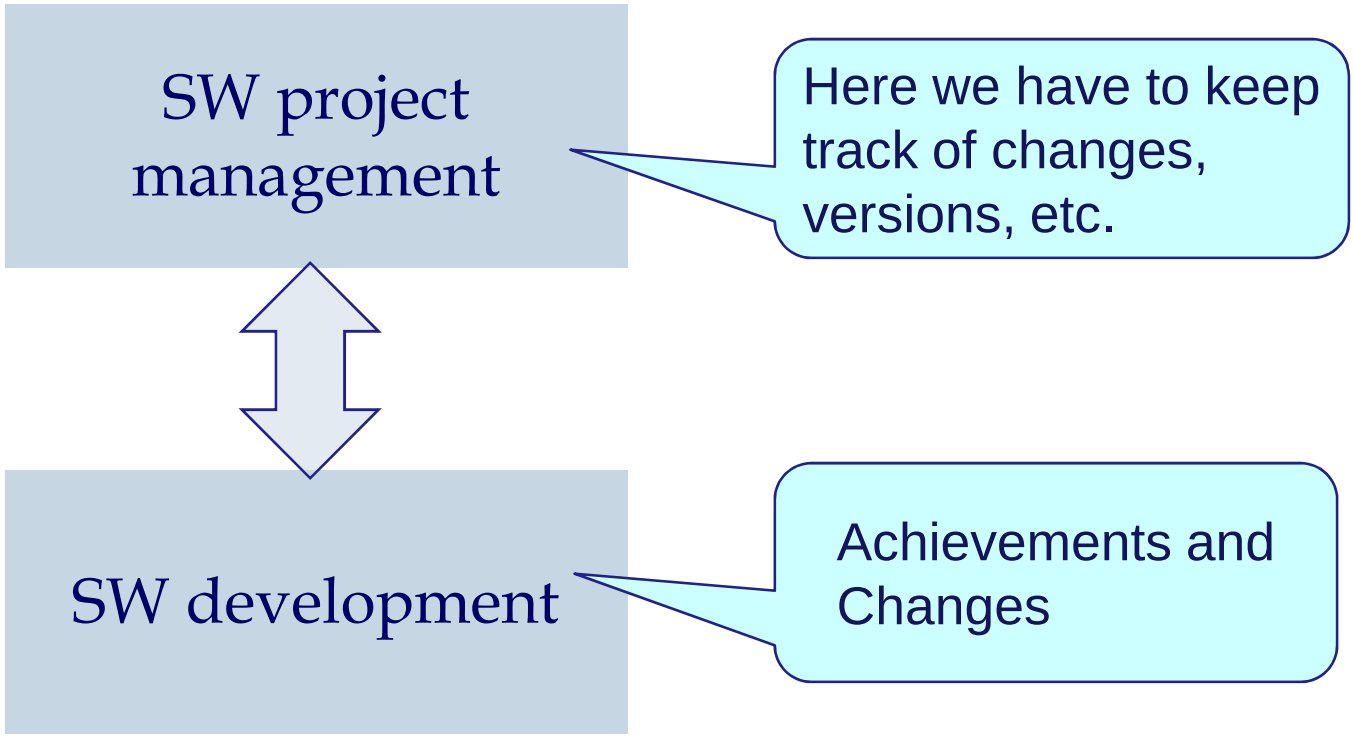
Communication



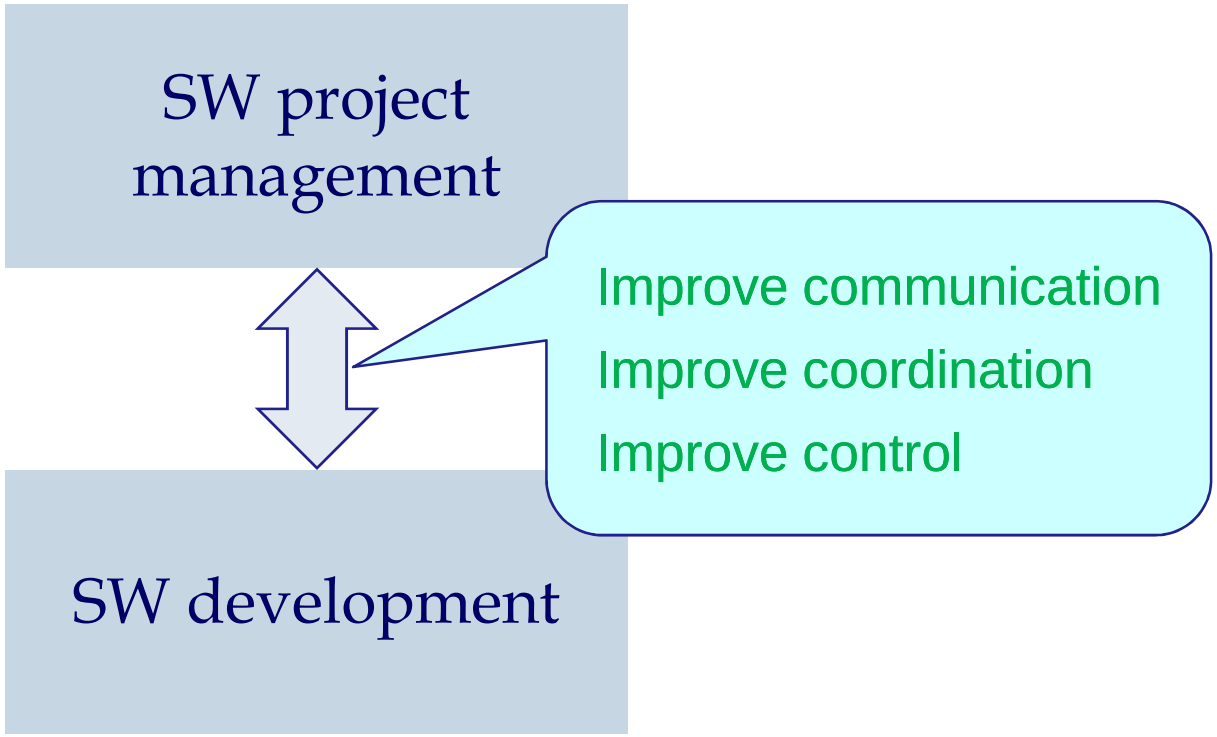
Control


SW project management
Plans
Corrective actions
Change control
…
SW development

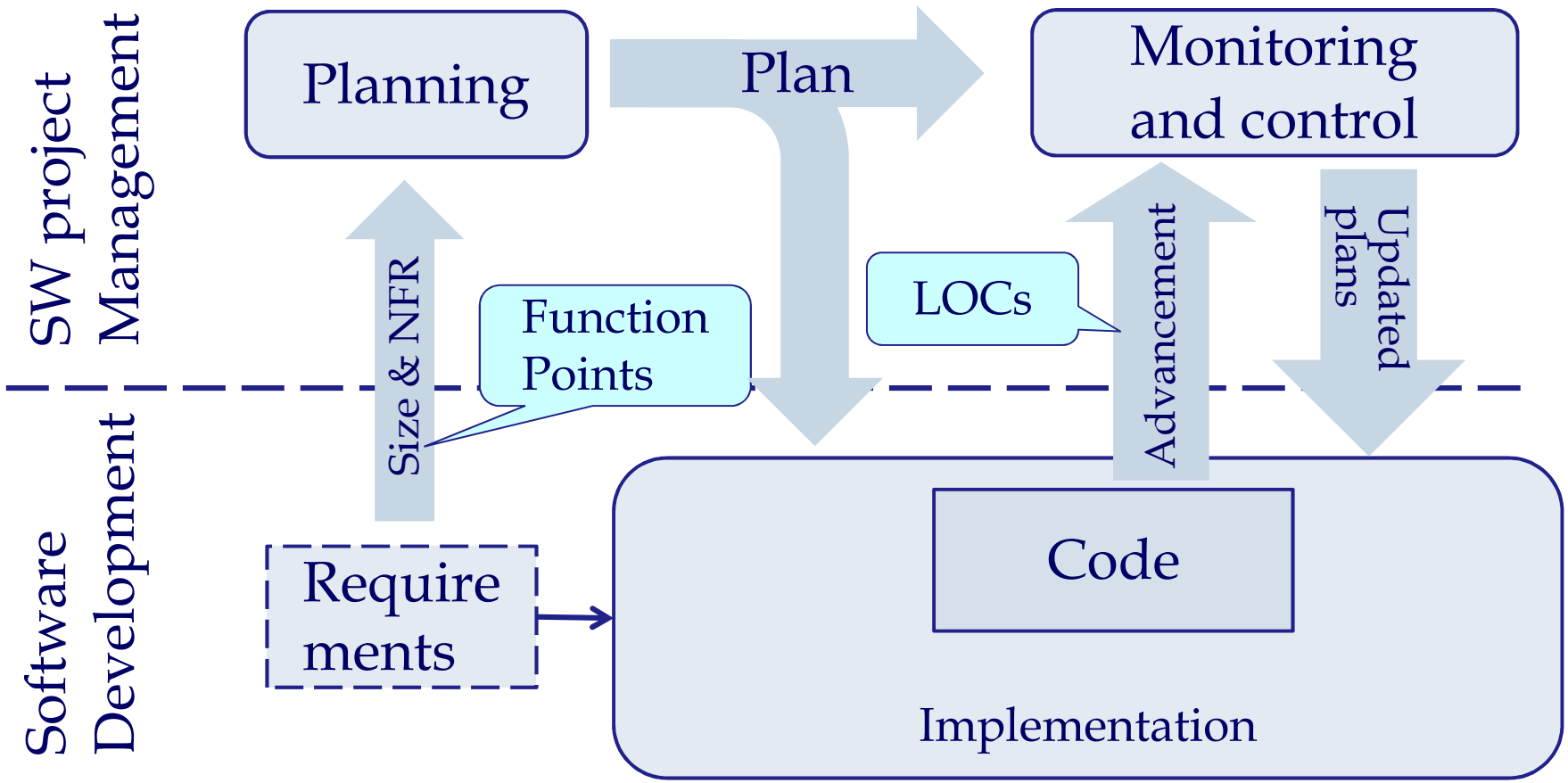
Coordination



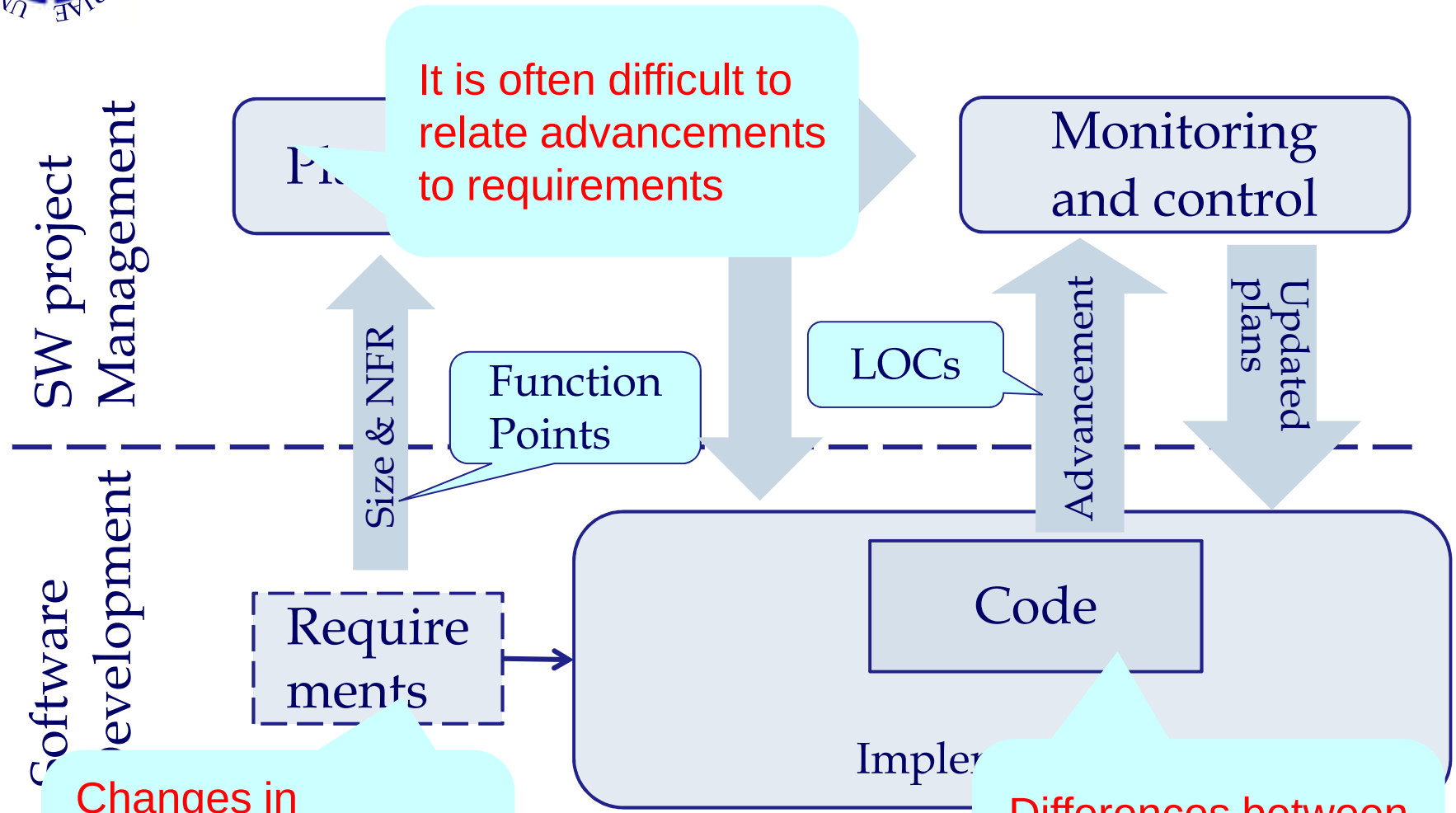
Opportunities



The current situation

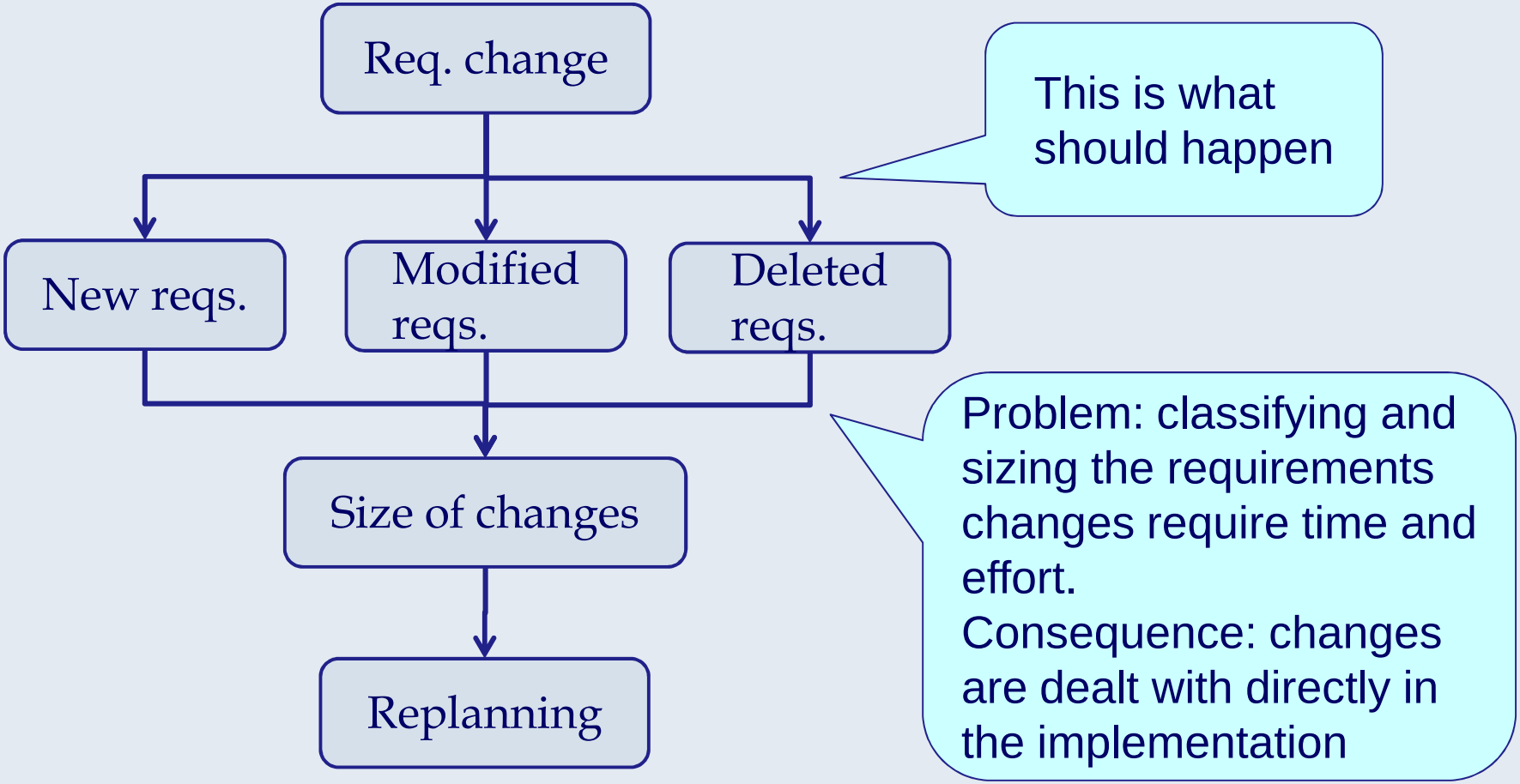


Problems with the current situation



Problems with the current situation (1)

- Changes in requirements are not always properly modeled and sized



Problems with the current situation (2)

- It is often difficult to relate advancements to requirements
  - Advancements are measured in LOC, requirements are measured in Function Points.
  - Relating a piece of code to a piece of requirement calls for traceability.
  - Proper versioning is required.
  - Measuring size in LOC is easy (thus it is often done) but is not sufficient. Additional measures are needed to support management:
    - How much has been the code reviewed?
    - How much has it been tested?
    - What is its McCabe complexity and what is the coverage of tests?
    - How many defects were found?
    - How much effort did it require?
    - …

Problems with the current situation (3)

- Differences between versions are usually not properly sized
  - Very often, the “advancement” is measured as the total lines of code accumulated to date.
  - Sometimes the data are not available at the file granularity.
  - When data are available at the file granularity, it is usually not known what are the differences between two subsequent versions.
    - E.g., the files are measured once a month, but in a month several independent changes (typically related to different requirements) are applied to the file.
  - Very seldom you have data on the lines added, deleted or modified.

