Digital Academy Thailand

Software Testing Foundation Level

Lecture 3 – Static Testing

Uwe Gühl

KASETSART UNIVERSITY

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

Kasetsart University Sriracha Campus

Contents

Contents

Static Testing Basics

- Dynamic testing techniques
  ⇨ requires the execution of software
- Static testing techniques
  ⇨ without execution of software
  ⇨ early test activity

Static Testing Basics

- Types of static testing
  - Manual examination of work products
    ⇨ ***Review***: A type of static testing in which a work product or process is evaluated by one or more individuals to detect defects or to provide improvements
  - Tool-driven evaluation of the code or other work products
    ⇨ ***Static analysis***: The process of evaluating a component or system without executing it, based on its form, structure, content, or documentation

Work Products that Can Be Examined by Static Testing

- Specifications, including
  - business requirements,
  - functional requirements,
  - security requirements.
- Epics, user stories, and acceptance criteria
- Architecture and design specifications
- Code
- Testware, including
  - test plans,
  - test cases,
  - test procedures, and
  - automated test scripts

Work Products that Can Be Examined by Static Testing

- User guides
- Web pages
- Contracts, project plans, schedules, and budget planning
- Configuration set up and infrastructure set up
- Models, such as activity diagrams,
  → related to Model-Based testing

Work Products that Can Be Examined by Static Testing

- How to conduct static testing?
  - Reviews can be applied to any work product
    Precondition: Corresponding skills/knowledge
  - Static analysis can be applied
    - ➢ to any work product with a formal structure (typically code or models)
      Precondition: an appropriate static analysis tool exists.
    - ➢ with tools that evaluate work products written in natural language such as requirements (e.g., checking for spelling, grammar, and readability).

Benefits of Static Testing

1.3

- Enabling the early detection of defects before dynamic testing is performed, for example in
  - requirements or design specifications reviews,
  - backlog refinement.
- Identifying defects which are not easily found by dynamic testing
- Preventing defects in design or coding by uncovering inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies in requirements

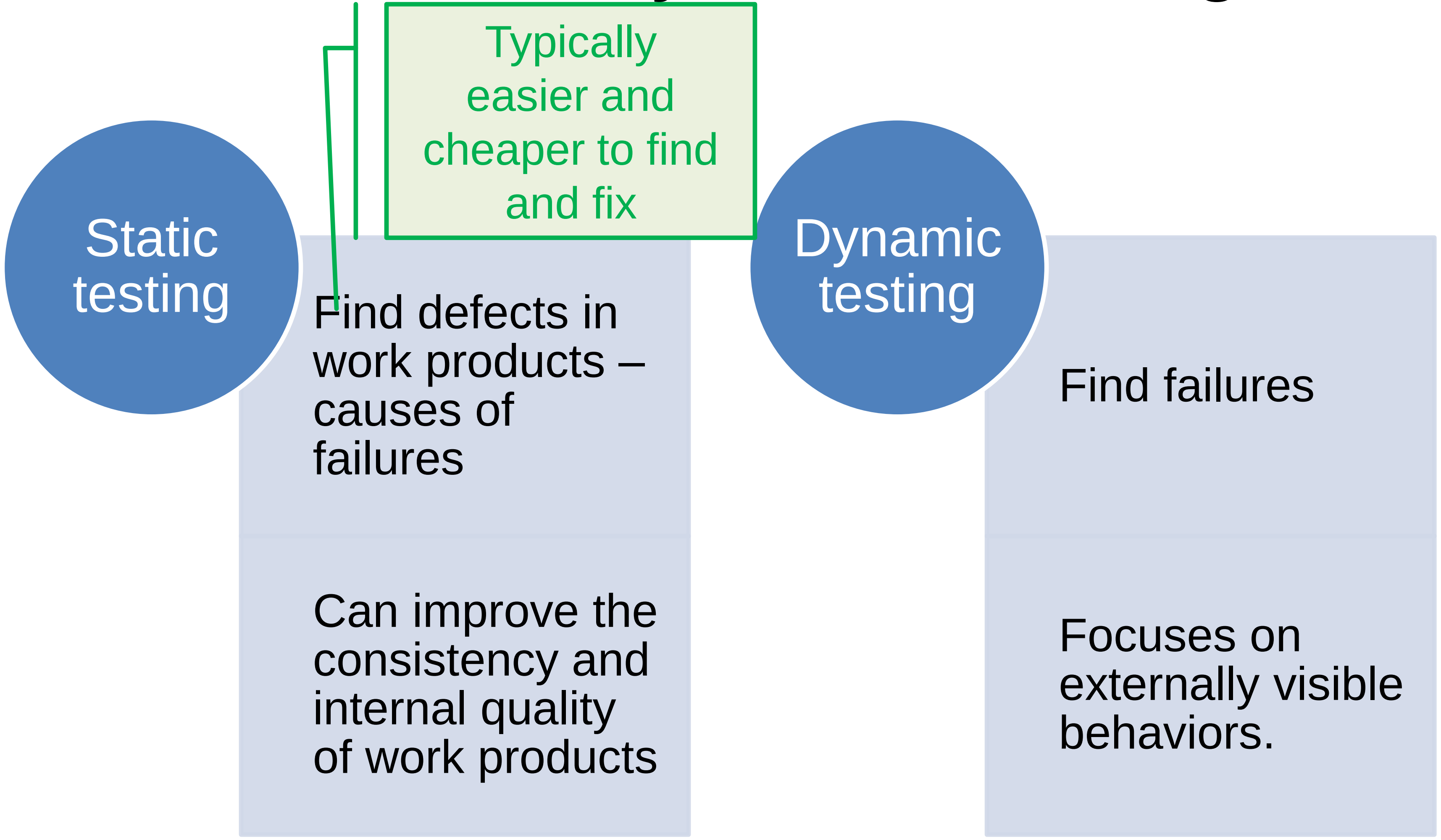
Benefits of Static Testing

- Increasing development productivity (e.g., due to improved design, more maintainable code)
- Reducing
  - development cost and time
  - testing cost and time
  - total cost of quality over the software's lifetime, due to fewer failures later in the lifecycle or after delivery into operation
- Improving communication between team members in the course of participating in reviews

Differences between Static and Dynamic Testing

- Static and dynamic testing
  - have the same objectives like 1.1
    - ➢ providing an assessment of the quality of the work products
    - ➢ identifying defects as early as possible
  - complement each other by finding different types of defects.

