PV260 COURSE INTRODUCTION

ROADMAP TO SOFTWARE QUALITY

Barbora Bühnová
buhnova@fi.muni.cz

LAB OF SOFTWARE ARCHITECTURES
AND INFORMATION SYSTEMS

FACULTY OF INFORMATICS
MASARYK UNIVERSITY, BRNO

Outline of the lecture

- Course introduction
  - Course motivation and goals
  - Course organization
  - Our team
- Roadmap to quality assurance methods
  - Define quality issues
  - Prevent quality issues
  - Detect quality issues
  - Repair quality issues
  - Keep track of quality issues
- Choose well, plan well

lasaris

Outline of the lecture

- Course introduction
  - Course motivation and goals
  - Course organization
  - Our team
- Roadmap to quality assurance methods
  - Define quality issues
  - Prevent quality issues
  - Detect quality issues
  - Repair quality issues
  - Keep track of quality issues
- Choose well, plan well

lasaris

Course motivation and goals

"People forget **how fast** you did a job – but they remember **how well** you did it" – some guy named Howard Newton

- The **aim of the course** is to help the students to
  - understand activities contributing to building **high-quality software**;
  - develop **critical thinking** and be able to identify **code flaws** related to reliability, performance, scalability, maintainability and testability;
  - be able to **refactor existing code** to improve different quality attributes;
  - have practical experience with software **testing** and related **tools**.

Outline of lectures

Lect 1. Course organization. Roadmap to **software quality engineering methods**.

Lect 2. Requirements and test cases. From unit testing to integration testing.

Lect 3. Quality and **testing in agile**. Practical **insights on QA** in real product development.

Lect 4. Clean Code & **SOLID principles**. Bad code smells and code **refactoring**.

Lect 5. Performance engineering and performance testing.

Lect 6. Automated testing and testability. Continuous integration.

Lect 7. Software **measurement and metrics**, and their role in quality improvement.

Lect 8. The role of **software architecture**.

Lect 9. Focus on **quality attributes and conflicts** between them.

Lect 10. Static code analysis and code reviews.

Lect 11. Challenges of quality management in **cloud applications**.

Lect 12. Software quality **management process**. Course summary.

Colloquium event

Course organization

- Lectures
  - Shared by us and **experts** from companies
  - May **not be recorded**
  - Final **colloquium event** after the end of semester
- Seminars
  - Practical assignments on computers
  - Teamwork, homework, projects
  - **2 Java groups** – taught by **LaSArIS lab members**
  - **1 C# group** – taught by **YSoft experts**

Course organization

- Evaluation
  - **45** points for seminar **assignments**
  - **10 seminar activity** points
  - **10 lecture activity** points
  - **35** points for **final colloquium assessment**, consisting of
    - obligatory **attendance** at the final colloquium event and
    - final **written test**
  - Minimum of **70 points for passing the course**

- Colloquium event
  - On **June 4, between 9-14h**
  - **Discussion groups** led by industrial experts
  - **Student presentations** of outcomes
  - **Written test** (at the end of the day, or on a separate term)