So, what do we need?

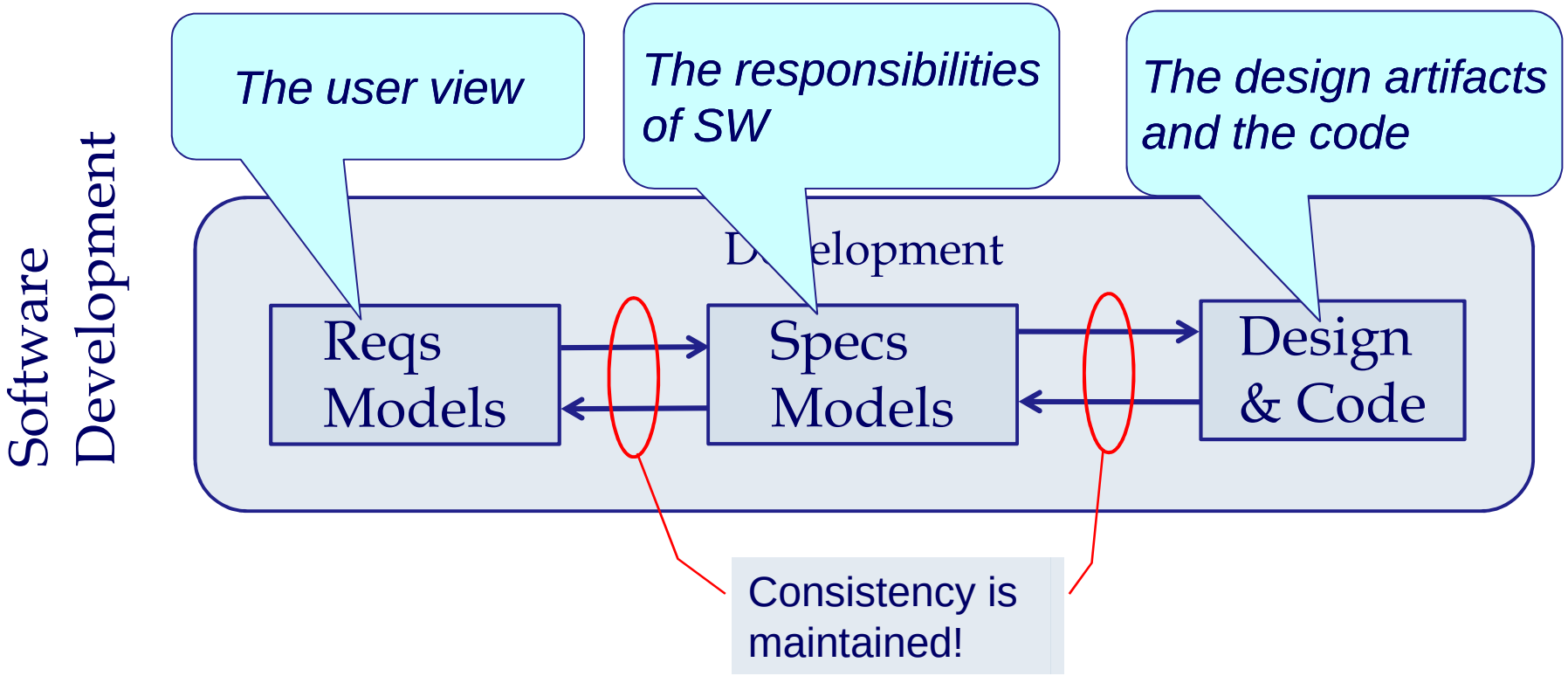
- Homogeneous representations of artifacts in requirements management and development; homogeneous measures.
- Configuration management applied to all the software artifacts; traceability.
- A sort of “semantic” configuration management. The entities and relationships being versioned are typed. Intelligent behavior can be associated to changes.
- Historical data. *Everything* should be measured and recorded.

How?

- Homogeneous representations of artifacts in requirements management and development; homogeneous measures.
- These objectives can be achieved by promoting the use of models.
- Up to now, modeling has been seen as a tool for supporting development (see MDA).
- But models can be used to support management too!
  - Good models make measurement easier and more reliable
  - Models favor the representation of relationships and dependences
  - Models can be used in any phase of the development process.