Differences between Static and Dynamic Testing


Static testing
Find defects in work products – causes of failures
Can improve the consistency and internal quality of work products
Typically easier and cheaper to find and fix
Dynamic testing
Find failures
Focuses on externally visible behaviors.

Differences between Static and Dynamic Testing

- Possible defects related to static testing:
  - Requirement defects (e.g., inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies)
  - Design defects (e.g., inefficient algorithms or database structures, high coupling, low cohesion)
  - Coding defects (e.g., variables with undefined values, variables that are declared but never used, unreachable code, duplicate code)
  - Deviations from standards (e.g., lack of adherence to coding standards)
  - Incorrect interface specifications (e.g., different units of measurement used by the calling system than by the called system)

Differences between Static and Dynamic Testing

- Possible defects related to static testing:
  - Security vulnerabilities (e.g., susceptibility to buffer overflows)
  - Gaps or inaccuracies in test basis traceability or coverage (e.g., missing tests for an acceptance criterion)
- Most types of maintainability defects can only be found by static testing
  - improper modularization,
  - poor reusability of components,
  - code that is difficult to analyze and modify without introducing new defects.

Summary

- Static testing:
  - no execution of software
  - finding defects in work products

  Dynamic testing:
  - execution of software
  - finding failures
- Both, static and dynamic testing, complement each other by finding different types of defects
- Static testing covers
  - reviews,
  - static analysis.
- Finding defects early is one of the most important benefits of static testing

Contents

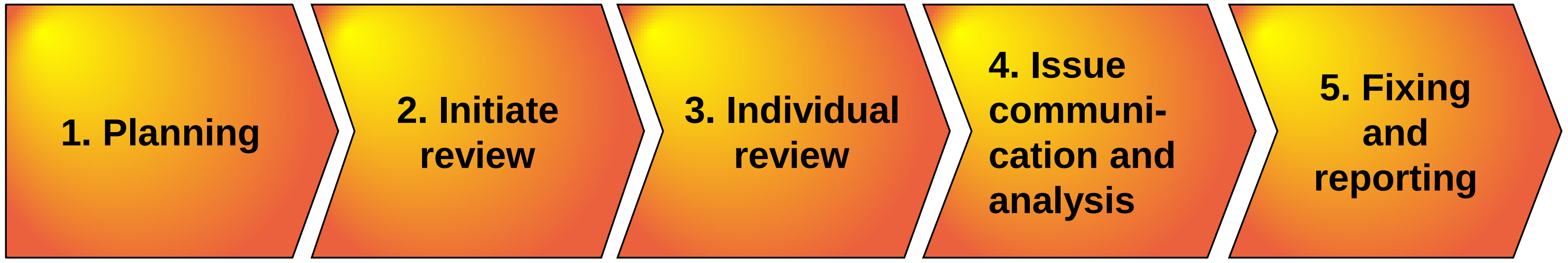
Review Process

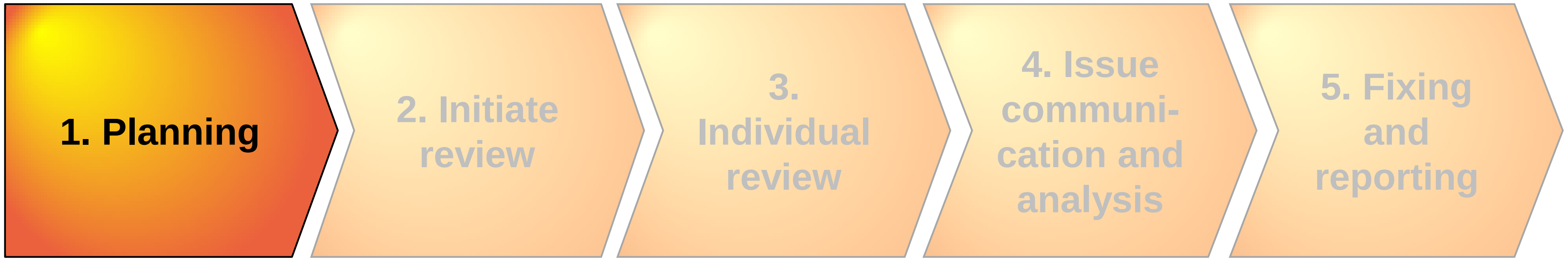
- Reviews vary from informal to formal.
  - ***Informal review***:
    A type of review that does not follow a defined process and has no formally documented output.
  - ***Formal review***:
    A type of review that follows a defined process with a formally documented output.