Our team

lasaris

- Barbora Bühnová
- Bruno Rossi
- David Ešvindr
- Stanislav Chren

YSOFT

- Ondřej Krajíček
- Martin Osovský
- Radim Göth
- and others

Honeywell

- Jaromír Skřivan
- Lukáš Pitoňák
- Jakub Papcun
- Jan Svoboda

solarwinds

- Jiří Pokorný

SIEMENS

- Jan Verner

PerfCake

- Pavel Macík
- Martin Večeřa

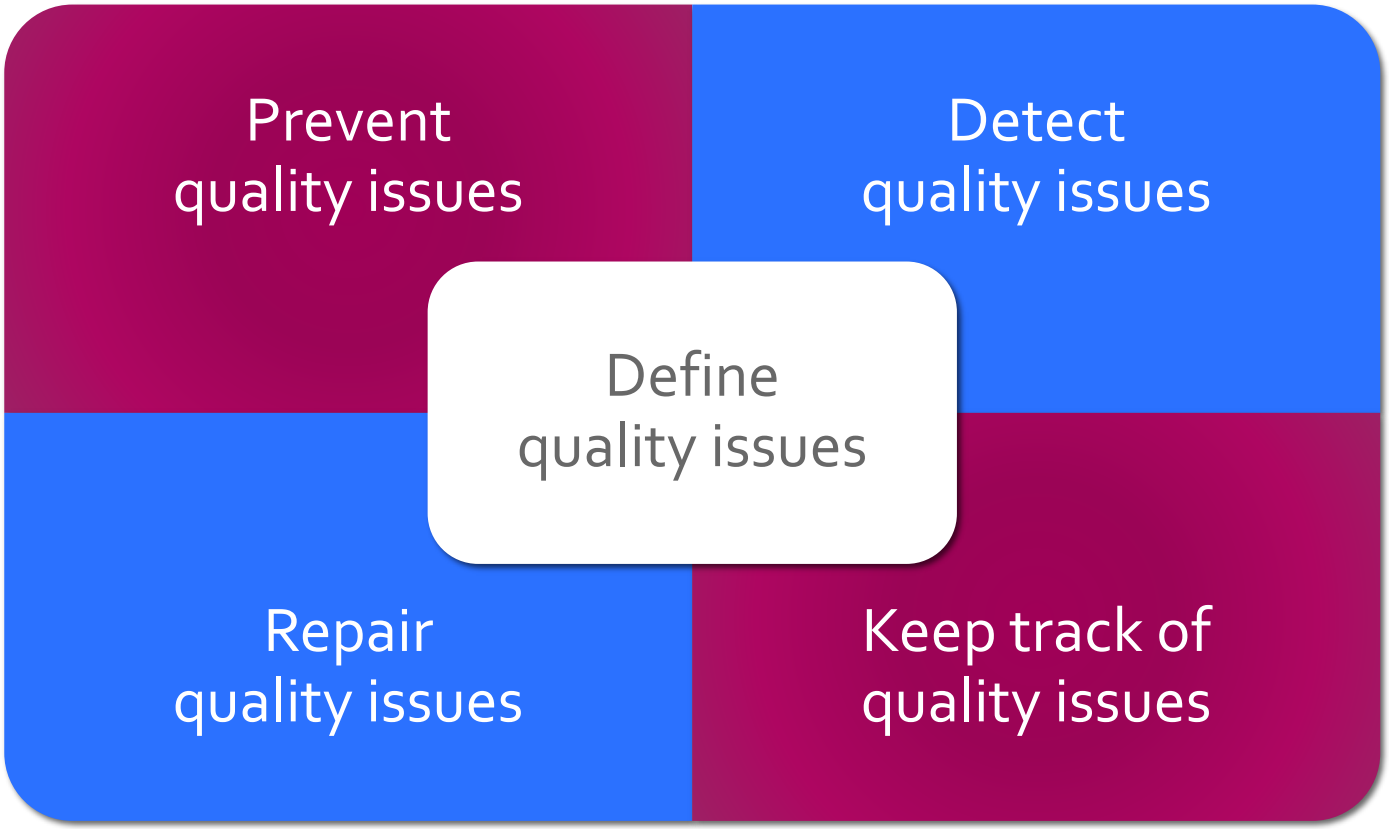
Outline of the lecture

- Course introduction
  - Course motivation and goals
  - Course organization
  - Our team
- Roadmap to quality assurance methods
  - Define quality issues
  - Prevent quality issues
  - Detect quality issues
  - Repair quality issues
  - Keep track of quality issues
- Choose well, plan well

lasaris

Quality Assurance (QA) methods

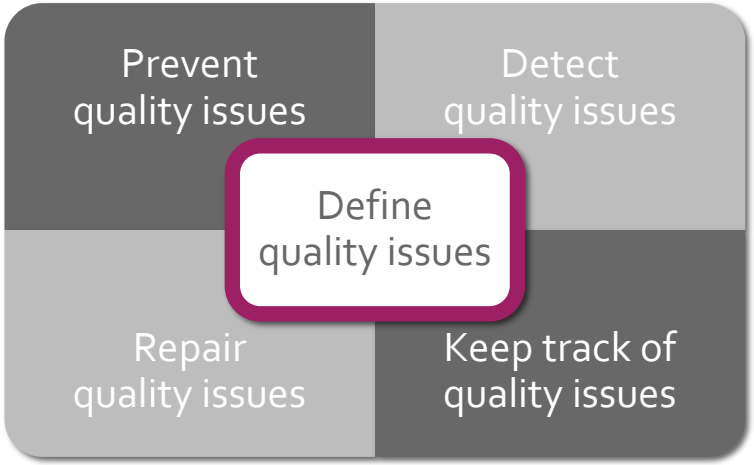
Roadmap to QA methods



Define quality issues

- Software quality is commonly defined as the capability of a software product to conform to requirements [ISO/IEC 9001].