Promoting the use of models (1)

Technique: problem frames
Language: UML
Principle: W,S → R; P,C → S



Measures

- The final target is satisfying the requirements.
- The measure of the requirements quantifies the goal.
  - Technique: Measurement-oriented modeling & model based measurement
    - [Luigi Lavazza, Vieri del Bianco, Carla Garavaglia, "Model-based Functional Size Measurement", *ESEM 2008, 2nd International Symposium on Empirical Software Engineering and Measurement*, Kaiserslautern. October 9-10, 2008.]
- The measure of advancement is given by the percentage of model that has been implemented
  - Technique: SCM applied to the mentioned models and their relations

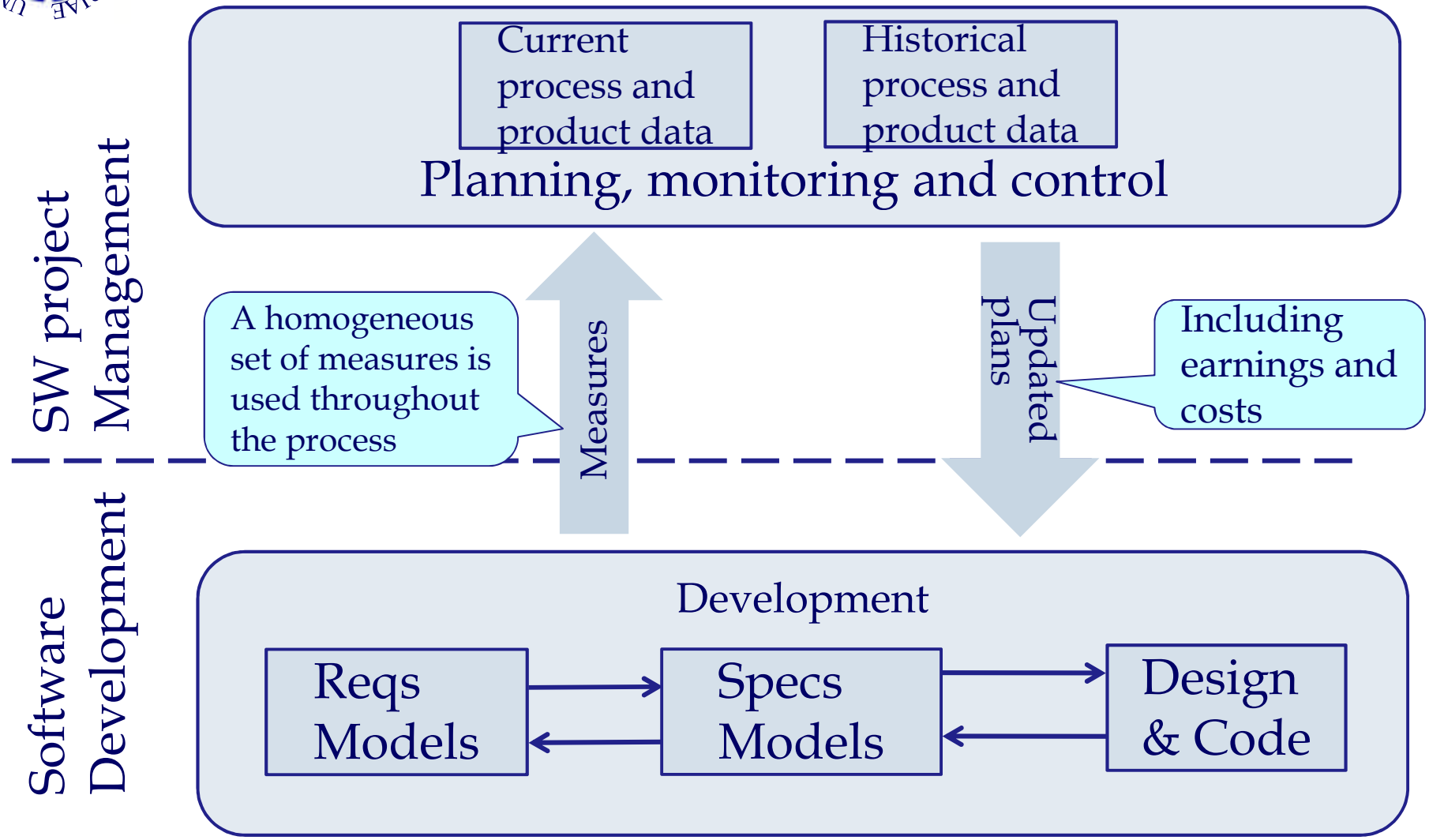
Dealing with changes

- All changes produce new versions.
- New versions are properly interrelated (using also the relations of the previous versions)
- The representation of dependencies is favored by the homogeneity of the notation
  - E.g., a class that represents information relevant to the user in the model of requirements is likely to correspond to a similar class in the model of design.
- If the elements of the models are properly typed, it is easy to characterize the changes
  - E.g., a class diagram is changed in terms of classes added/modified/deleted; a class is changed in terms of attributes, operations and relations added/modified/deleted, etc.

