3rd Workshop on Architectures for Dependable Systems (WADS) – 26th ICSE May 2004 - Edinburgh

ARCHITECTURAL CHOICES FOR DEPENDABLE SYSTEMS

Nicole Levy
Laboratoire PRISM
Université de Versailles St.-Quentin
Nicole.Levy@prism.uvsq.fr

Francisca Losavio
LaTecS Laboratory, Centro ISYS
Universidad Central de Venezuela
flosav@cantv.net

AGENDA

- Goal
- Motivation
- Terminology
- Requirements classification model
- Pattern-based architectural design
- Application
- Conclusion

GOAL

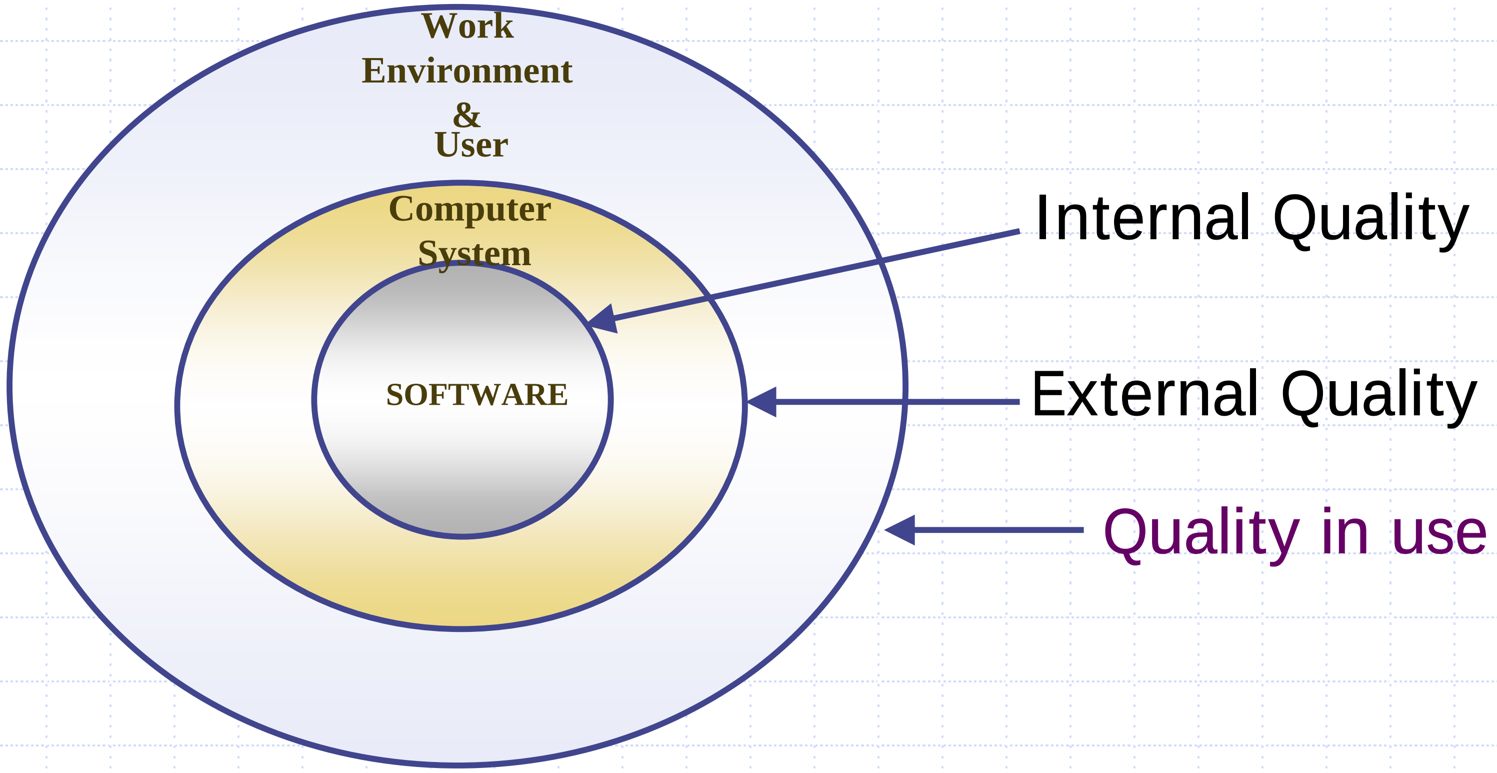
- Present an architectural design based on
  - Identification of the problem’s domain
    - Dependable systems in a wireless context
  - Definition of the problem
    - functional requirements
    - nonfunctional requirements
  - Selection of patterns (architectural solutions) from a pattern library, according to quality properties