customer needs



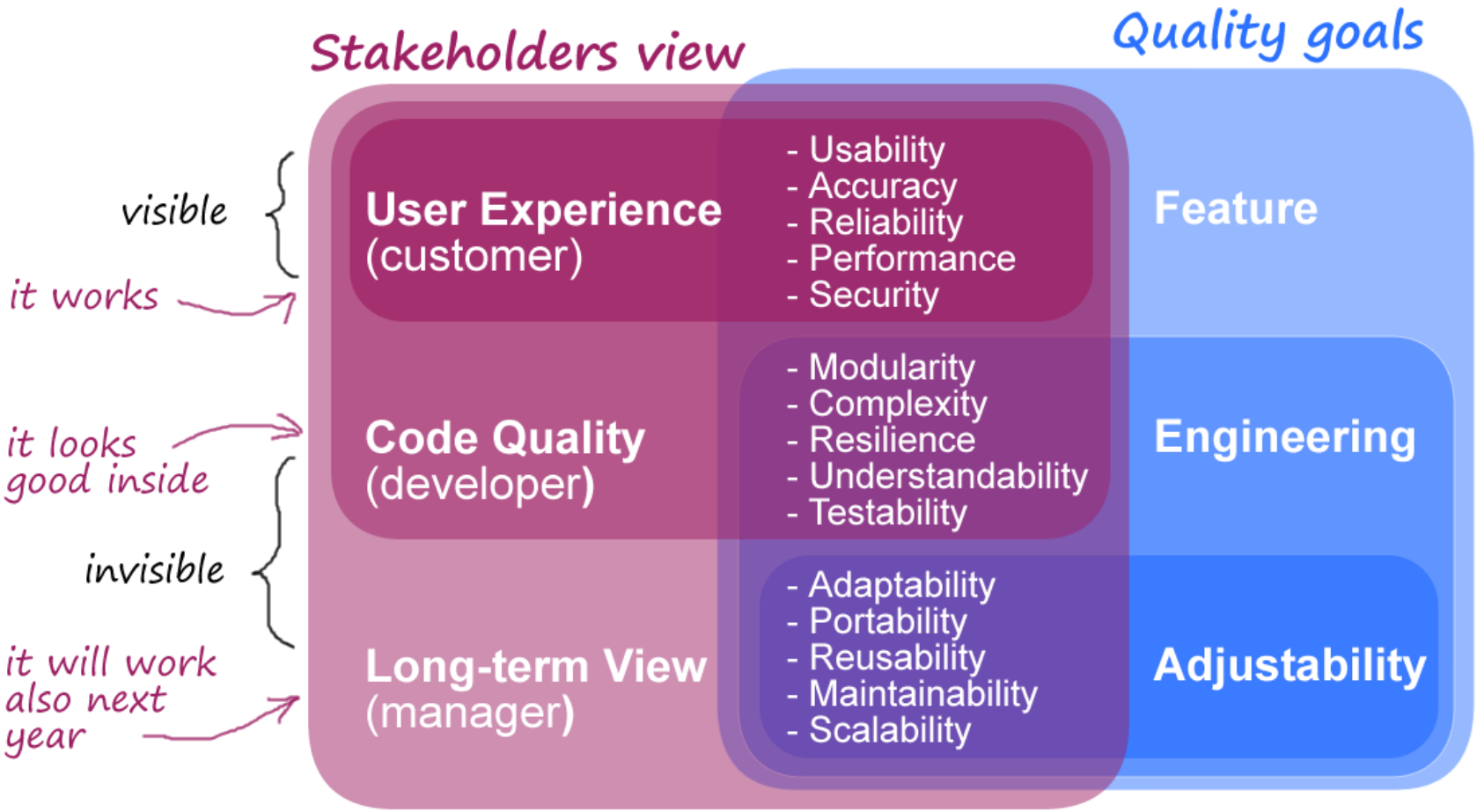
- Requirements engineering
- Software metrics
  - 'You cannot manage what you cannot measure'
- Quality attributes
  - Of a **product**, process and resources

lasaris

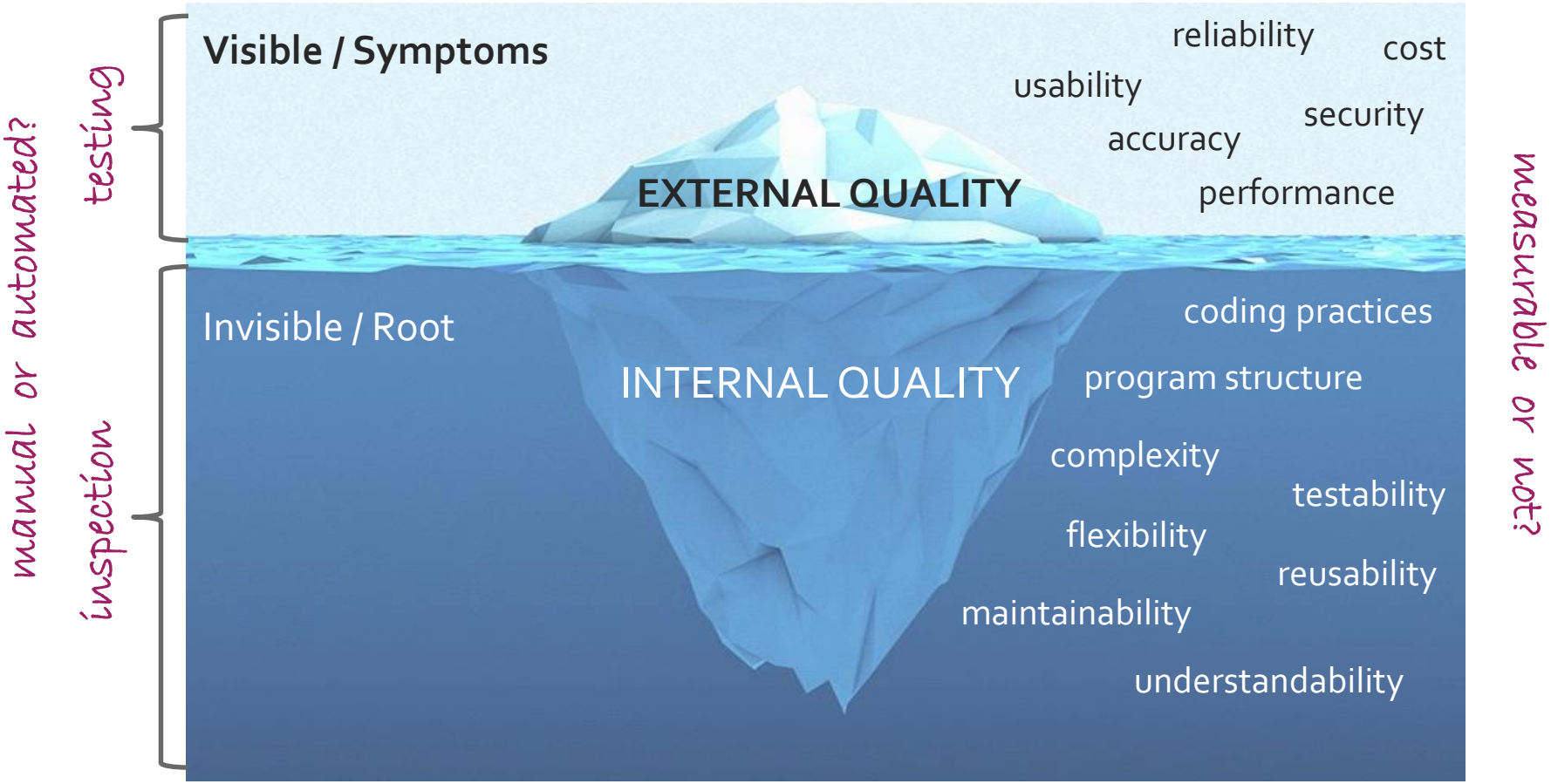
… and your customer?

