SOFTWARE REQUIREMENT ENGINEERING

LECTURE NO: 14

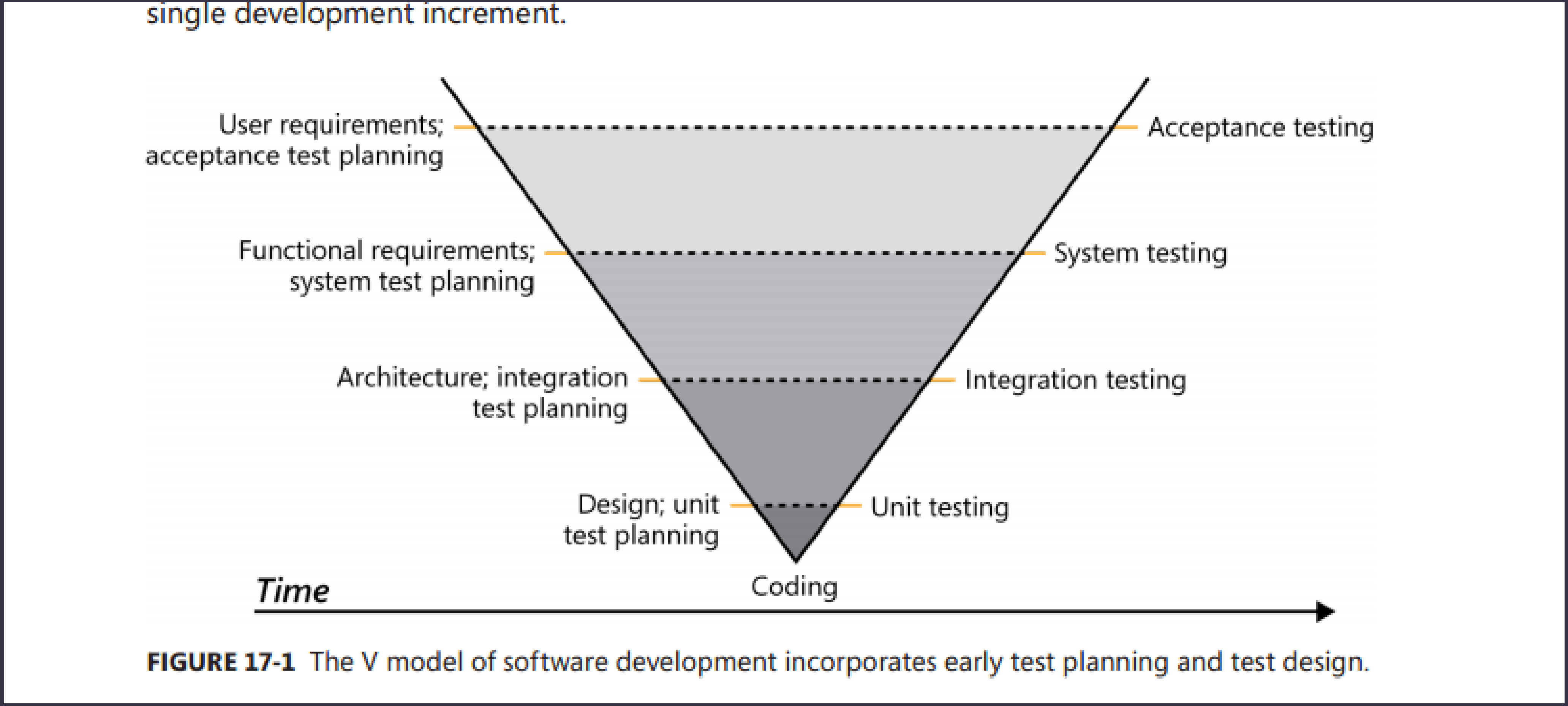
BY: NAZISH NOUMAN

Validating the requirements

Introduction..

- ❑On many projects, testing is a late-stage activity.
- ❑Requirements-related problems linger in the product until they’re finally revealed through time-consuming system testing or—worse—by the end user.
- ❑If you start your test planning and test-case development in parallel with requirements development, you’ll detect many errors shortly after they’re introduced.
- ❑This prevents them from doing further damage and minimizes your development and maintenance costs.