Review Process

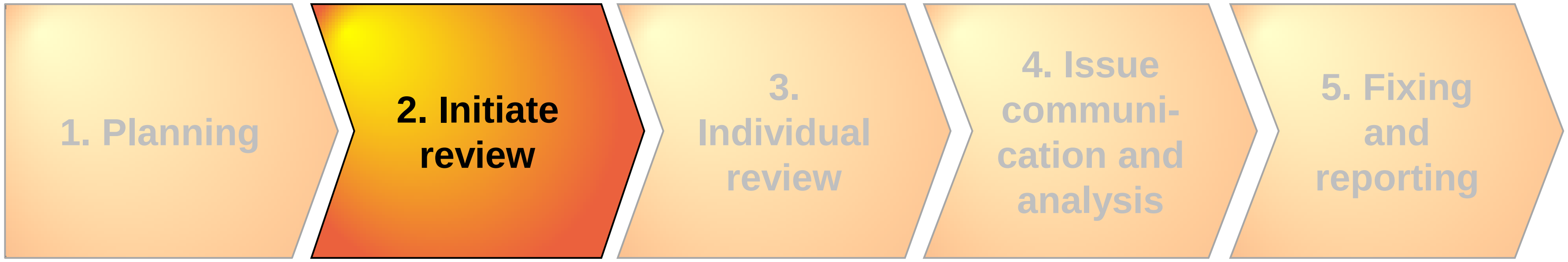
- The formality of a review process relates to
  - software development lifecycle model,
  - maturity of the development process,
  - complexity of the work product to be reviewed,
  - any legal or regulatory requirements,
  - need for an audit trail.
- The focus depends on agreed objectives
  - Finding defects
  - Gaining understanding
  - Educating participants such as testers and new team members
  - Discussing and deciding by consensus
- Standard ISO/IEC 20246 informs about reviews

Work Product Review Process

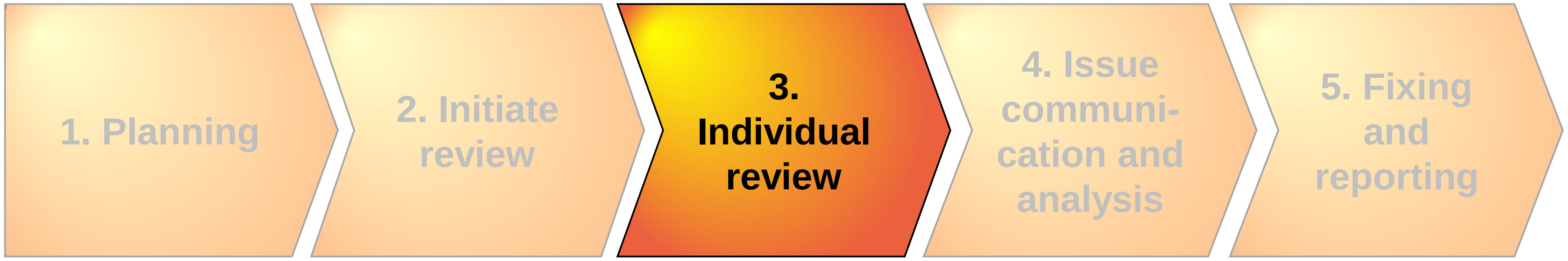




- Defining the scope
  - purpose of the review,
  - what documents or parts of documents to review, and
  - quality characteristics to be evaluated.
- Estimating effort and timeframe
- Identifying review characteristics such as the review type with roles, activities, and checklists
- Selecting the people to participate in the review and allocating roles
- Defining the entry and exit criteria for more formal review types like inspections
- Checking that entry criteria are met – for more formal review types



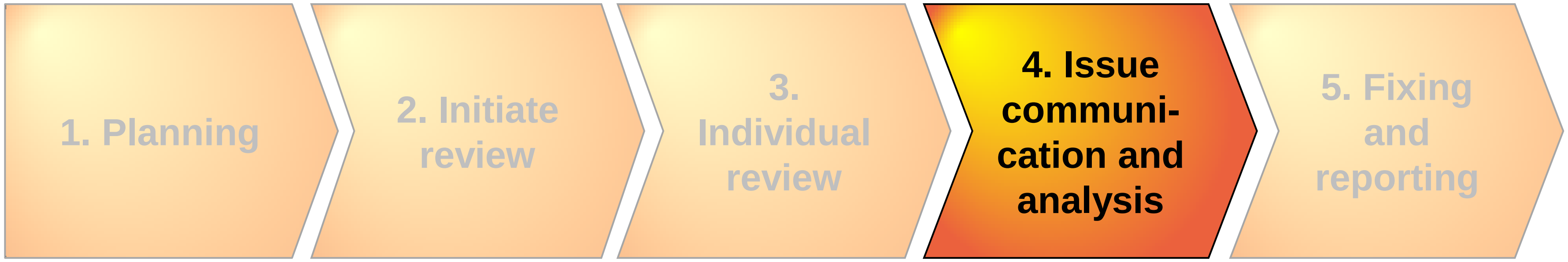
- Distributing
  - the work product (physically or by electronic means),
  - issue log forms,
  - checklists,
  - related work products.
- Explaining to the participants
  - scope,
  - objectives,
  - process,
  - roles,
  - work products.
- Answering all questions of participants about the review



In general executed as individual preparation

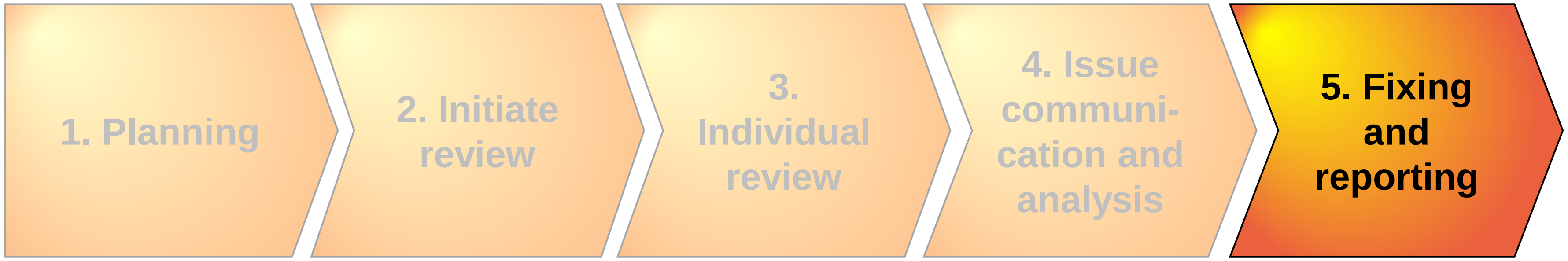
- Reviewing all or part of the work product
- Noting
  - potential defects,
  - recommendations,
  - questions.

