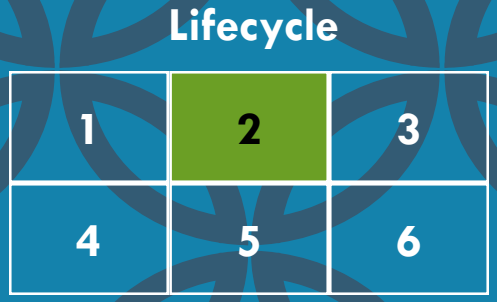


ISTQB / ISEB Foundation Exam Practice

CONTENTS

Software Development Models

The **software development models** are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals.

There are many development life cycle models that have been developed in order to achieve different required objectives.

The models specify the various stages of the process and the order in which they are carried out. The selection of model has very high impact on the testing that is carried out.

It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

Types of Software development models

1. Waterfall model
2. V model
3. Incremental model
4. RAD model
5. Agile model
6. Iterative model
7. Spiral model
8. Prototype model

VV&T

Verification

- the process of evaluating a system or component to determine whether the products of the given development phase satisfy the conditions imposed at the start of that phase
- Are we building the product right?

Validation

- determination of the correctness of the products of software development with respect to the user needs and requirements
- Are we building the right product?

- It is entirely possible that a product passes when verified but fails when validated. This can happen when, say, a product is built as per the specifications but the specifications themselves fail to address the user's needs.

Waterfall Model

The Waterfall Model was first Process Model to be introduced.

It is also referred to as a **linear-sequential life cycle model**.

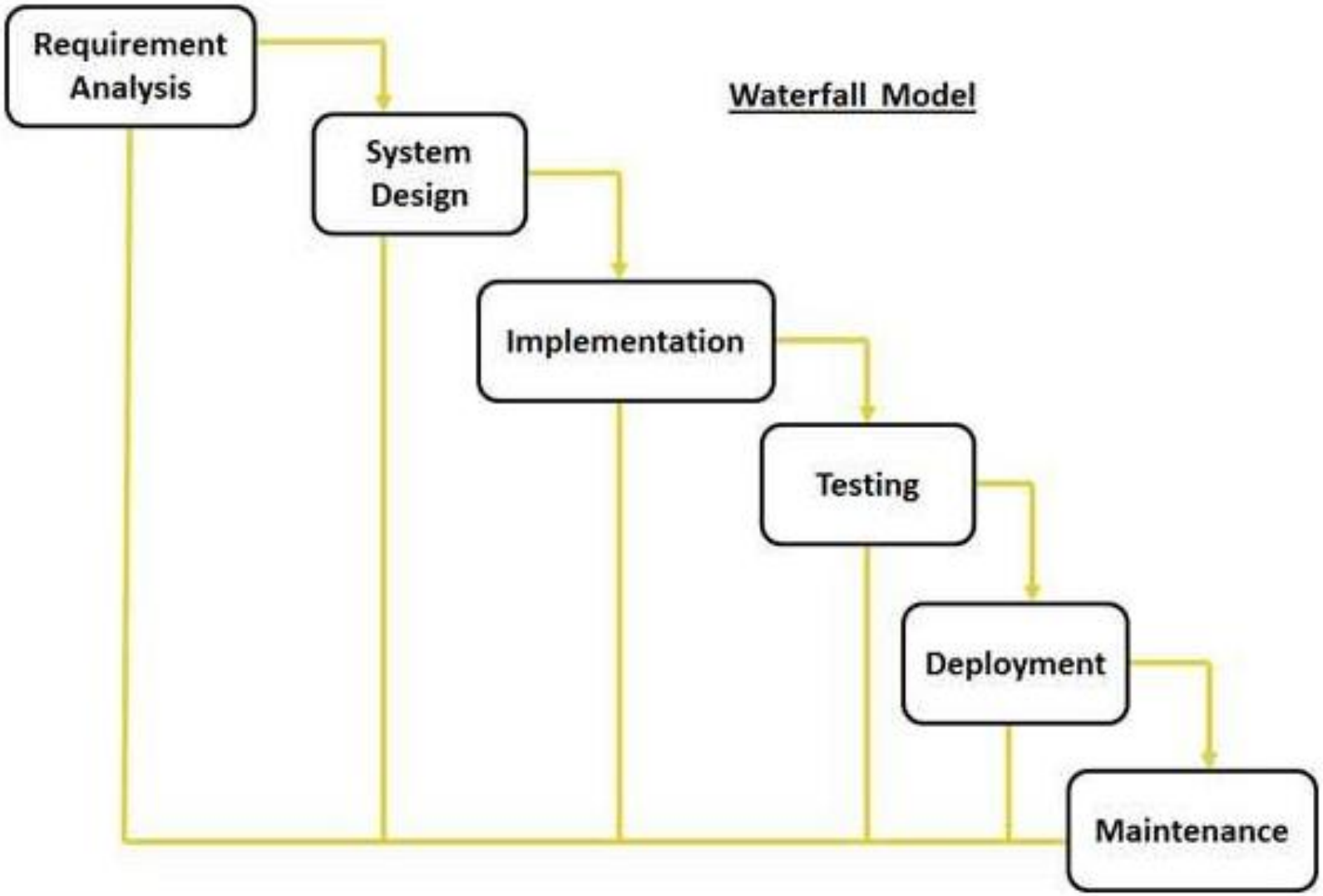
It is very simple to understand and use.