The V model of software development



single development increment.

User requirements; acceptance test planning
Acceptance testing
Functional requirements; system test planning
System testing
Architecture; integration test planning
Integration testing
Design; unit test planning
Unit testing
Coding
Time

FIGURE 17-1 The V model of software development incorporates early test planning and test design.

Validation and Verification

- Requirements validation is the fourth component of requirements development, along with elicitation, analysis, and specification.
- Verifying requirements to ensure that they have all the desired properties of high-quality requirements is also an essential activity.
- Precisely speaking, validation and verification are two different activities in software development.
- *Verification* determines whether the product of some development activity meets its requirements (doing the thing right).
- *Validation* assesses whether a product satisfies customer needs (doing the right thing).

Validation and Verification

Extending these definitions to requirements,

Verification determines whether you have written the requirements right: your requirements have the desirable properties

Validation of requirements assesses whether you have written the right requirements: they trace back to business objectives.

Validating requirements

Validating requirements allows teams to build a correct solution that meets the stated business objectives. Requirements validation activities attempt to ensure that:

■ The software requirements accurately describe the intended system capabilities and properties that will satisfy the various stakeholders’ needs.
■ The software requirements are correctly derived from the business requirements, system requirements, business rules, and other sources.
■ The requirements are complete, feasible, and verifiable.
■ All requirements are necessary, and the entire set is sufficient to meet the business objectives.
■ All requirements representations are consistent with each other.
■ The requirements provide an adequate basis to proceed with design and construction

- Validation isn’t a single discrete phase that you perform after eliciting and documenting all the requirements. Some validation activities, such as incremental reviews of the growing requirements set, are threaded throughout the iterative elicitation, analysis, and specifiation processes.
- Other activities, such as formal inspections, provide a fial quality gate prior to baselining a set of requirements.
- Include requirements validation activities as tasks in your project plan. Of course, you can validate only requirements that have been documented, not implicit requirements that exist only in someone’s mind.

Reviewing requirements

Anytime someone other than the author of a work product examines the product for problems, a *peer review* is taking place.

Reviewing requirements is a powerful technique for identifying ambiguous or unverifible requirements, requirements that aren’t defied clearly enough for design to begin, and other problems.

Different kinds of peer reviews go by a variety of names

Informal Reviews

Informal reviews are useful for educating other people about the product and collecting unstructured feedback. However, they are not systematic, thorough, or performed in a consistent way. Informal review approaches include:

■ A ***peer deskcheck***, in which you ask one colleague to look over your work product.
■ A ***passaround***, in which you invite several colleagues to examine a deliverable concurrently.
■ A ***walkthrough***, during which the author describes a deliverable and solicits comments on it.

Formal Reviews

- Formal peer reviews follow a well-defied process.
- A formal requirements review produces a report that identifies the material examined, the reviewers, and the review team’s judgment as to whether the requirements are acceptable.
- The principal deliverable is a summary of the defects found and the issues raised during the review.
- The members of a formal review team share responsibility for the quality of the review, although authors ultimately are responsible for the quality of the deliverables they create.
- The best-established type of formal peer review is called an *inspection*.

The inspection

- ❑Inspection of requirements documents is one of the highest-leverage software quality techniques available.
- ❑Inspection has been recognized as a software industry best practice (Brown 1996).
- ❑Any software work product can be inspected, including requirements, design documents, source code, test documentation, and project plans.

The inspection process

- ❑Inspection is a well-defied multistage process. It involves a small team of participants who carefully examine a work product for defects and improvement opportunities. Inspections serve as a quality gate through which project deliverables must pass before they are baselined.

Participants:

- ❑**The author of the work product and perhaps peers of the author**
- ❑**People who are the sources of information that fed into the item being inspected**
- ❑**People who will do work based on the item being inspected**
- ❑**People who are responsible for interfacing systems that will be affected by the item being inspected**

Participants

The author of the work product and perhaps peers of the author The business analyst who wrote the requirements document provides this perspective. Include another experienced BA if you can, because he'll know what sorts of requirements-writing errors to look for.