MOTIVATION

- Functional requirements must be implemented
- Nonfunctional requirements are related to the problem's context and to the system's execution or operational environment
    - affect the implementation of the functional req.
    - originate implicit functionalities
  - **Ex. Transient connections in the context of mobile ad hoc networks**
  - **Ex. Existence of legacy systems**

TERMINOLOGY

Quality Views

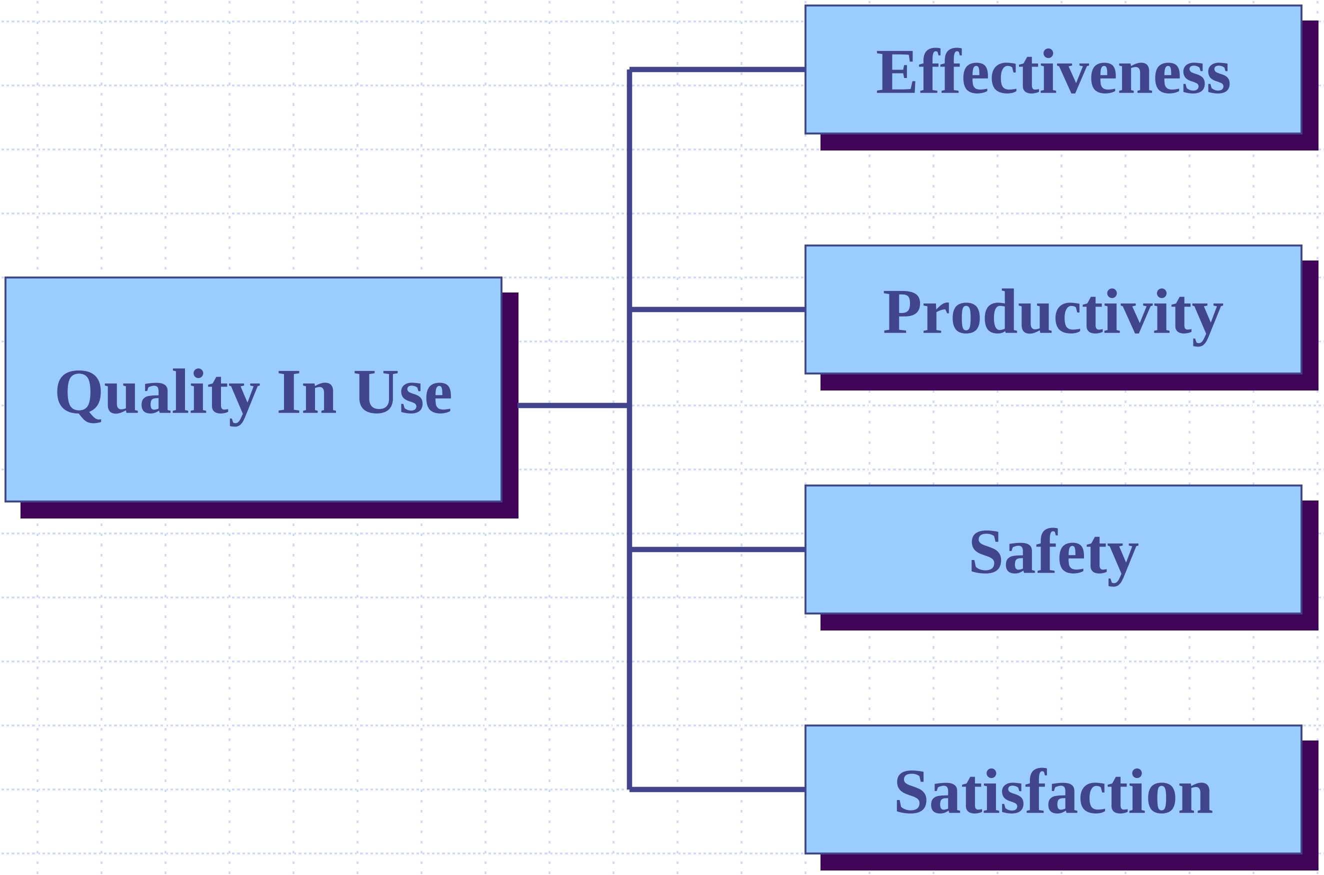


ISO/IEC 9126-1 [2001] - Quality Model
External & Internal Characteristics

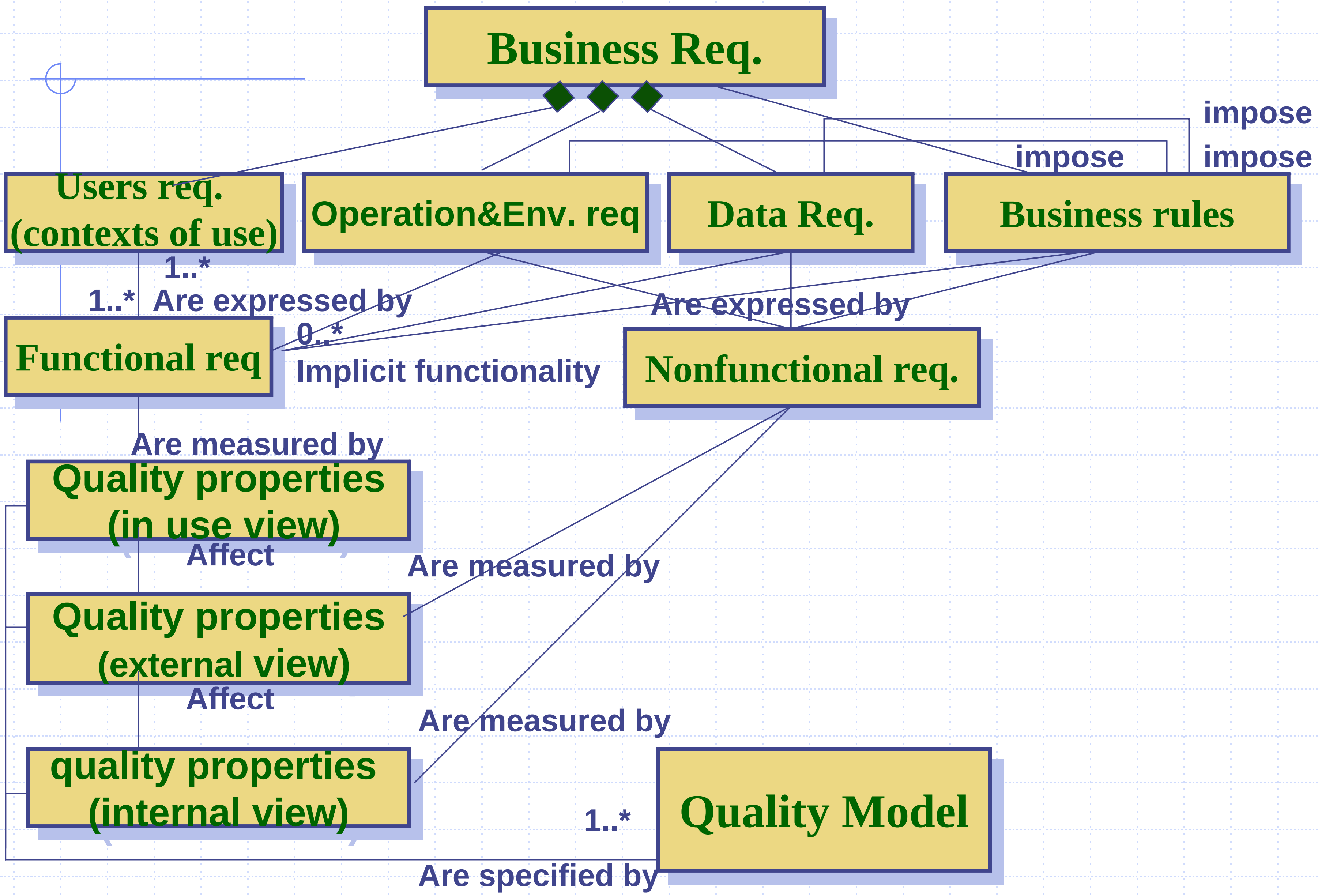
Quality Characteristics — **Subcharacteristics**

- **Functionality**
  - Suitability
  - Accuracy
  - Interoperability
  - Security
  - Compliance
- **Reliability**
  - Maturity
  - Fault tolerance
  - Recoverability
  - Compliance
- **Usability**
  - Understandability
  - Learnability
  - Operability
  - Comp
  - Attractiveness
- **Efficiency**
  - Time behavior
  - Resource utilization
  - Compliance
- **Maintainability**
  - Analyzability
  - Changeability
  - Stability
  - Testability
  - Compliance
- **Portability**
  - Adaptability
  - Installability
  - Co-existence
  - Replaceability
  - Comp