What "quality" means to you?

… and your manager?



The Software Quality Iceberg



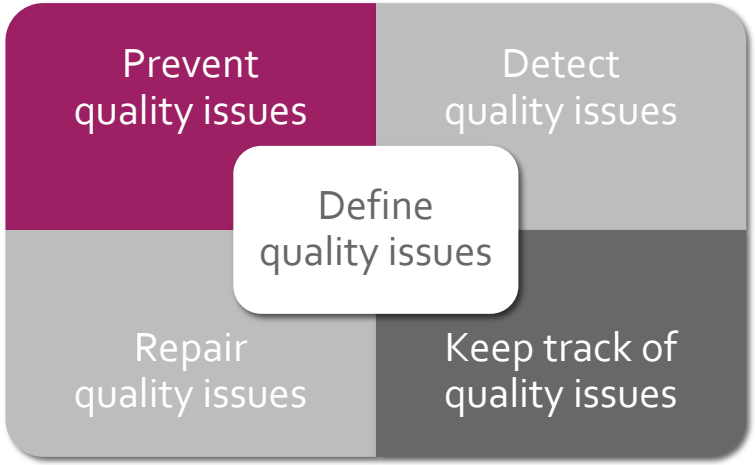
Inspiration from [5]

The big five

- Along the course we will focus on:
  - **Maintainability** – **ease of change** (without increased technical debt)
  - **Performance** – **response time** and efficiency in resource utilization
  - **Reliability** – probability of **failure-free operation** over a period of time
  - **Testability** – degree to which the system **facilitates testing**
  - **Scalability** – system's ability to handle **growing work load**
- Quality attributes studied in related courses:
  - **Security** – system's ability to **protect itself** from attacks
  - **Usability** – ease of system **use and learnability**

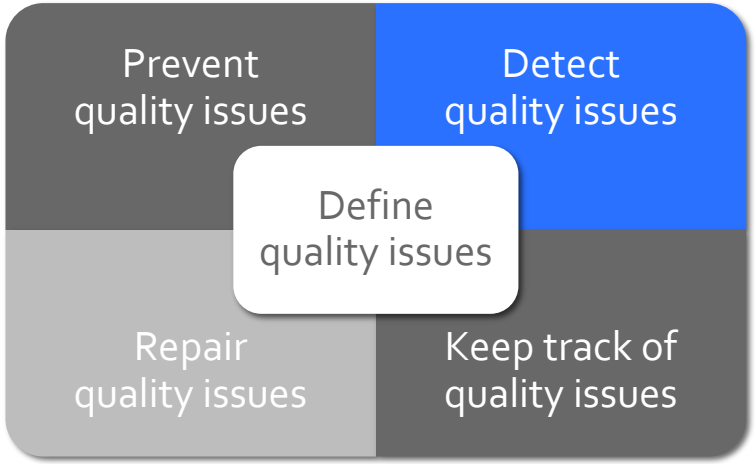
Prevent quality issues

- Coding best practices
  - Clean code, SOLID principles
  - Design patterns
  - Pair programming
- Code conventions
  - Language specif. recommendations
- Quality assurance processes
  - V-model of testing
- Standards for development process improvement
  - CMMI and ITIL reference models
  - ISO 9000, ISO/IEC 25010