■ **People who are the sources of information that fed into the item being inspected** These participants could be actual user representatives or the author of a predecessor specification. In the absence of a higher-level specification, the inspection must include customer representatives, such as product champions, to ensure that the requirements describe their needs correctly and completely

Participants

People who will do work based on the item being inspected For an SRS, you might include a developer, a tester, a project manager, and a user documentation writer because they will detect different kinds of problems. A tester is most likely to catch an unverifiable requirement; a developer can spot requirements that are technically infeasible.

■ **People who are responsible for interfacing systems that will be affected by the item being inspected** These inspectors will look for problems with the external interface requirements. They can also spot ripple effects, in which changing a requirement in the SRS being inspected affects other systems

Inspection roles

Author The author created or maintains the work product being inspected. The author of a requirements document is usually the business analyst who elicited customer needs and wrote the requirements.

Moderator: The moderator plans the inspection with the author, coordinates the activities, and facilitates the inspection meeting.

The moderator distributes the materials to be inspected, along with any relevant predecessor documents, to the participants a few days before the inspection meeting. responsibilities include starting the meeting on time, encouraging contributions from all participants, and keeping the meeting focused on finding major defects rather than resolving problems or being distracted by minor stylistic issues and typos.

The moderator follows up on proposed changes with the author to ensure that the issues that came out of the inspection were addressed properly.

Inspection roles

Reader :One inspector is assigned the role of reader.

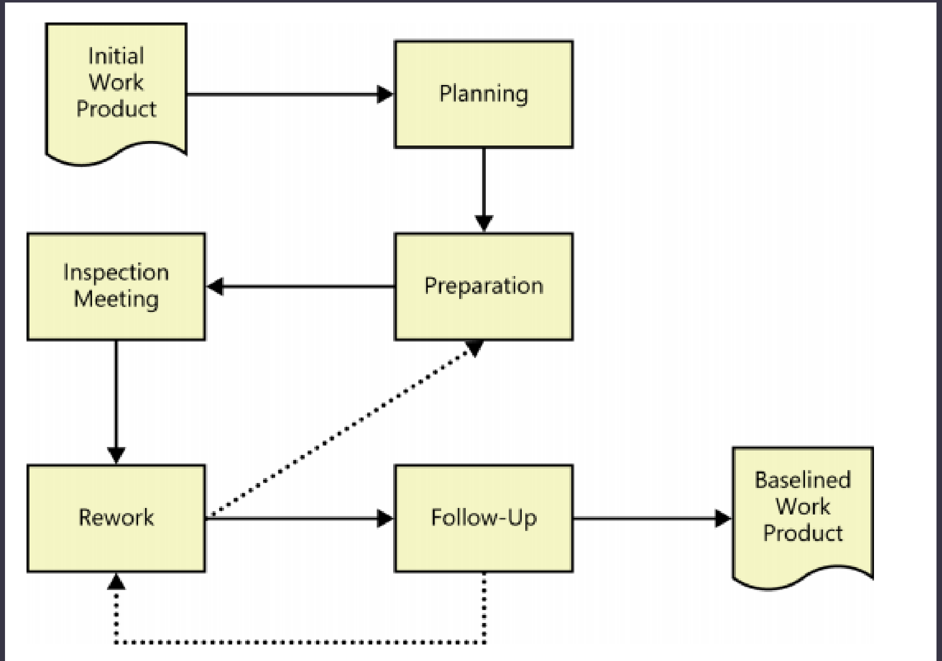
- During the inspection meeting, the reader paraphrases the requirements and model elements being examined one at a time.
- The other participants then point out potential defects and issues that they see.
- By stating a requirement in her own words, the reader provides an interpretation that might differ from that held by other inspectors.

Recorder The recorder uses standard forms to document the issues raised and the defects found during the meeting.