ISO/IEC 9126-1 - Quality Model
Quality In Use Characteristics



REquirements CLassification MOdel



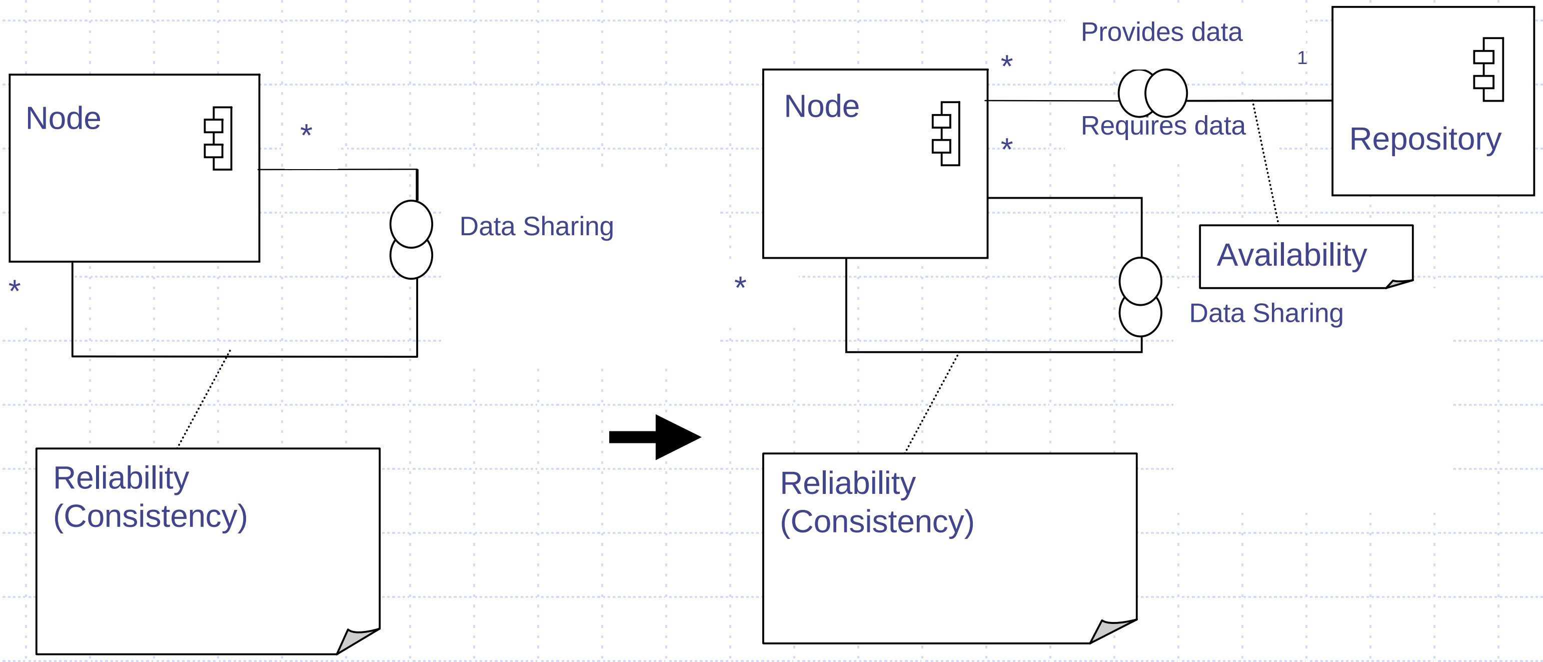
Pattern-based Architectural design – proposition

- We aim at providing helps to guide the application of architectural patterns
- Patterns are
  - defined as <problem-solution> couples
  - specified in UML 2.0
  - described in a pattern library
- Both problem and solution include functional and nonfunctional requirements.
- The architectural decisions are driven by quality goals derived from the nonfunctional requirements

Pattern-based Architectural design – proposition

- We focus on the problem part of the patterns:
  - The functionality of the problem is added.
  - Nonfunctional requirements are added
  - A *Quality* clause is added, expressed as decorations (by UML tags)

Architectural Pattern: Repository



Architectural Design Process Model

SPEM (Soft. Process Eng. Metamodel Spec.), OMG 2001

Our Architectural design

Software Architect

- Identification of business requirements
- Identification of nonfunctional requirements
- Identification of quality characterstics for each functionality
- First definition of the architecture
- Architecture refinement. This activity is applied iteratively for each quality characteristics, until all of them have been considered