Digital Academy Thailand



- Communicating identified potential defects, typically in a review meeting
- Analyzing potential defects, assigning ownership and status to them
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make a review decision
  - reject,
  - major changes needed,
  - accept,
  - accept with minor changes.

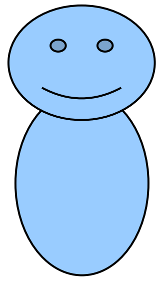
Digital Academy Thailand



- Creating defect reports
  for findings that require changes to a work product
- Fixing defects found in the work product reviewed
  - typically done by the author
- Communicating defects to the appropriate person or team (when found in a work product related to the work product reviewed)
- Recording updated status of defects (in formal reviews), potentially including the agreement of the comment originator
- Gathering metrics (for more formal review types)
- Checking that exit criteria are met (for more formal review types)
- Accepting the work product when the exit criteria are reached

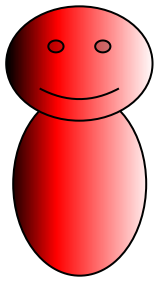
Roles and responsibilities in a formal review

- Author
  - Creates the work product under review
  - Fixes defects in the work product under review if necessary



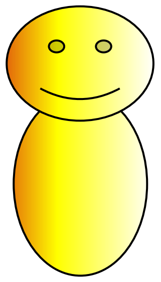
Roles and responsibilities in a formal review

- Management
  - Is responsible for review planning
  - Decides on the execution of reviews
  - Assigns staff, budget, and time
  - Monitors ongoing cost-effectiveness
  - Executes control decisions in the event of inadequate outcomes


Manager

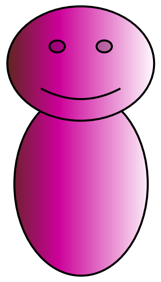
Roles and responsibilities in a formal review

- ***Moderator***
  (Synonyms: *inspection leader, facilitator):*
  The person responsible for running review meetings.
  - Ensures effective running of review meetings when held
  - Mediates, if necessary, between the various points of view
  - Is often the person upon whom the success of the review depends


Moderator

Roles and responsibilities in a formal review

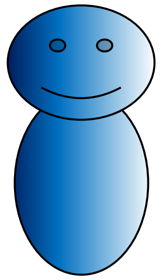
- Review leader
  - Takes overall responsibility for the review
  - Decides who will be involved and organizes when and where it will take place



Review leader

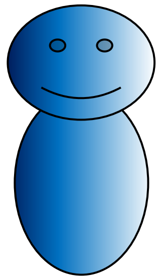
Roles and responsibilities in a formal review

- ***Reviewer*** ***(Synonyms:*** *checker, inspector):*
  A participant in a review, who identifies issues in the work product.
  - Background:
    - ➢ Subject matter experts,
    - ➢ Persons working on the project,
    - ➢ Stakeholders with an interest in the work product,
    - ➢ Individuals with specific technical or business backgrounds


Reviewer

Roles and responsibilities in a formal review

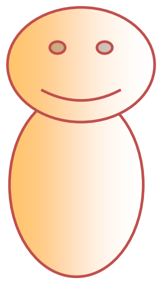
- Reviewer
  - Identifies potential defects in the work product under review
  - Typically represents different perspectives like
    - tester,
    - developer,
    - user,
    - operator,
    - business analyst,
    - usability expert.



Reviewer

Roles and responsibilities in a formal review

- ***Scribe*** *(**Synonym:** recorder)*:
  A person who records information during the review meetings.
  - Collates potential defects found during the individual review activity
  - Records from a review meeting (when held)
    - ➢ new potential defects,
    - ➢ open points, and
    - ➢ decisions
  - Least important role

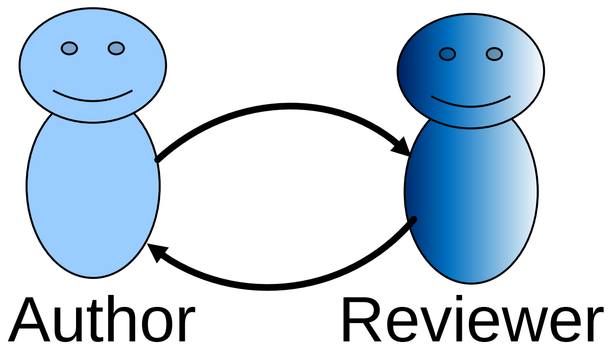

Scribe

Roles and responsibilities in a formal review

- Based on review type
  - one person may play more than one role,
  - actions associated with each role may vary.
- Standard ISO/IEC 20246 describes more roles

Review Types

- All presented review types
  - help finding defects
  - could be combined for one work product like
    - ➢ first an informal review,
    - ➢ then a technical review
  - can be done as ***peer reviews***:
    A type of review of work products performed by others qualified to do the same work.



Review Types

- **Informal review** (e.g., buddy check, pairing, pair review)
  - Main purpose: detecting potential defects
  - Possible additional purposes:
    - ➢ generating new ideas or solutions,
    - ➢ quickly solving minor problems
  - Not based on a formal (documented) process
  - May not involve a review meeting
  - May be performed by a colleague of the author (buddy check) or by more people
  - Results may be documented
  - Varies in usefulness depending on the reviewers
  - Use of checklists is optional
  - Very commonly used in Agile development

Review Types

- ***Walkthrough*** (***Synonym:*** *structured walkthrough):* A type of review in which an author leads members of the review through a work product and the members ask questions and make comments about possible issues.
  - Main purposes:
    - ➢ find defects,
    - ➢ improve the software product,
    - ➢ consider alternative implementations,
    - ➢ evaluate conformance to standards and specifications.