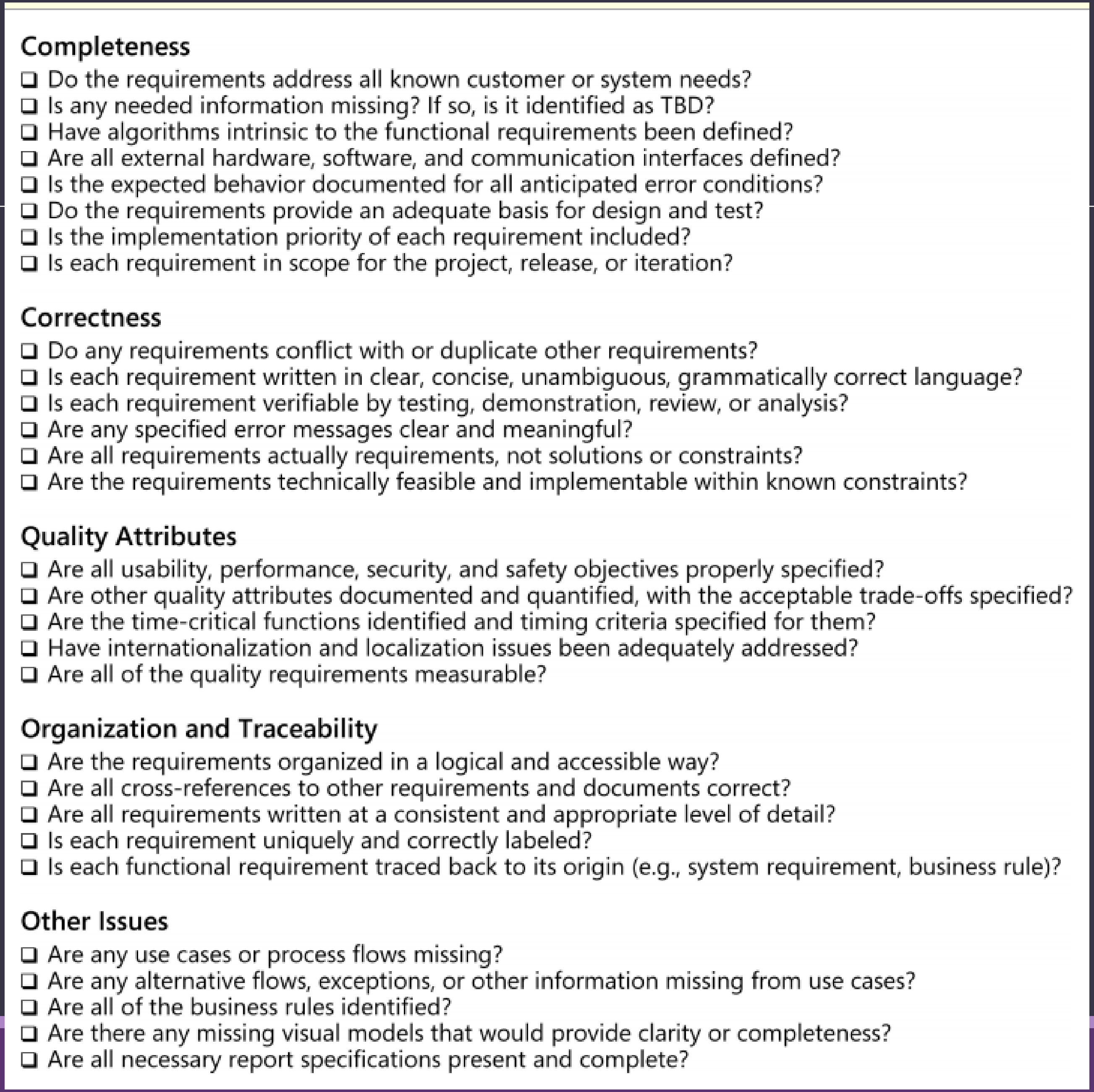
Inspection stages

An inspection is a multistep process.

You can inspect small sets of requirements at a time—perhaps those allocated to a specific development iteration—thereby eventually covering the full requirements collection.



Defect checklist



Completeness

- ❑ Do the requirements address all known customer or system needs?
- ❑ Is any needed information missing? If so, is it identified as TBD?
- ❑ Have algorithms intrinsic to the functional requirements been defined?
- ❑ Are all external hardware, software, and communication interfaces defined?
- ❑ Is the expected behavior documented for all anticipated error conditions?
- ❑ Do the requirements provide an adequate basis for design and test?
- ❑ Is the implementation priority of each requirement included?
- ❑ Is each requirement in scope for the project, release, or iteration?