Cahier des charges

Relation of quality rerequirements wwith functionalities

Quality Model for problem domain

Use case Model

Sequence a StateChart Diagrams

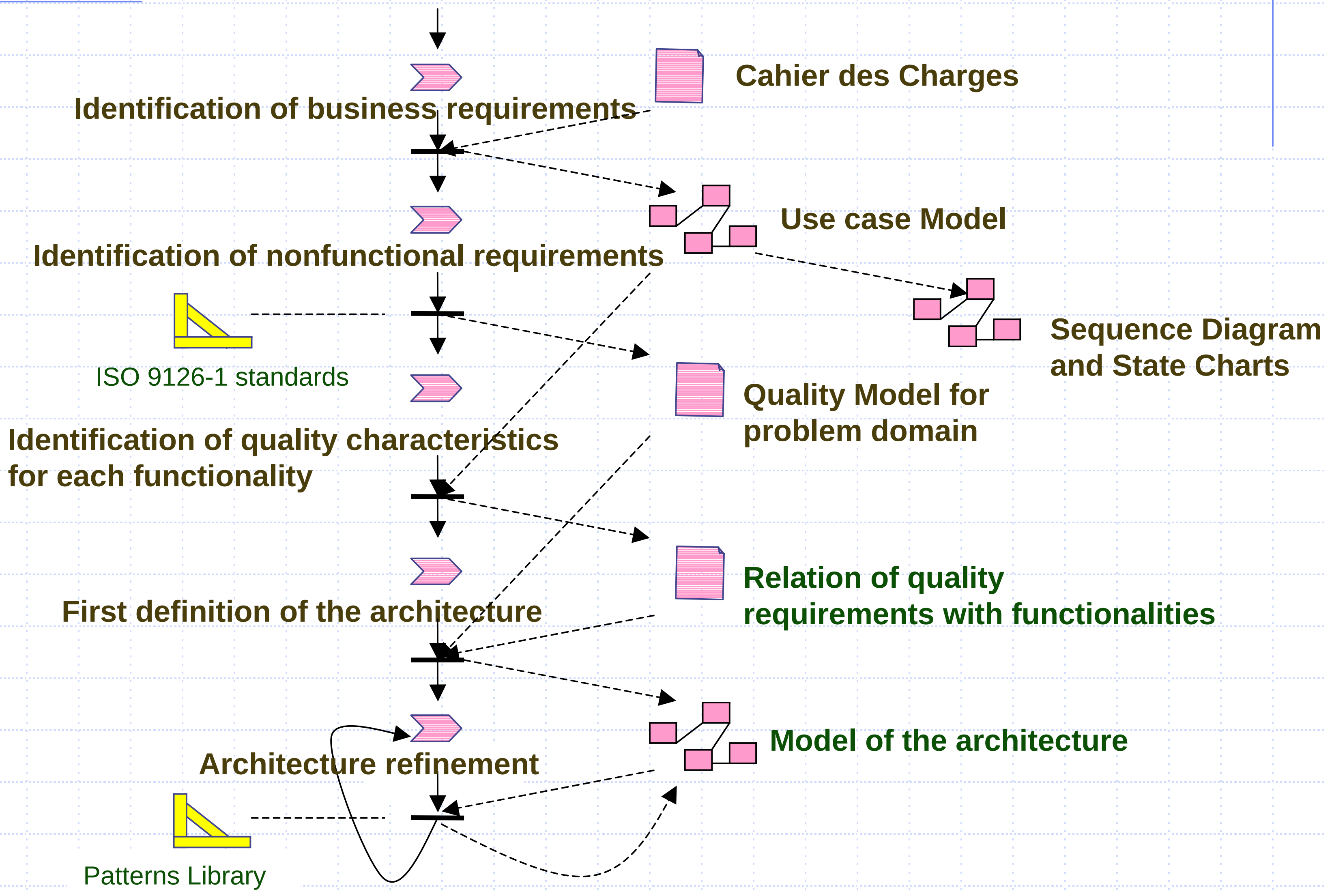
Model of the architecture

ISO 9126-1 standards

Patterns Library

Activity Duagram of the Architect

SPEM (Soft. Process Eng. Metamodel Spec.), OMG 2001


Cahier des Charges
Identification of business requirements
Use case Model
Identification of nonfunctional requirements
Sequence Diagram and State Charts
ISO 9126-1 standards
Quality Model for problem domain
Identification of quality characteristics for each functionality
Relation of quality requirements with functionalities
First definition of the architecture
Model of the architecture
Architecture refinement
Patterns Library

Pattern-based Architectural design – process application

1. **Express the problem and its context**

   1.1 Problem definition

Accomplish collaborative work in a mobile ad hoc network context.