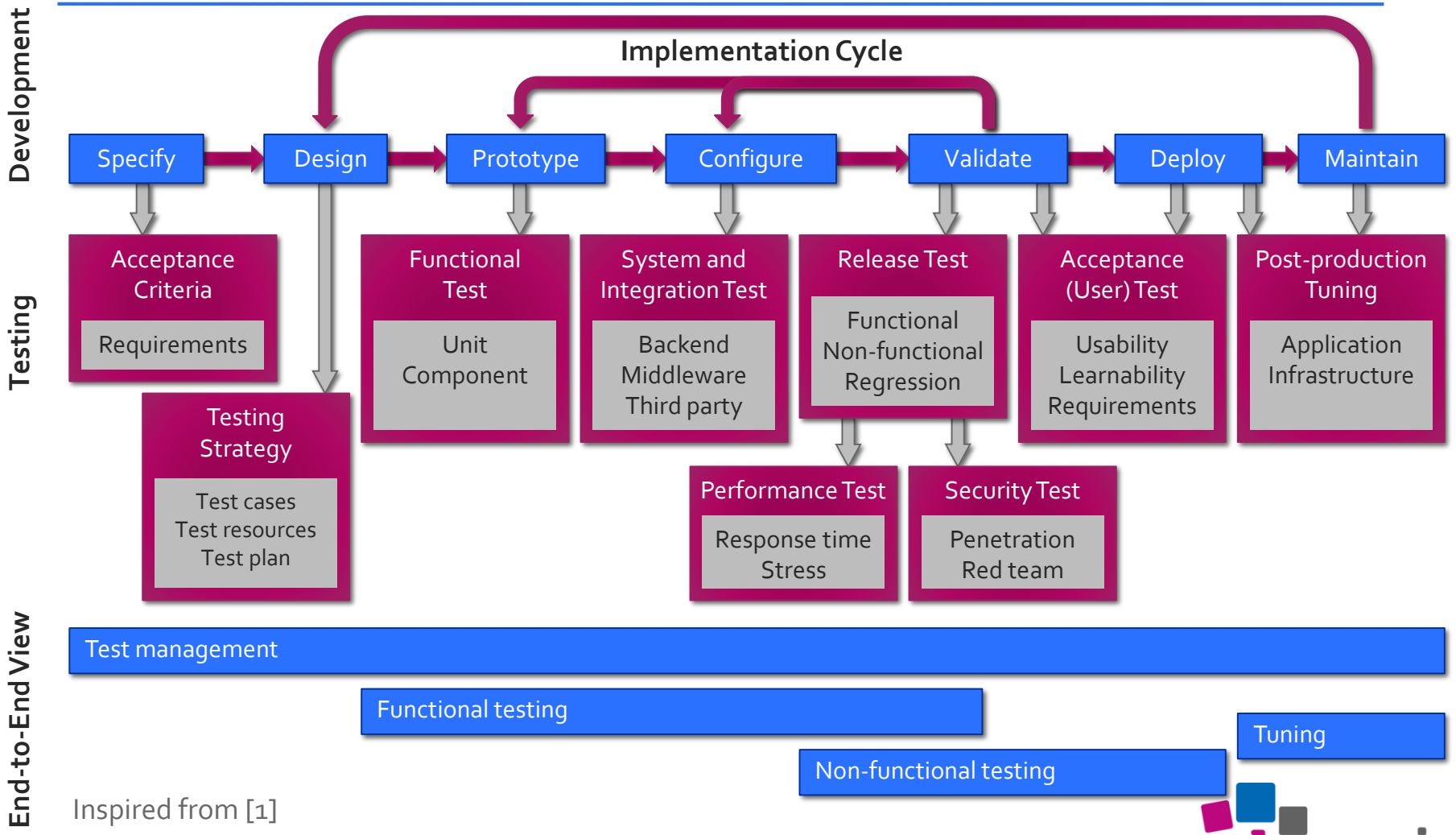


Detect quality issues

- Testing functional requirements
  - Manual or automated
- Testing non-functional req.
  - Performance, usability, security testing
- Design inspections
  - Manual inspections of design artifacts
- Code reviews
  - Manual inspections of code
- Automated static code analysis



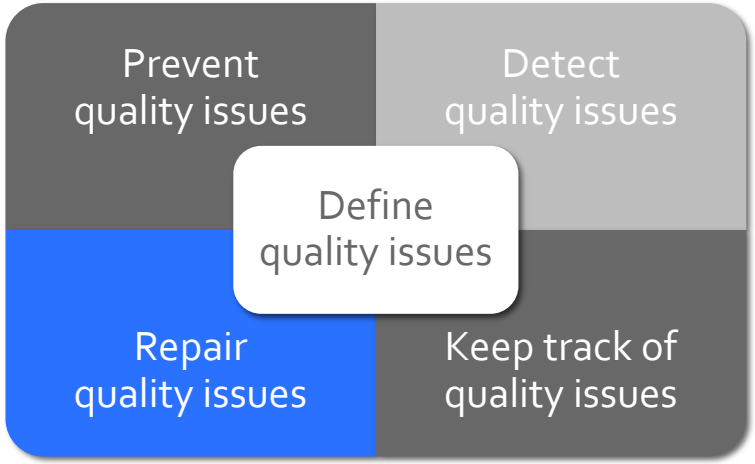
Roadmap to software testing



Inspired from [1]