Correctness

- ❑ Do any requirements conflict with or duplicate other requirements?
- ❑ Is each requirement written in clear, concise, unambiguous, grammatically correct language?
- ❑ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ❑ Are any specified error messages clear and meaningful?
- ❑ Are all requirements actually requirements, not solutions or constraints?
- ❑ Are the requirements technically feasible and implementable within known constraints?

Quality Attributes

- ❑ Are all usability, performance, security, and safety objectives properly specified?
- ❑ Are other quality attributes documented and quantified, with the acceptable trade-offs specified?
- ❑ Are the time-critical functions identified and timing criteria specified for them?
- ❑ Have internationalization and localization issues been adequately addressed?
- ❑ Are all of the quality requirements measurable?

Organization and Traceability

- ❑ Are the requirements organized in a logical and accessible way?
- ❑ Are all cross-references to other requirements and documents correct?
- ❑ Are all requirements written at a consistent and appropriate level of detail?
- ❑ Is each requirement uniquely and correctly labeled?
- ❑ Is each functional requirement traced back to its origin (e.g., system requirement, business rule)?

Other Issues

- ❑ Are any use cases or process flows missing?
- ❑ Are any alternative flows, exceptions, or other information missing from use cases?
- ❑ Are all of the business rules identified?
- ❑ Are there any missing visual models that would provide clarity or completeness?
- ❑ Are all necessary report specifications present and complete?

Prototyping requirements

Prototypes are validation tools that make the requirements real. They allow the user to experience some aspects of what a system based on the requirements would be like.

All kinds of prototypes allow you to fid missing requirements before more expensive activities like development and testing take place

paper mock-up can be used to walk through use cases, processes, or functions to detect any omitted or erroneous requirements.

Proof-of-concept prototypes can demonstrate that the requirements are feasible. Evolutionary prototypes allow the users to see how the requirements would work when they are implemented, to validate that the result is what they expect.

Testing the requirements

- Tests that are based on the functional requirements or derived from user requirements help make the expected system behaviors tangible to the project participants.
- The simple act of designing tests will reveal many problems with the requirements long before you can execute those tests on running software.
- Writing functional tests crystallizes your vision of how the system should behave under certain conditions.