Review Types

- **Walkthrough**
  - Possible additional purposes:
    - ➢ exchanging ideas about techniques or style variations,
    - ➢ training of participants,
    - ➢ achieving consensus.
  - Individual preparation before the review meeting is optional
  - Review meeting is typically led by the author of the work product

Review Types

- **Walkthrough**
  - Scribe is mandatory
  - Use of checklists is optional
  - May take the form of
    - scenarios,
    - dry runs, or
    - simulations.
  - Potential defect logs and review reports are produced
  - May vary in practice from quite informal to very formal

Review Types

- ***Technical review***:
  A type of formal review by a team of technically-qualified personnel that examines the quality of a work product and identifies discrepancies from specifications and standards.
  - Main purposes:
    - gaining consensus,
    - detecting potential defects.

Review Types

- Technical review
  - Possible further purposes:
    - ➢ evaluating quality and building confidence in the work product,
    - ➢ generating new ideas, motivating
    - ➢ enabling authors to improve future work products,
    - ➢ considering alternative implementations.
  - Reviewers should be technical peers of the author, and technical experts in the same or other disciplines

Review Types

- Technical review
  - Individual preparation before the review meeting is required
  - Review meeting is optional, ideally led by a trained facilitator (typically not the author)
  - Scribe is mandatory, ideally not the author
  - Use of checklists is optional
  - Potential defect logs and review reports are produced

Review Types

- ***Inspection***:
  A type of formal review to identify issues in a work product, which provides measurement to improve the review process and the software development process.
  - Main purposes:
    - ➢ detecting potential defects,
    - ➢ evaluating quality and building confidence in the work product,
    - ➢ preventing future similar defects through author learning and root cause analysis

Review Types

- **Inspection**
  - Possible further purposes:
    - ➢ motivating and enabling authors to improve future work products and the software development process,
    - ➢ achieving consensus.
  - Follows a defined process
    - ➢ with formal documented outputs,
    - ➢ based on rules and checklists.

Review Types

- **Inspection**
  - Uses clearly defined roles
    - ➢ may include a dedicated reader
      During the review meeting he reads the work product aloud often paraphrase – describes it in own words
  - Individual preparation before the review meeting is required
  - Reviewers are
    - ➢ peers of the author or
    - ➢ experts in other disciplines that are relevant to the work product

Applying Review Techniques

- Different review techniques could be used during the individual review to uncover defects.
- The effectiveness of the techniques may differ depending on the type of review used.

Applying Review Techniques

- ***Ad hoc review***: A review technique performed informally without a structured process.
  - little or no guidance on how a review should be performed.
  - Reviewers often
    - ➢ read the work product sequentially
    - ➢ identify and document issues as they encounter them
  - commonly used technique
  - highly dependent on reviewer skills
  - may lead to many duplicate issues being reported by different reviewers.

Applying Review Techniques

- ***Checklist-based review***: A review technique guided by a list of questions or required attributes.
  - Review checklists
    - are distributed at review initiation
    - consist of a set of questions based on potential defects, which may be derived from experience.
    - should be specific to the type of work product under review
    - should be maintained regularly to cover issue types missed in previous reviews.
  - Main advantage: Systematic coverage of typical defect types.
  - Care should be taken not to simply follow the checklist in individual reviewing, but also to look for defects outside the checklist.

Applying Review Techniques

- Scenarios and dry runs
  - ***Scenario-based reviewing***: A review technique in which a work product is evaluated to determine its ability to address specific scenarios.
    - ➢ Reviewers get structured guidelines how to read through the work product.
    - ➢ Supports reviewers to do “dry runs” on the work product based on expected usage of the work product
    - ➢ Scenarios provide reviewers with better guidelines on how to identify specific defect types than simple checklist entries.
  - As with checklist-based reviews, in order not to miss other defect types (e.g., missing features), reviewers should not be constrained to the documented scenarios.

Applying Review Techniques

- ***Perspective-based reading***
  (Synonym: *perspective-based reviewing)*:
  A review technique in which a work product is evaluated from the perspective of different stakeholders with the purpose to derive other work products.
  - Typical stakeholder viewpoints include
    - end user,
    - marketing,
    - designer,
    - tester,
    - operations.

Applying Review Techniques

- Perspective-based reading
  - Using different stakeholder viewpoints leads to
    - ➢ more depth in individual reviewing
    - ➢ less duplication of issues across reviewers
  - Checklists often used
  - Example:
    - ➢ Work product: requirements specification
    - ➢ Task: A tester should generate draft acceptance tests
    - ➢ Perspective-based reading => all information there?
  - Result of empirical studies:
    - ➢ Perspective-based reading is the most effective general technique for reviewing requirements and technical work products.

Applying Review Techniques

- ***Role-based reviewing***: A review technique in which a work product is evaluated from the perspective of different stakeholders.
  - Specific end user types like
    - ➢ experienced/inexperienced,
    - ➢ senior/child.
  - Specific roles in the organization, such as
    - ➢ user administrator,
    - ➢ system administrator,
    - ➢ performance tester.
  - Same principles as in perspective-based reading

Success Factors for Reviews

- Organizational success factors for reviews
  - Each review has clear objectives, defined during review planning, and used as measurable exit criteria
  - Review types are applied which are suitable to achieve the objectives and are appropriate to the type and level of software work products and participants
  - Any review techniques used, such as checklist-based or role-based reviewing, are suitable for effective defect identification in the work product to be reviewed
  - Any checklists used address the main risks and are up to date

Success Factors for Reviews

- Organizational success factors for reviews
  - Large documents
    - ➢ are written and reviewed in small chunks
    - ➢ quality control is exercised by providing authors early and frequent feedback on defects
  - Participants have adequate time to prepare
  - Reviews are scheduled with adequate notice
  - Management supports the review process (e.g., by incorporating adequate time for review activities in project schedules)
  - Reviews are integrated in the company's quality and/or test policies.

