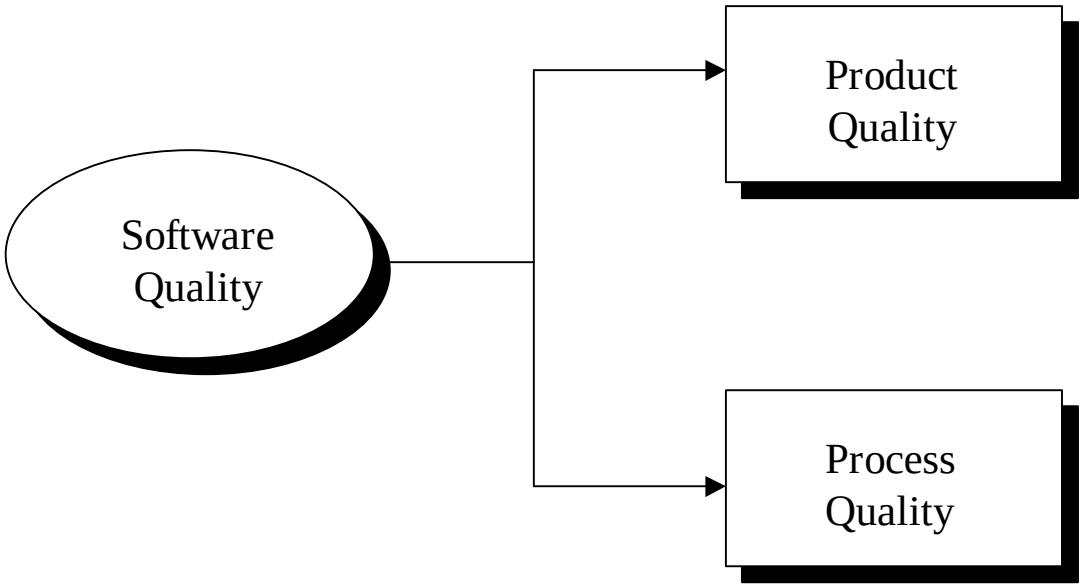
Aims of the LHCb Architecture Review

Outline Of Talk

- Software quality…some principles
- Architecture Review Goals
- Scenarios
- Software Architecture Analysis Method (SAAM)
- Questionnaires and checklists
- Conducting the review in practice
  - preparation
  - participation
  - documents to be reviewed
  - the inspection itself
  - output of the review

Software Quality



•documents
•designs
•code
•tests

•techniques
•tools
•people

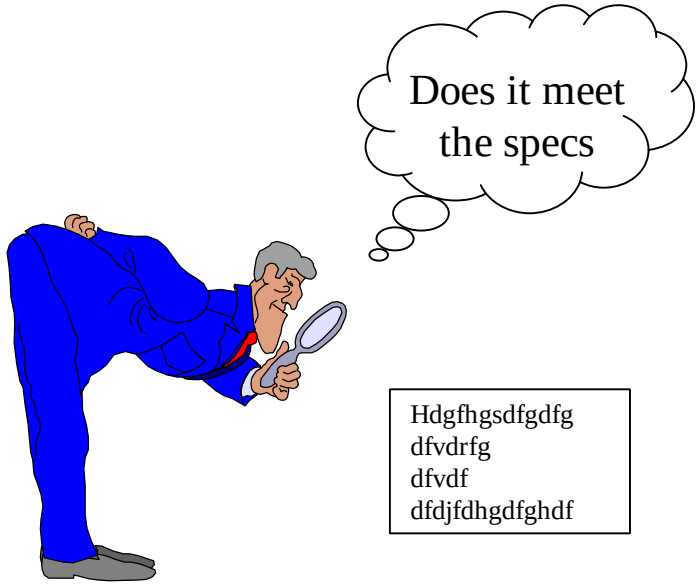
- Different stakeholders see quality in different terms :
  - End user …do job better, faster, easier
  - Developer …few defects, error free at delivery
  - Maintainer …understandable, testable, modifiable
- You can't achieve quality unless you specify it