In a waterfall model, each phase must be completed fully before the next phase can begin.

This type of model is basically used for the for the project which is small and there are no uncertain requirements.

At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.

In this model the testing starts only after the development is complete.
In **waterfall model phases** do not overlap.


Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

Advantages of waterfall model

This model is simple and easy to understand and use.

It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.

In this model phases are processed and completed one at a time. Phases do not overlap.

Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model

Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.

No working software is produced until late during the life cycle.

High amounts of risk and uncertainty.

Not a good model for complex and object-oriented projects.

Poor model for long and ongoing projects.

Not suitable for the projects where requirements are at a moderate to high risk of changing.

V-model

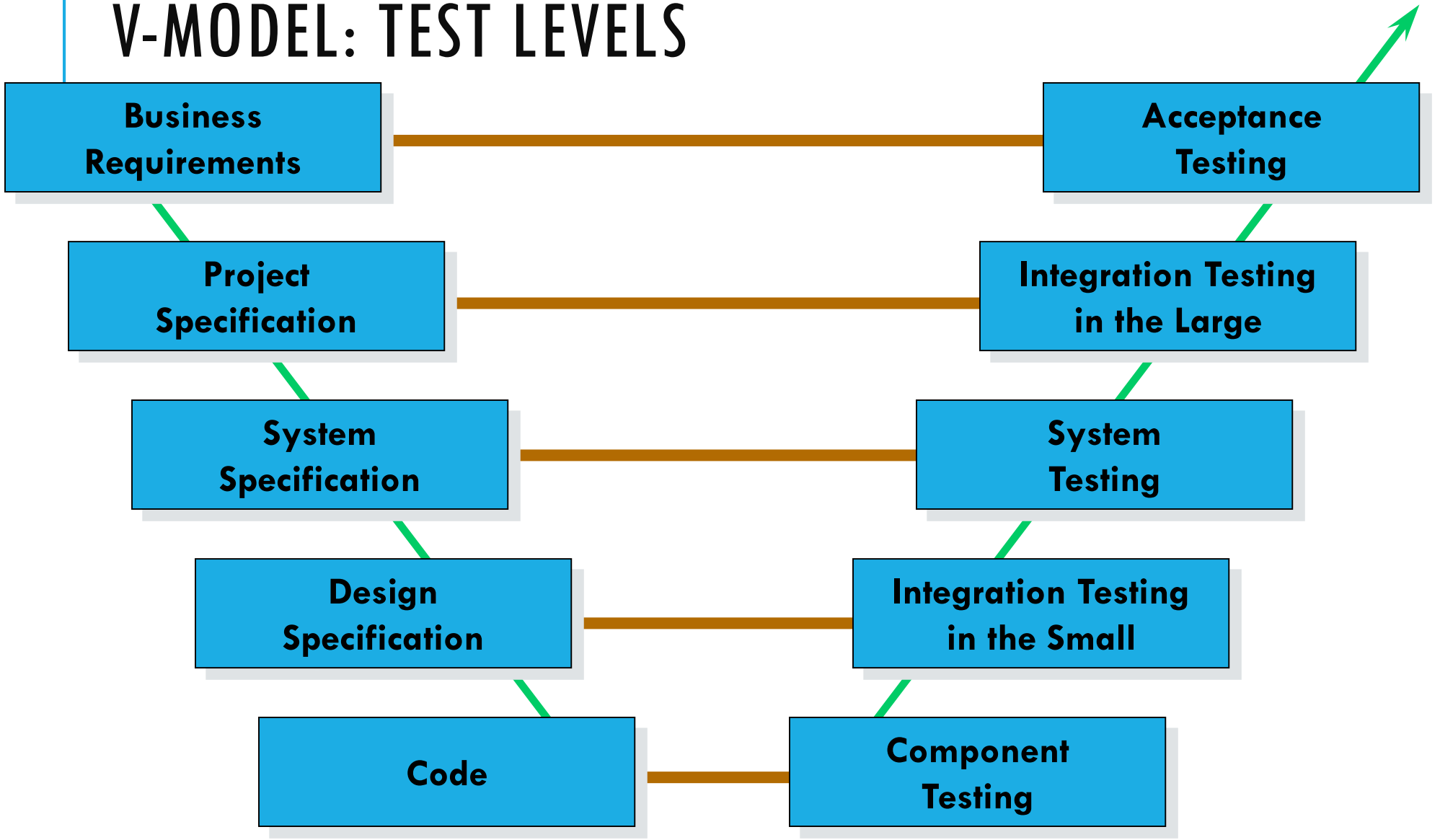
V- model means Verification and Validation model.