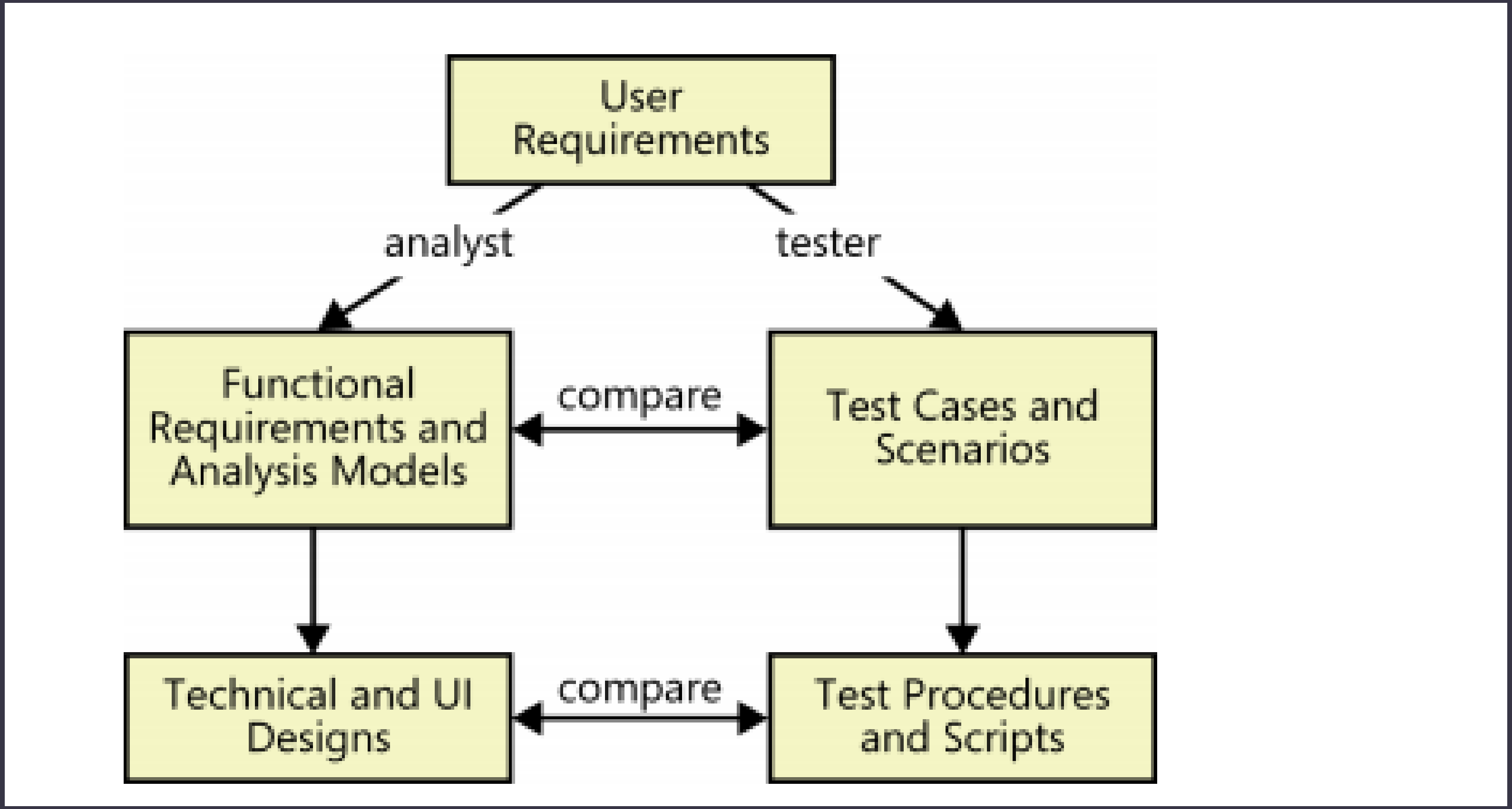
Sample Test Template

Project Name			
Test Id		Test Designed By:	
Test Title:		Test Designed Date:	
Description			
S.No	Test Steps	Test Input	Expected Results

Sample Testcase Example

Project Name	E-banking System		
Test Id	01	Test Designed By:	N.Nouman
Test Title:	Test the login functionality	Test Designed Date:	20-5-17
Description	Verify login with valid username and password		
S.No	Test Steps	Test Input	Expected Results
1	Open URL	www.cms.com	Open website
2	Enter User Name	admin1234	Invalid Login Id/Password
3	Enter Password	123334	
4	Press Submit button		Site should not login

Development and testing work products are derived from a common source



Validating requirements with acceptance criteria

- ❑Software developers might believe that they’ve built the perfect product, but the customer is the final arbiter.
- ❑Customers need to assess whether a system satisfies its predefined *acceptance criteria*. Acceptance criteria—and hence acceptance testing—should evaluate whether the product satisfies its documented requirements and whether it is fi for use in the intended operating environment
- ❑Having users devise acceptance tests is a valuable contributor to effective requirements development. The earlier that acceptance tests are written, the sooner they can help the team filter out defects in the requirements and, ultimately, in the implemented software.

Acceptance Criteria

- Working with customers to develop acceptance criteria provides a way to validate both the requirements and the solution itself
- Acceptance criteria define the minimum conditions for an application to be considered business-ready.
- Thinking about acceptance criteria offers a shift in perspective from the elicitation question of "What do you need to do with the system?" to "How would you judge whether the solution meets your needs?" Encourage users to use the SMART mnemonic—*Specific*, *Measurable*, *Attainable*, *Relevant*, and *Time-sensitive*—when defining acceptance criteria

Acceptance tests

Acceptance tests constitute the largest portion of the acceptance criteria.

Creators of acceptance tests should consider the most commonly performed and most important usage scenarios when deciding how to evaluate the software's acceptability. Focus on testing the normal flows of the use cases and their corresponding exceptions, devoting less attention to the less frequently used alternative flows.

offers a wealth of guidance for writing requirements-based acceptance tests.