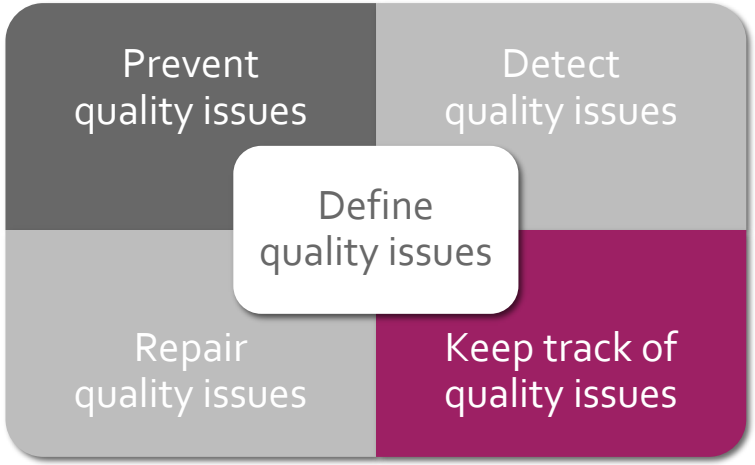
Repair quality issues

- Functional issue
  - Code repair
- Reliability issue
  - Fault tolerance mechanisms
- Performance issue
  - Concurrency, effective resource utilization, identify and remove system bottlenecks
- Security issue
  - Identify and remove system vulnerabilities (single points of failure)
- Maintainability issue
  - Refactoring to clean code principles, to design patterns

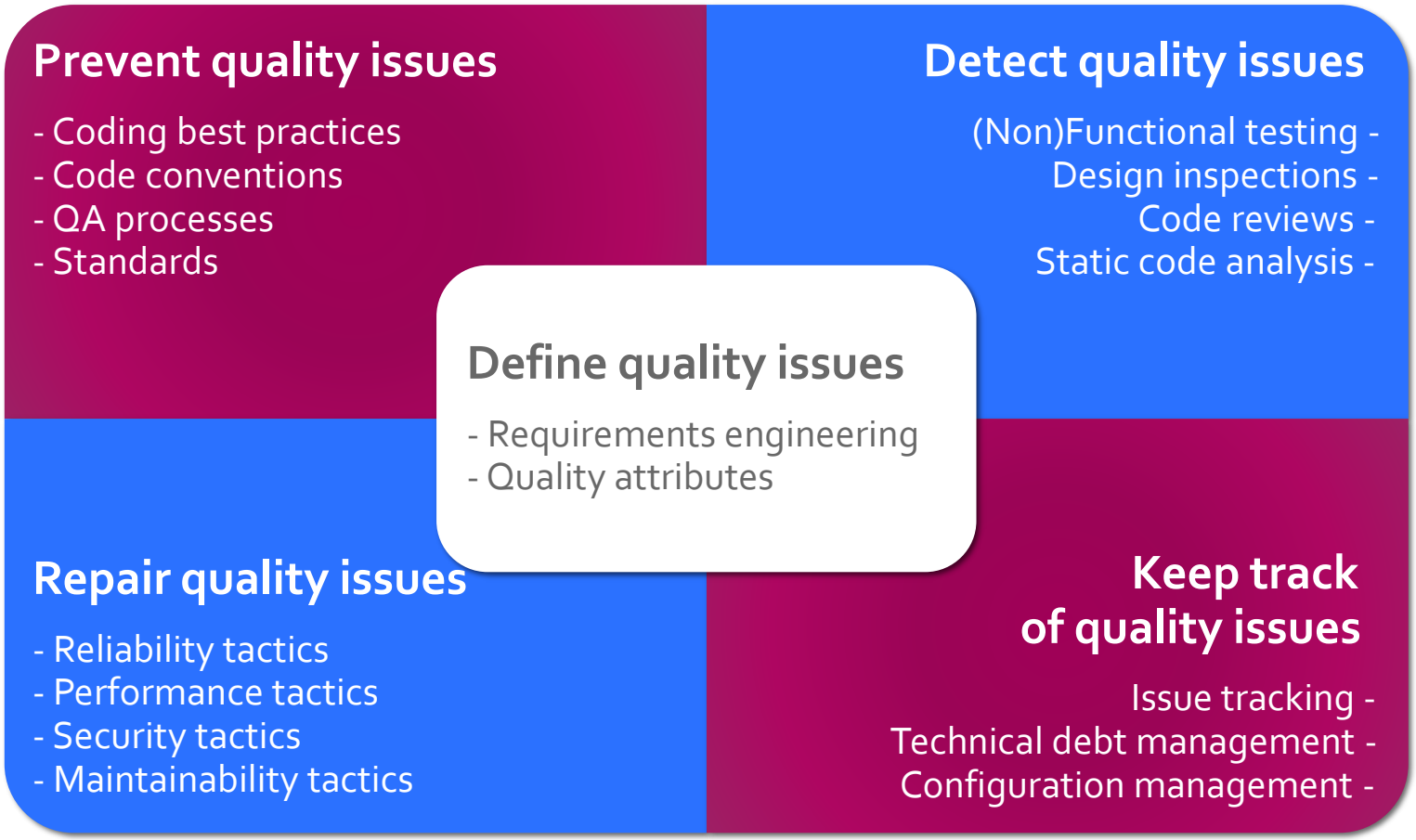


Keep track of quality issues

- Issue tracking
  - Supports the management of issues reported by customers
- Technical debt management
  - Level of code quality degradation
  - Work that needs to be done before a particular job can be considered complete or proper
- Configuration management
  - Version management and release management
  - System integration