- A group member is defined as a mobile device or entity (node).
- A member may leave a group because he failed, is explicitly requested to leave or is expelled by other members. Similarly, a member may join a group because he explicitly requests it or recovers from failure.
- Failure can be caused by the member's resource scarcity (battery, memory, etc.) or by disconnection (connection fails or he is more within the group connection range).
- Group membership is constrained by the relative location of the member nodes or group connection range, limited to 1 or 2 hops for ad hoc models.
- Members must consistently share data within the group connection range and must also have a consistent view of the group, despite network failures.
- Group membership may be restricted to authorized member nodes (security domain).
- The minimization of resource consumption, in particular energy, on mobile member nodes is mandatory since there is no infrastructure, requiring minimization of the number of messages exchanged among group members to guarantee the overall ad hoc network reliability.

Pattern-based Architectural design – process application

1.2 Functional requirements

- Message exchange among group members (user's req.)
- Data sharing among group members (user's req.)
- Group Membership Management (environment req.)
  - Discovering group members
  - Initializing the group
  - Updating the group membership

1.3 Nonfunctional requirements

- Handling transient connections (environment req.). It implies:
  - Decentralized solution, where responsibility is distributed among nodes (technology business rule)
  - Managing data consistency (environment req.)
- Restricted membership (policy business rule)
- Minimization of resource consumption (in particular energy) on mobile entities is mandatory (policy business rule)

Pattern-based Architectural design – process application

Nonfunctional requirements (Quality model)	*Quality characteristics*			
	Reliability	*Security*	*Efficiency*	*Maintainability*
Minimization of resource consumption			-Performance with respect to resource utilization: battery, memory, CPU load, bandwidth -Attribute: measure of the resource consumption for each device -Metrics: percentage [0..1]	

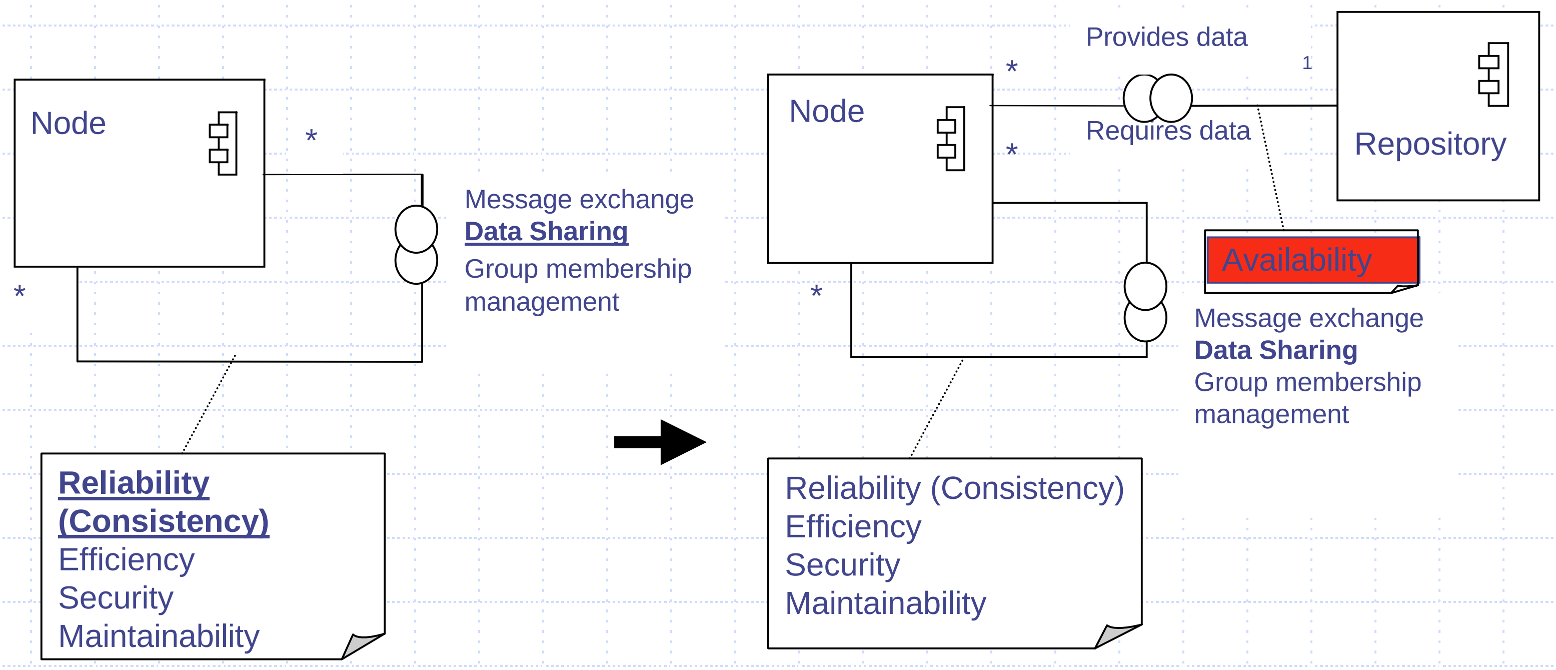
Pattern-based Architectural design – process application