Quality - as measured by the user

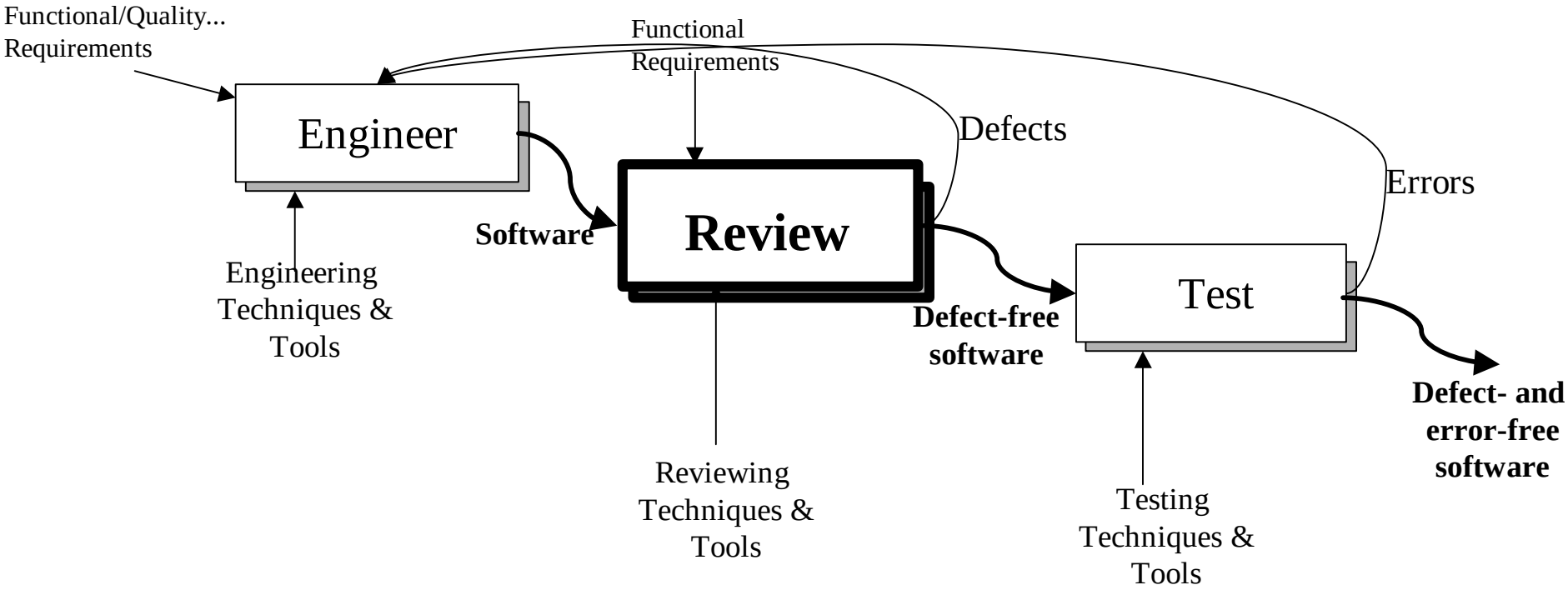
Correctness	does software do what its supposed to?
Efficiency	how many resources are needed?
Expandability	how easy is it to expand software?
Flexibility	how easy is it to change it?
Integrity	how secure is it?
Interoperability	does it interface easily?
Maintainability	how easy is it to repair?
Manageability	is it easily managed?
Portability	how easy is it to transport?
Usability	how easy is it to use?
Reliability	how often will it fail?
Reusability	is it reusable in other systems?
Safety	does it prevent hazards?
Survivability	Can it survive during failure?
Verifiability	can it be tested?

What is a review ?

- A Review is one of three steps in quality life-cycle
  - Step 1 : ***Engineer*** - build-in quality
  - Step 2 : ***Review*** - shake out major defects
  - Step 3 : ***Test*** - eradicate remaining errors



What is a Review ?



N.B. Software means Design + Code

Architecture Review Goals

- Evaluate early and carefully before it becomes a blueprint for software
- Input to review is the description of the architecture
  - assignment of functionality, nature of interfaces
  - exact contents depend on, or will determine, what aspect is to be assessed e.g.
    - performance - ***task and communication structure***
    - modifiability - ***component structure*** and ***work assignments***
- Point out places where architecture fails to meet requirements and show alternative designs that would work better.
- Determine where finer-grain depictions of architecture are needed
- Ensure consistency across entire system
- Disseminate ideas on what constitutes a good architecture
    - better understanding, deeper insights on architecture
- Determine whether can proceed to next stage of development
- Reapply - put in regular use

Review Techniques - Scenarios

- Quality attributes don't have a meaning in isolation but only within a context
- Context based evaluation is most effective - use scenarios
- Each scenario is a brief description of a single interaction of a stakeholder with a system
  - e.g. operator : "I want to change background colour on all windows to blue"
- Evaluation process revolves around characterising an architecture in terms of how well it supports those scenarios that represent the stakeholders' overriding concerns.
- The total number of scenarios must be manageable, so they must be chosen carefully. Need a selection process
  - brainstorming
  - refine and select - mark, cluster by subject, rank, merge or delete

Software Architecture Analysis Method (SAAM) -I

- 1 - Develop scenarios
  - illustrate kind of **functions** system must support
  - illustrate kinds of **changes** you anticipate will be made to system
  - capture all important **uses** of system
  - capture all important **users** of system
  - capture all important **qualities** system must satisfy
  - Is each design module correlated with a scenario?
  - Is each major requirement correlated with a scenario?
- 2 - Describe candidate architecture
  - use well understood notation (by reviewers as well)
  - **static** - must cover computation and data components, and their connections
  - **dynamic** - must cover behaviour with time