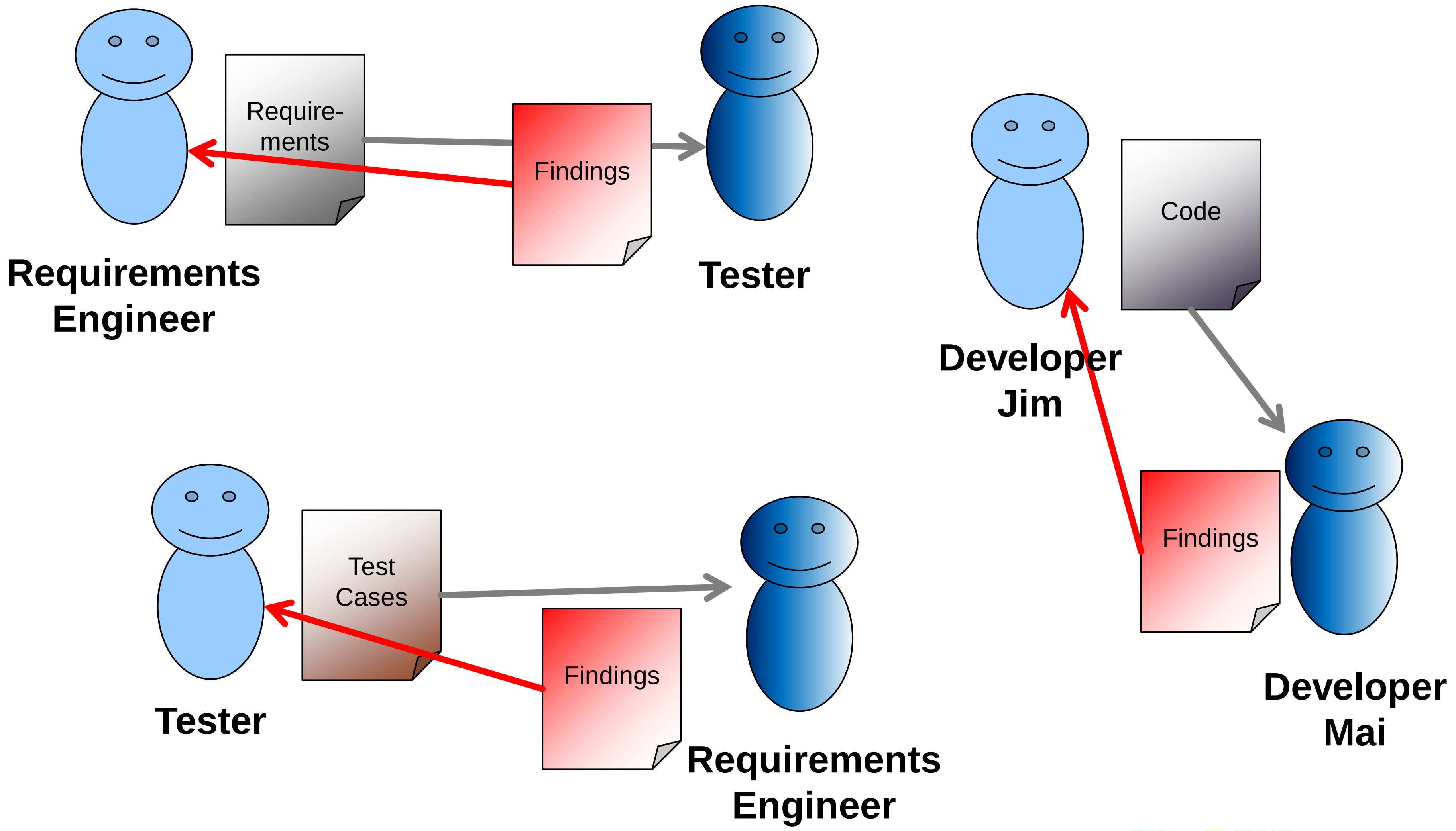
Success Factors for Reviews

- People-related success factors
  - The right people are involved to meet the review objectives, for example, people with different skill sets or perspectives, who may use the document as a work input
  - Testers are seen as valued reviewers
    - ➢ contribute to the review
    - ➢ learn about the work product,
    - ➢ enables them to prepare earlier more effective tests,
  - Participants dedicate adequate time and attention to detail
  - Reviews are conducted on small chunks
    - ➢ Reviewers do not lose concentration during
      - ❖ individual review and/or
      - ❖ the review meeting (when held)

Success Factors for Reviews

- People-related success factors
  - Defects found are acknowledged, appreciated, and handled objectively
  - The meeting is well-managed, valuable use of time
  - The review is conducted in an atmosphere of trust; the outcome will not be used for the evaluation of the participants
  - Participants avoid body language and behaviors that might indicate boredom, exasperation, or hostility to other participants
  - Adequate training is provided, especially for more formal review types such as inspections
  - A culture of learning and process improvement is promoted

Examples for Reviews



Summary

- Main review process activities are
  1. Planning
  2. Initiate review
  3. Individual review
  4. Issue communication and analysis
  5. Fixing and reporting

- Roles in reviews
  - Author
  - Management
  - Moderator (or facilitator)
  - Review leader
  - Reviewers
  - Scribe (or recorder)

Summary

- The four most common types of reviews are
  - Informal review
  - Walkthrough
  - Technical review
  - Inspection
- Review techniques
  - Ad hoc
  - Checklist-based
  - Scenarios and dry runs
  - Perspective-based
  - Role-based

- Success factor for reviews:
  Testers are seen as valued reviewers

Contents

Static Analysis by Tools

- Static analysis
  - important for safety-critical computer systems (e.g., aviation, medical, or nuclear software),
  - important part of security testing,
  - often incorporated into automated software build and distribution tools, for example in
    - ➢ agile development,
    - ➢ continuous delivery, and
    - ➢ continuous deployment.