Process measures

- Development activity are measured in terms of
  - Volume of change
  - Effort spent
  - …
- Moreover, they are classified in terms of goal
  - Implementing a requirement
  - Implementing a required change
  - Correcting a defect
  - …
- Used in conjunction with product measures, they provide correct productivity measures
  - E.g., how much effort is required for performing a change of a given type to achieve a goal of a given size on a piece of product having given size, complexity, etc.

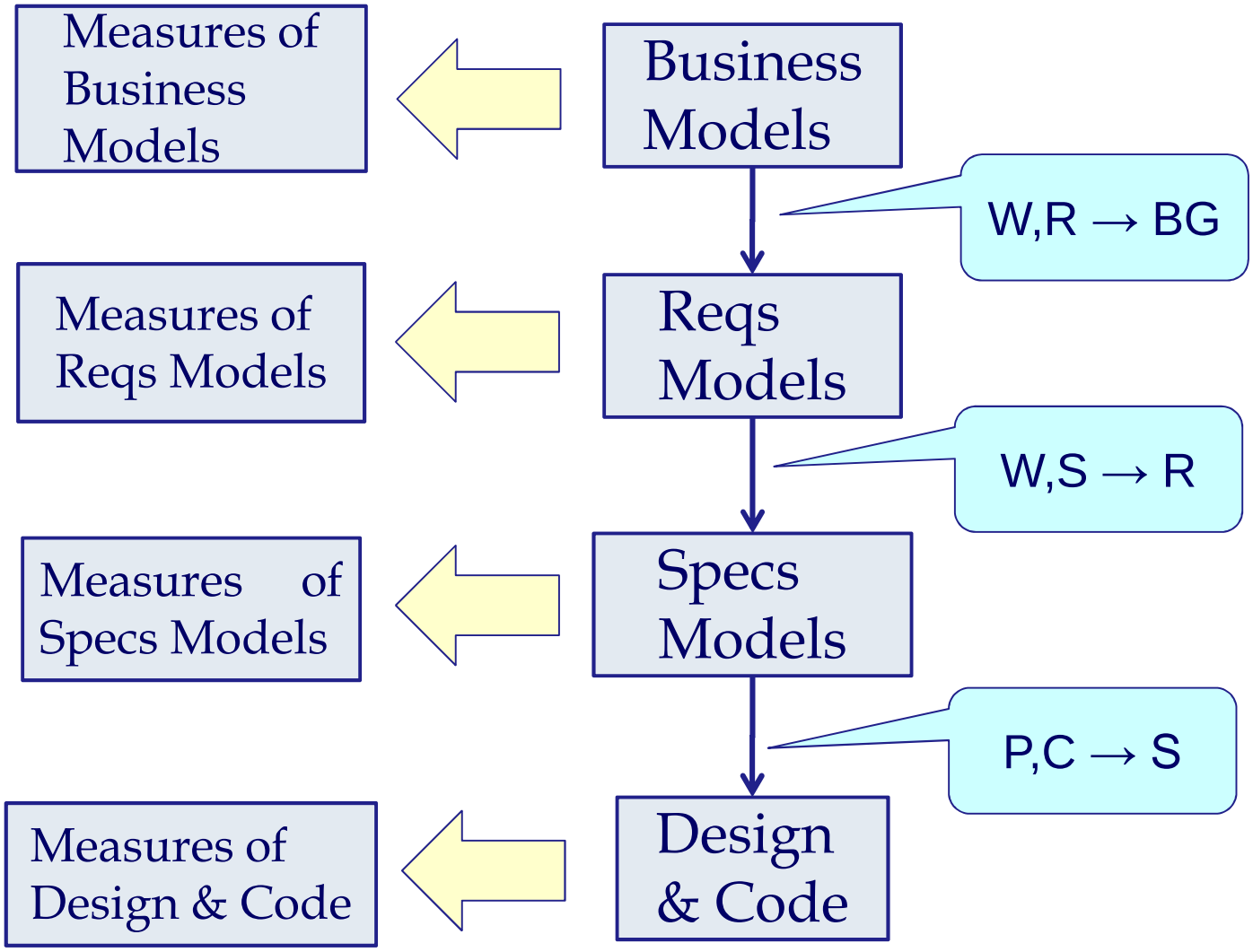
Promoting the use of models



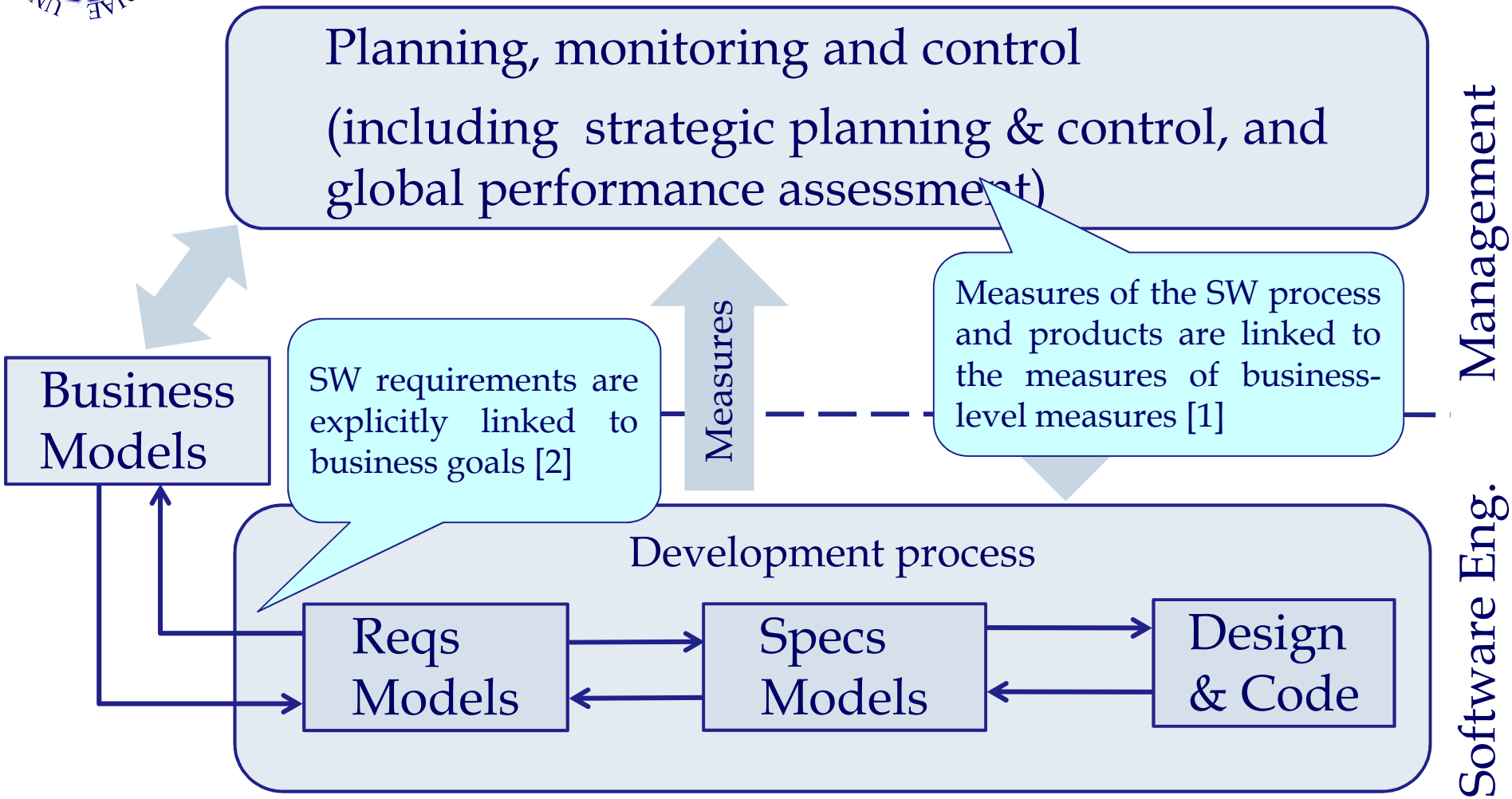
What about the upper management?

- Software is typically functional to some other type of business
- How can we relate the management of software to the management of the “core” business of a company?

Requirements and artifacts hierarchy



Further promoting the use of models



[1] Vic Basili et al. "Linking Software Development and Business Strategy through Measurement", IEEE Computer, April 2010

[2] L. Lavazza "User needs vs. user requirements: a Problem Frame-based View", IWAAPO 2010

Conclusion

- We can define SE practices that are not functional just to software development (i.e., to technical tasks) but also to management.
- The management and SW development process are deeply interwined.
- Software requirements, specifications and artifacts are linked to business rules, constrains and goals.
- A uniform set of models can help defining, understanding and quantitative-based managing the whole process.

- The good news: the needed techniques exist. We just have to use them!