Nonfunctional requirements (Quality model)	*Quality characteristics*			
	Reliability	*Security*	*Efficiency*	*Maintainability*
Management of Data consistency … … …	-*consistency:* a mechanism must be provided, for example to manage the update of replicated data on each member node -Attribute: presence of mechanism -Metrics: boolean			

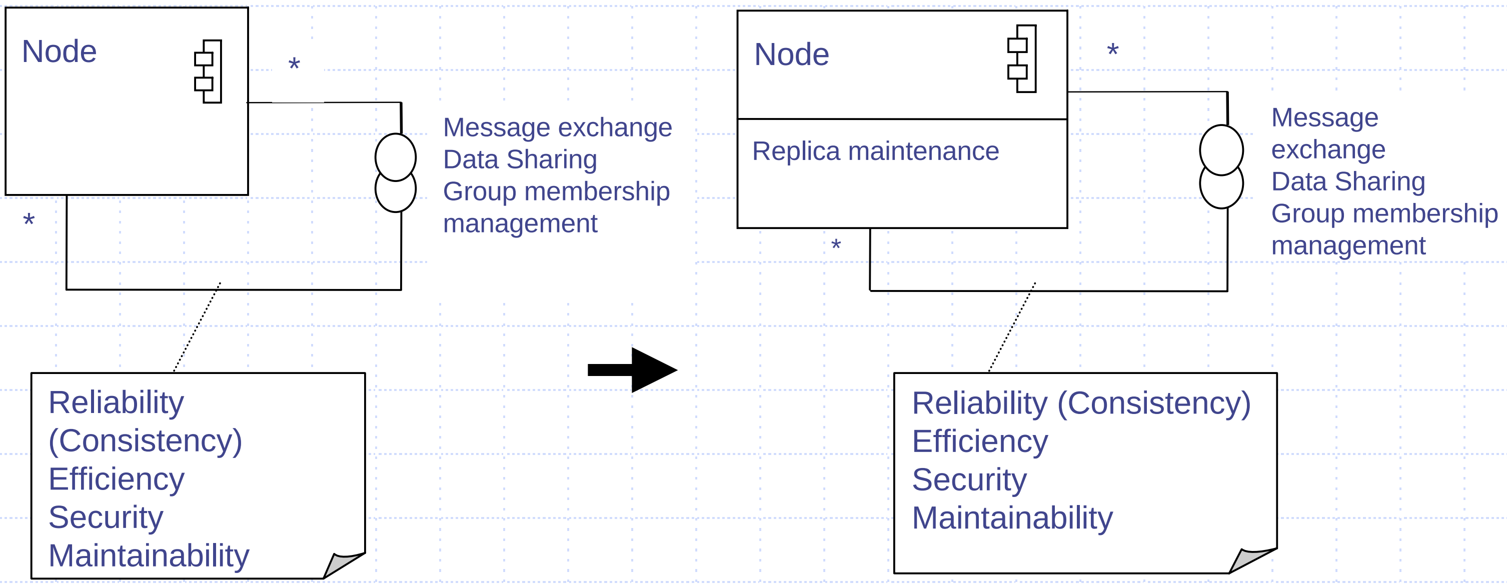
Pattern-based Architectural design – process application

Functional requirements	*Quality Characteristics*			
	Reliability	*Secur ity*	*Efficiency*	*Maintainability*
Group membership	Reliability (availability)	Secur ity	Performance with respect to resource utilization: battery, memory, CPU load, bandwidth. - Attribute: resource consumption for each device - Metrics: percentage [0..1]	Changeability
Message exchange among group members	Reliability (availability)		Efficiency (performance with respect to Resource Utilization … …)	
Data sharing among group members	Reliability (consistency)			

Pattern-based Architectural design – process application



Pattern-based Architectural design – process application



CONCLUSION

- The precise identification of software requirements is mandatory for architectural design
- A standard quality model is used to specify the quality requirements
- The requirements engineering classical process is extended with the explicit analysis of the quality requirements
- Requirements engineering for critical systems is still a major challenge:

→ Introduce quality goals at the same time as the functional requirements

CONCLUSION

- Architectural choices and their documentation are essential to ensure quality of critical systems
- Software engineering practices (repeatable process) are still needed for a sound software requirements engineering.
- Architectural design needs quality requirements engineering

→Integrated case tool is needed with support to

→Requirements engineering

→Architectural design

→Quality control and evaluation

Thank you!!! – Questions?