Roadmap to QA methods – the Big Picture



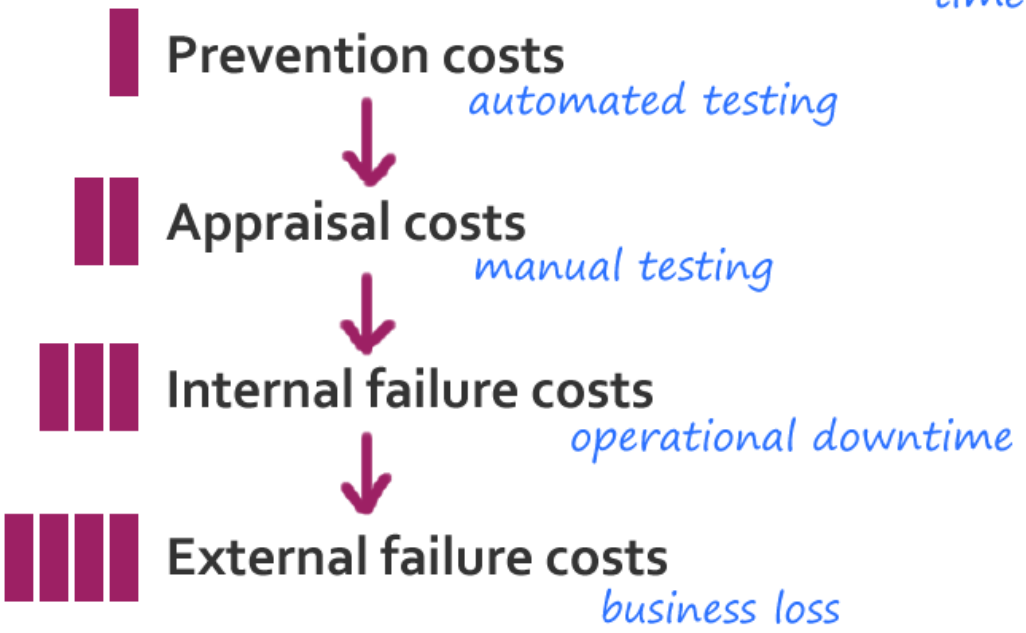
Outline of the lecture

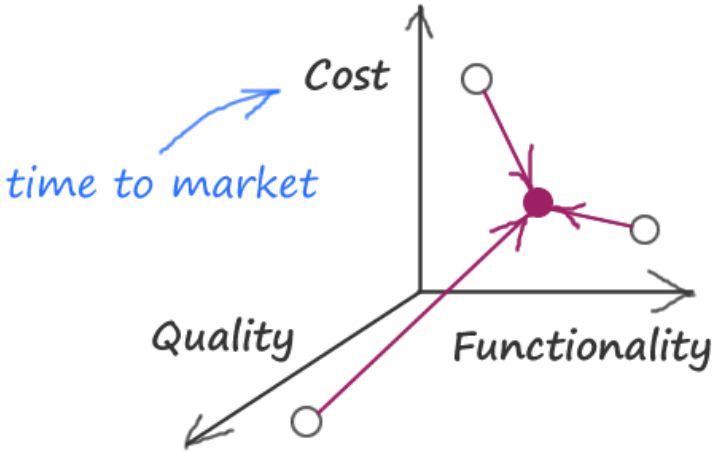
- Course introduction
  - Course motivation and goals
  - Course organization
  - Our team
- Roadmap to quality assurance methods
  - Define quality issues
  - Prevent quality issues
  - Detect quality issues
  - Repair quality issues
  - Keep track of quality issues
- Choose well, plan well