Static Analysis by Tools

Excurses

- Static analysis tools analyse
  - program code like
    - ➢ control flow
    - ➢ data flow
  - generated output like
    - ➢ HTML
    - ➢ XML

Static Analysis by Tools

Excurses

- Benefits
  - Early detection of defects prior to test execution
  - Early warning about suspicious aspects of the code or design by the calculation of metrics, such as a high complexity measure
  - Identification of defects not easily found by dynamic testing
  - Detecting dependencies and inconsistencies in software models such as links
  - Improved maintainability of code and design
  - Prevention of defects,
    if lessons are learned in development

Static Analysis by Tools

Excurses

- Typical defects discovered
  - Referencing a variable with an undefined value
  - Inconsistent interfaces between modules and components
  - Variables that are not used or are improperly declared
  - Unreachable (dead) code
  - Missing and erroneous logic (potentially infinite loops)
  - Overly complicated constructs
  - Programming standards violations
  - Security vulnerabilities
  - Syntax violations of code and software models

Data flow analysis

Excurses

- For every variable there is a status
  - d = defined
    The variable gets defined.
    A value gets assigned, the variable has a value.
  - r = referenced
    The variable gets read or is used.
  - u = undefined
    The variable has no defined value.

Data flow analysis

Excurses

- Anomalies
  - dd (defined / defined)
    Defined, then gets defined again before first value gets used
  - du (defined / undefined)
    Defined, then gets invalid or undefined without use
  - ur (undefined / referenced)
    Undefined variable read or used

Data flow analysis

Excurses

- Anomalies – examples
  - dd

```
int x = function1();
x = function2();     // redefinition of x → dd
```

  - du

```
{
  int x = 2;
}                    // x undefined at exit → du
```

  - ur

```
int x;               // x undefined
int y = x;           // x referenced → ur
```

Data flow analysis

Excurses

Example: Function `MinMax` should sort 2 numbers

	Help	Min	Max
`void MinMax(int& Min, int& Max)`		d	d
`{`			
`    int Help;`	u		
`    if (Min > Max)`		r	r
`    {`			
`        Max = Help;`	r		d
`        Max = Min;`		r	d
`        Help = Min;`	d	r	
`    }`			
`}`	u		

ur Anomaly

dd Anomaly

du Anomaly

Tools for Static Code Analysis

Excurses

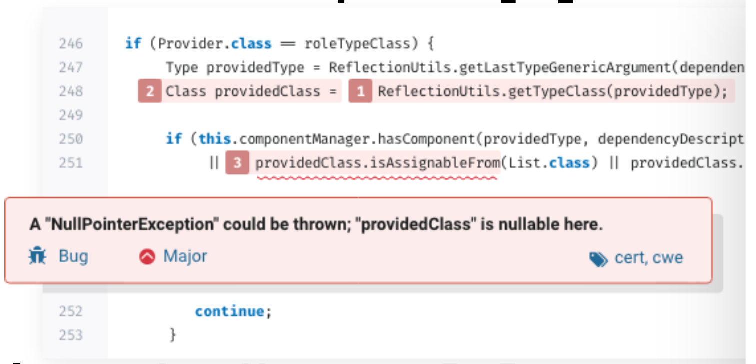
- Tools for static code analysis for different program languages were collected [1], [2]
- 4 static analysis tools for Java have been compared [3]
  - Jtest has had the highest defection ratio
  - Findbugs as open source tool was second
  - Advice from the authors: Take the respective advantage of several tools for detecting bugs in different categories

Sources: [1] https://www.codeanalysistools.com/
[2] https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis/
[3] *Md. Abdullah Al Mamun, Aklima Khanam, Håkan Grahn, and Robert Feldt: Comparing Four Static Analysis Tools for Java Concurrency Bugs, 2010,* *http://robertfeldt.net/publications/grahn_2010_comparing_static_analysis_tools_for_concurrency_bugs.pdf*

Tools for Static Code Analysis

Excurses

- Example: Sonarqube [1]



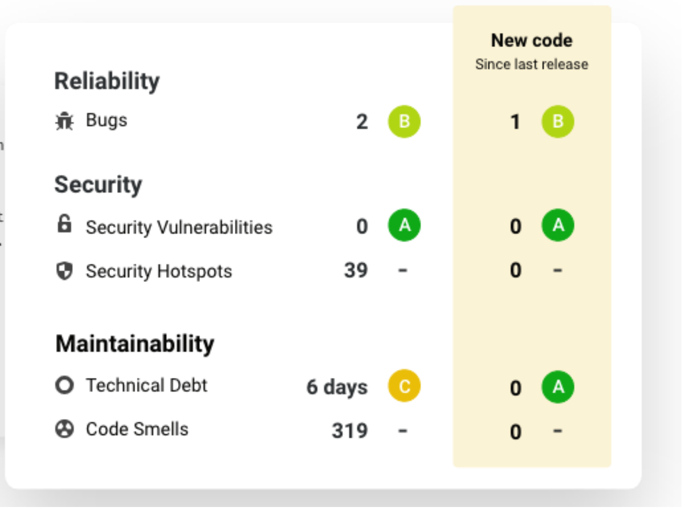
```
        if (Provider.class == roleTypeClass) {
            Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependen
            Class providedClass = ReflectionUtils.getTypeClass(providedType);

            if (this.componentManager.hasComponent(providedType, dependencyDescript
                || providedClass.isAssignableFrom(List.class) || providedClass.
```

A "NullPointerException" could be thrown; "providedClass" is nullable here.