MOTIVATION

- Software systems must accomplish
  - functional requirements, services offered
- Software systems are characterized by
  - nonfunctional requirements, constraints that will be expressed by quantifiable properties, on the implementation and the execution of the services.
  - all the requirements are elicitated or captured from the “cahier des charges”

MOTIVATION

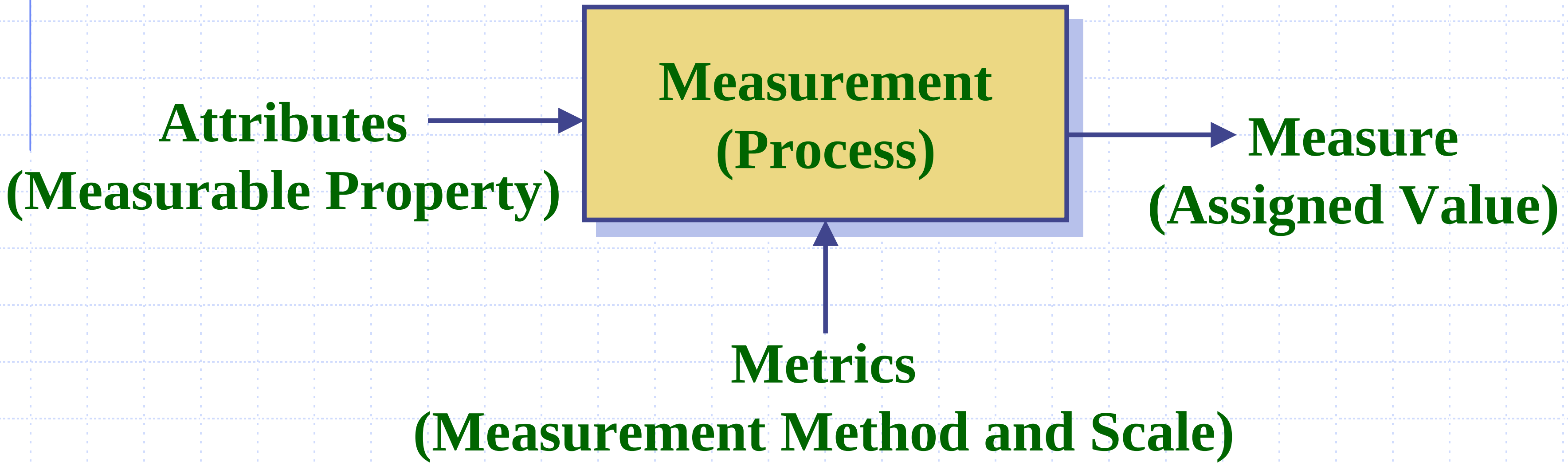
- Historically, efforts have been concentrated more on the modeling of functional requirements.
- Nonfunctional requirements have been poorly considered, in particular, those related to the software product quality

Consequence:

poor nonfunctional requirements specification

- Delivery of applications that possibly do not comply with all stakeholders expectations, increasing project risks
- Development of critical applications, where the architecture plays a central role, lacks sound repeatable processes

Metrics Concept in ISO 9126 and ISO 14598



Pattern-based Architectural design – proposition

- We aim at providing helps to guide the application of architectural patterns
- We add to the actual descriptions, the definition of the problem part with both functional and nonfunctional requirements
- The pattern structure usually contains several clauses concerning both the *problem part* and the *solution part*.

Pattern-based Architectural design – pattern library

The problem is usually described within several clauses:

- The ***Intent*** clause as rationale contains the design issues addressed. A scenario given in the ***Motivation*** clause may give more specific information about the design problem. **The functionality of the problem** is added.
- The ***Participants*** are Classes or object structures already existing that can be used as parameters of the pattern; they are partially described or defined in the ***Structure*** clause. **UML 2.0 models** are used here.
- List of conditions described as situations in which the design pattern can be applied, the poor designs that the pattern can address. These are mainly described in the ***Applicability*** clause.
- In the ***Context*** clause, **the nonfunctional requirements** are added
- The new ***Quality* clause** is added, expressed as decorations (by **UML tags**) containing the quality model associated to the problem domain. The quality model follows the ISO 9126-1 standard definition [ISO/IEC 2001].