Software Architecture Analysis Method (SAAM) - II

- 3 - Classify scenarios
  - **direct** - architecture directly supports scenario
  - **indirect** - change to system needed that can be represented architecturally
  - degree of modification must be captured when evaluating a system's response to a scenario.
- 4 - Perform scenario evaluations
  - For each scenario, the changes to the architecture required to support it must be listed, and the cost of making change estimated
    - addition of a new component
    - change in specification of existing component
  - Produce summary table
    - for each scenario describe impact on architecture
    - weight degree - coarse grained e.g. major, minor; useful if comparing architectures

Software Architecture Analysis Method (SAAM) - III

- 5 - Reveal Scenario interaction
  - reveal components that are the focus of too many changes
  - when 2 or more indirect scenarios require changes in a single component they are said to **interact** in that component.
  - Interaction of semantically unrelated scenarios shows which components are implementing semantically unrelated functions - indicates poor cohesion

Capture Results of Review

- Scenario Summary Table

Scenario	*Description*	*Direct/ Indirect*	*Changes Required*
1.	Compare new file representations	Indirect	Mods to *diff* and *vsdiff*
….			

- Scenario interactions by module

Module	*Number of changes*
Main	2
Algorithm factory	7
Event selector	1

  - Algorithm factory
    - scenarios all of same class - good sign…functionality sensibly allocated
    - scenarios different but component can be further subdivided…OK refine arch.
    - scenarios different classes and component cannot be subdivided..too complex

Review techniques - Questioning

- Scenario-based : such as SAAM
- Questionnaire-based : list of general and open questions that apply to all architectures
  - way architecture was generated
    - is there a project architect?
    - Is a standard description language used?
  - Details of architecture description itself
    - are user interface aspects separated from functional aspects
  - Utility is related to ease with which domain can be characterised
- Checklist-based
  - detailed set of questions after experience analysing common set of systems
  - usually domain specific
- Scenarios are usually system specific, can grow to others after experience
  - what happens when divide by zero occurs? (scenario)
  - is there error recovery code (checklist)

Review Stages

- Preparation
  - select right people
  - circulate review documents in advance
  - individual review of documentation
- Inspection
  - questions asked and errors found should be recorded in a database
  - statistics accumulated
- Re-work
  - designers rectify defects
- Follow-up
  - verify that the rework has been done

Preparation of Review

- Scope must be kept under control
- Firstly have architecture discovery review
  - held early, lightweight, before architectural decisions set in stone
  - expect people to be receptive to changes
- Have in mind set of specific goals, which should be enumerated
- Documents related to the project distributed before it takes place
  - need a project librarian to prepare this
- Organisational expectations
  - who will be told what as a result of review (developers, managers, reviewers)
- Need a detailed but flexible agenda for the inspection

Present during the Inspection

- Moderator - runs the meeting
  - John
- Reader - paraphrases the design
  - Pere
- Reviewers - questions the reader when necessary
  - Christian, RD, Lassi, Vincenzo, Dirk
- Recorder - notes down details on special forms
  - Marco
- Development team - must play passive role i.e. only answer questions
  - Pavel, Markus, Florence, Andrei, Rado, Iain
- Total 14 people
- What about users of system?

Documents to be reviewed

- Materials that describe the architecture
  - component model, assignment of functionality, definition of interfaces
  - message trace diagrams demonstrating dynamic behaviour
  - rational behind key architectural decisions taken (e.g. transient vs persistent, converters)
- A ranking of the most important (5-10) quality and functional requirements of the system
  - if required, additional attributes can be expressed but labelled as essential or desirable
- List of scenarios
  - clustered according to attribute under test
  - coverage of stakeholders interests, all aspects of model
- Any checklists or questionnaires to be used during inspection
- Descriptions of prototypes (if they exist)
  - test suitability of user interface, give feedback on performance issues
- Project Plan
  - work breakdown structure and assignment to individuals

Review activities - I

- Understand essential functions of system and see how each function is unambiguously defined in architecture design documents.
- Make sure that there are written requirements in all key areas
- Check that system acceptance criteria exist
- To evaluate performance related and other goals need to know:
  - workload information …number of concurrent users, request rates etc.
  - execution paths, expectation of execution of each component, repetitions, protocol for contention resolution
  - environmental information
- To evaluate resource utilisation need information on
  - cpu utilisation, i/o activity, database accesses, memory usage

Review activities - II

- To evaluate modifiability best done with questioning techniques to reveal how vulnerable architecture is to specific modifications.
- Warning signs of problems are :
  - architecture forced to match current organisation
  - top level components number more than 25
  - one requirement drives rest of design
  - architecture depends on alternatives in OS
  - proprietary components are being used where standard components would do
  - component definition comes from hardware builder
  - redundancy not needed for relaibility (e.g. 2 databases, 2 error handling components)
  - design is exception driven I.e. emphasis on extensibility and not core commonalities
  - no architect for the system

Review Output

- Categorised and ranked issues are formally documented
- Formal report delivered to review sponsor and participants
- Review process should be documented and aggregated output of several reviews should be collected and used to devise improvements and for training.