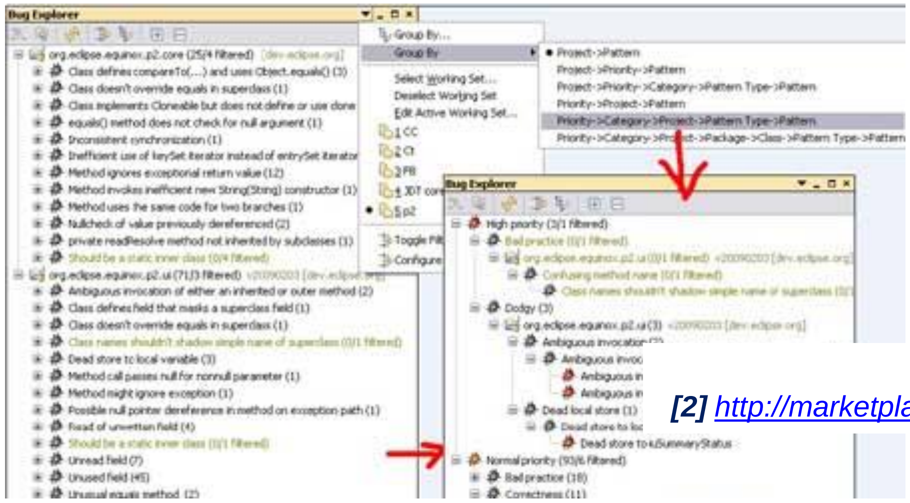
Bug | Major | cert, cwe

```
                continue;
            }
```



			New code Since last release	
Reliability				
Bugs	2	B	1	B
Security				
Security Vulnerabilities	0	A	0	A
Security Hotspots	39	-	0	-
Maintainability				
Technical Debt	6 days	C	0	A
Code Smells	319	-	0	-

- Example: Findbugs [2]



Bug Explorer

```
⊟ org.eclipse.equinox.p2.core (25/4 filtered) [dev.eclipse.org]
   ⊞ Class defines compareTo(...) and uses Object.equals() (3)
   ⊞ Class doesn't override equals in superclass (1)
   ⊞ Class implements Cloneable but does not define or use clone
   ⊞ equals() method does not check for null argument (1)
   ⊞ Inconsistent synchronization (1)
   ⊞ Inefficient use of keySet iterator instead of entrySet iterator
   ⊞ Method ignores exceptional return value (12)
   ⊞ Method invokes inefficient new String(String) constructor (1)
   ⊞ Method uses the same code for two branches (1)
   ⊞ Nullcheck of value previously dereferenced (2)
   ⊞ private readResolve method not inherited by subclasses (1)
   ⊞ Should be a static inner class (0/4 filtered)
⊟ org.eclipse.equinox.p2.ui (71/3 filtered) v20090203 [dev.eclipse.org]
   ⊞ Ambiguous invocation of either an inherited or outer method (2)
   ⊞ Class defines field that masks a superclass field (1)
   ⊞ Class doesn't override equals in superclass (1)
   ⊞ Class names shouldn't shadow simple name of superclass (0/1 filtered)
   ⊞ Dead store to local variable (3)
   ⊞ Method call passes null for nonnull parameter (1)
   ⊞ Method might ignore exception (1)
   ⊞ Possible null pointer dereference in method on exception path (1)
   ⊞ Read of unwritten field (4)
   ⊞ Should be a static inner class (0/1 filtered)
   ⊞ Unread field (7)
   ⊞ Unused field (45)
   ⊞ Unusual equals method (2)
```

Group By...
Group By
Select Working Set...
Deselect Working Set
Edit Active Working Set...
1 CC
2 Ct
3 FB
4 JDT core
5 p2
Toggle Filter
Configure
Project->Pattern
Project->Priority->Pattern
Project->Priority->Category->Pattern Type->Pattern
Priority->Project->Pattern
Priority->Category->Project->Pattern Type->Pattern
Priority->Category->Project->Package->Class->Pattern Type->Pattern

Bug Explorer

```
⊟ High priority (3/1 filtered)
   ⊟ Bad practice (0/1 filtered)
      ⊟ org.eclipse.equinox.p2.ui (0/1 filtered) v20090203 [dev.eclipse.org]
         ⊟ Confusing method name (0/1 filtered)
               Class names shouldn't shadow simple name of superclass (0/1
   ⊟ Dodgy (3)
      ⊟ org.eclipse.equinox.p2.ui (3) v20090203 [dev.eclipse.org]
         ⊟ Ambiguous invocation (2)
            ⊟ Ambiguous invoc
                  Ambiguous in
                  Ambiguous in
         ⊟ Dead local store (1)
            ⊟ Dead store to loc
                  Dead store to isSummaryStatus
⊟ Normal priority (93/6 filtered)
   ⊞ Bad practice (18)
   ⊟ Correctness (11)
```

Sources: [1] https://www.sonarqube.org/
[2] *http://marketplace.eclipse.org/content/findbugs-eclipse-plugin*

Summary

Excurses

- Static Analysis by Tools offers a lot of benefits, especially early detection of defects prior to test execution
- Data flow analysis to detect anomalies
  - dd (defined / defined)
  - du (defined / undefined)
  - ur (undefined / referenced)
- Several tools for static code analysis for different programming languages are available, commercial and open source versions