lasaris

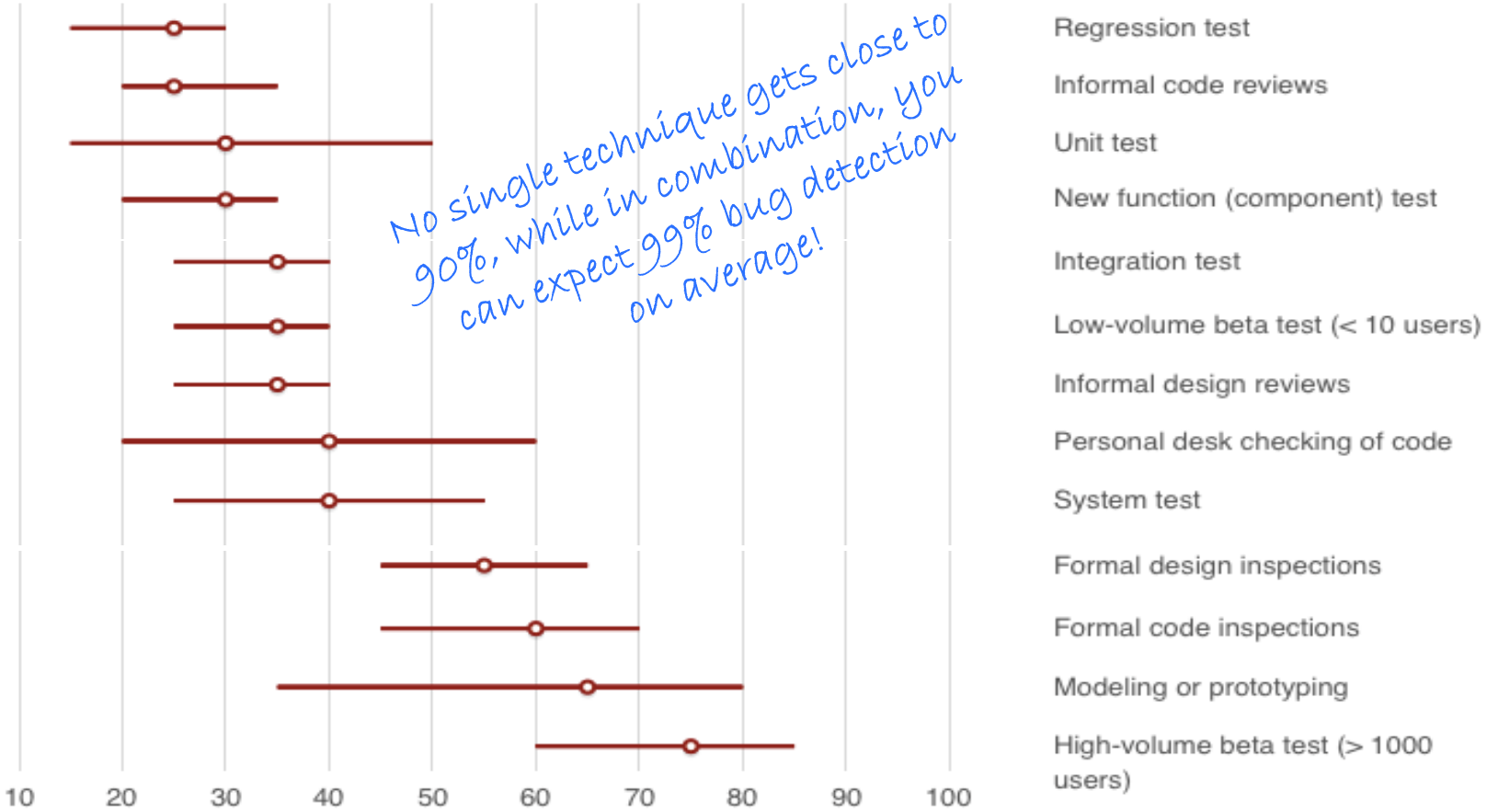
Choose well, plan well

- Think well about your **requirements** and the **cost of the quality**





Choose well – Combination is the key



From [2,3], see also RebelLabs reports [4]

Plan well – The Power of Analogy

- **Airplane Servicing**
  - Requires **regular servicing** e.g. every 100,000 miles.
  - Takes place even if everything seems to work all right, because we cannot afford a failure.
- **Technical Debt Management**

  - Introduced by Ward Cunningham
  - Analogy of **quality degradation with financial debt** – if not paid off, interests increase. One can get into trouble.
- **Sometimes it is wise to "borrow money"**
  - When one expects to have more money in the future (start-up company)
  - When one needs to act fast not to miss a market opportunity
  - When one expects money devaluation (e.g. developers will become more experienced, it will be easier to understand user needs)

Takeaways

- **Quality assurance (QA)** is much more than **testing**, including many different methods to
  - **prevent**, **detect**, **repair** and **keep track** of quality issues
- **Combination of the methods is the key** to successful QA
  - But **choose well** and **plan well**, not all methods are best for your project!
- Make sure you understand the **needs of your customer**
  - Balance **both internal and external quality attributes** for both the present and the future

thanks for listening

Barbora Bühnová, FI MU Brno
buhnova@fi.muni.cz
www.fi.muni.cz/~buhnova

contact me

lasaris

References

- [1] Testing You Perform When You Develop a Siebel Application. Available online at http://docs.oracle.com/cd/E14004_01/books/DevDep/Overview5.html
- [2] Steve McConnell. Code Complete: A Practical Handbook of Software Construction, Second Edition. Microsoft Press, June 2004.
- [3] Kevin Burke. Why code review beats testing: evidence from decades of programming research. Available online athttps://kev.inburke.com/kevin/the-best-ways-to-find-bugs-in-your-code/
- [4] RebelLabs. 2013 Developer Productivity Report. Available online at http://zeroturnaround.com/rebellabs/developer-productivity-report-2013-how-engineering-tools-practices-impact-software-quality-delivery/
- [5] Jonathan Bloom. Titanic Dilemma: The Seen Versus the Unseen. Available online at http://blog.castsoftware.com/titanic-dilemma-the-seen-versus-the-unseen/