Just like the **waterfall model**, the V-Shaped life cycle is a sequential path of execution of processes.

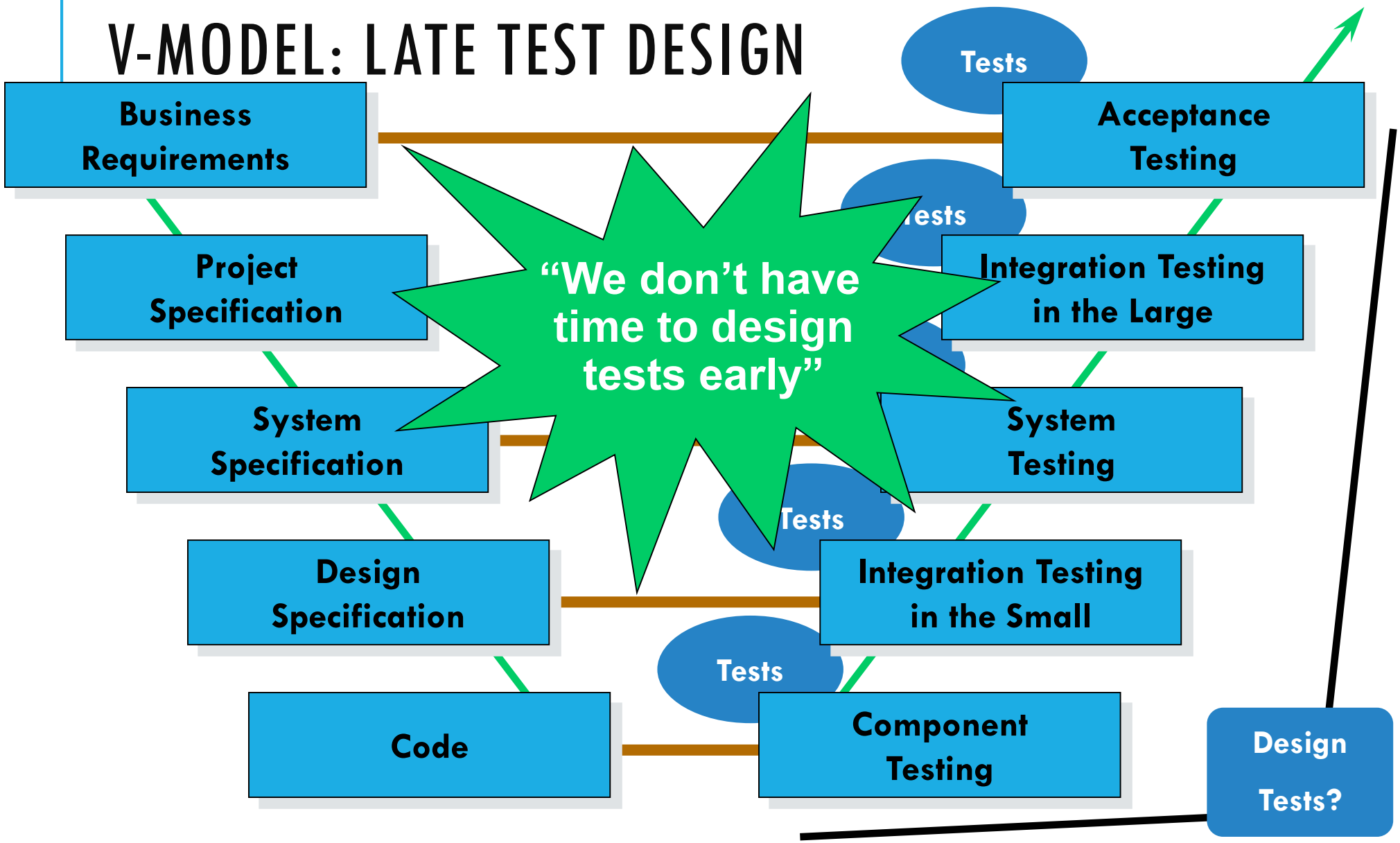
Each phase must be completed before the next phase begins.

Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

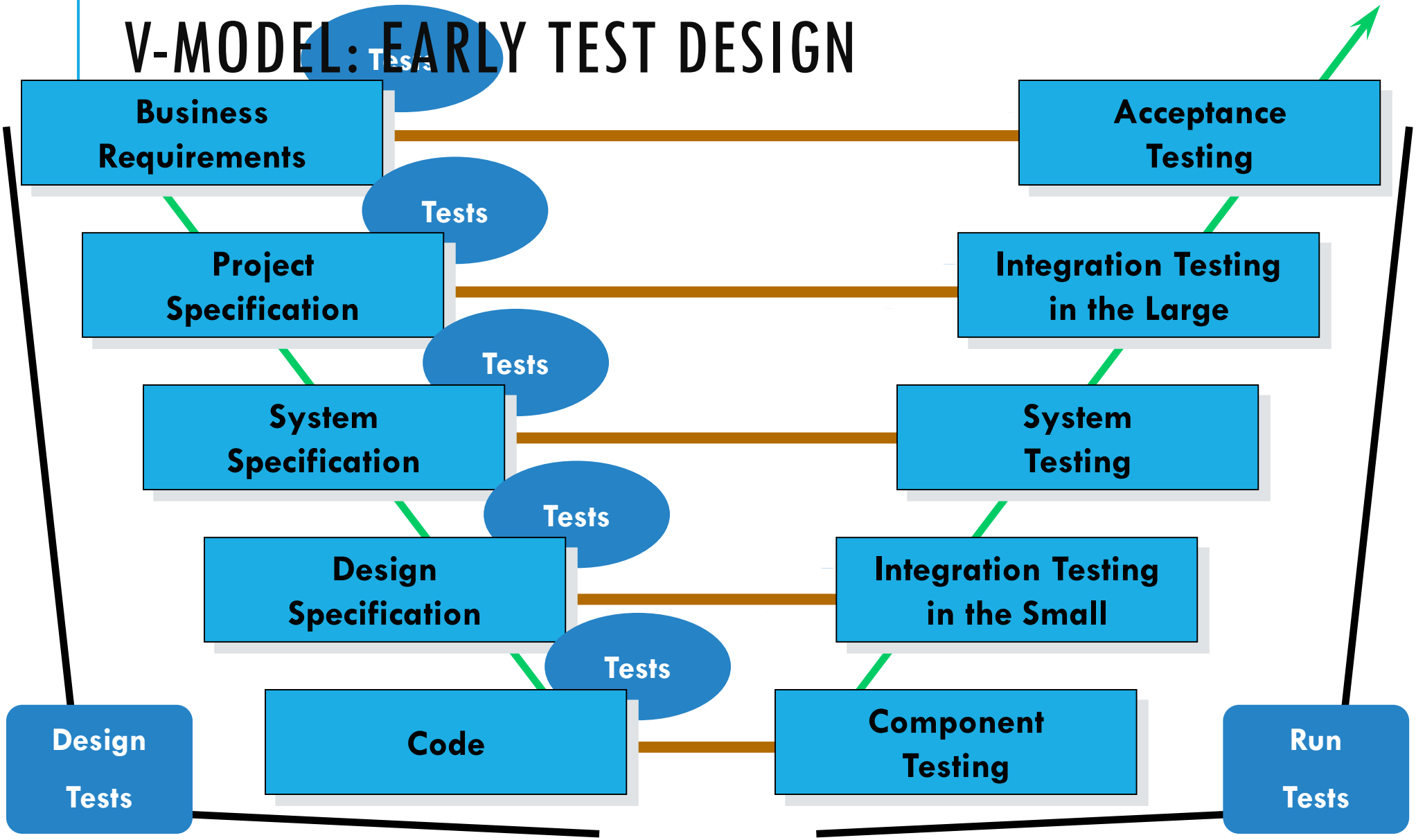
V-MODEL: TEST LEVELS



V-MODEL: LATE TEST DESIGN



V-MODEL: EARLY TEST DESIGN



Advantages of V-model

Simple and easy to use.

Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.

Proactive defect tracking – that is defects are found at early stage.

Avoids the downward flow of the defects.

Works well for small projects where requirements are easily understood.

Disadvantages of V-model

Very rigid and least flexible.

Software is developed during the implementation phase, so no early prototypes of the software are produced.

If any changes happen in midway, then the test documents along with requirement documents has to be updated.

Incremental model

In incremental model the whole requirement is divided into various builds.

Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle.

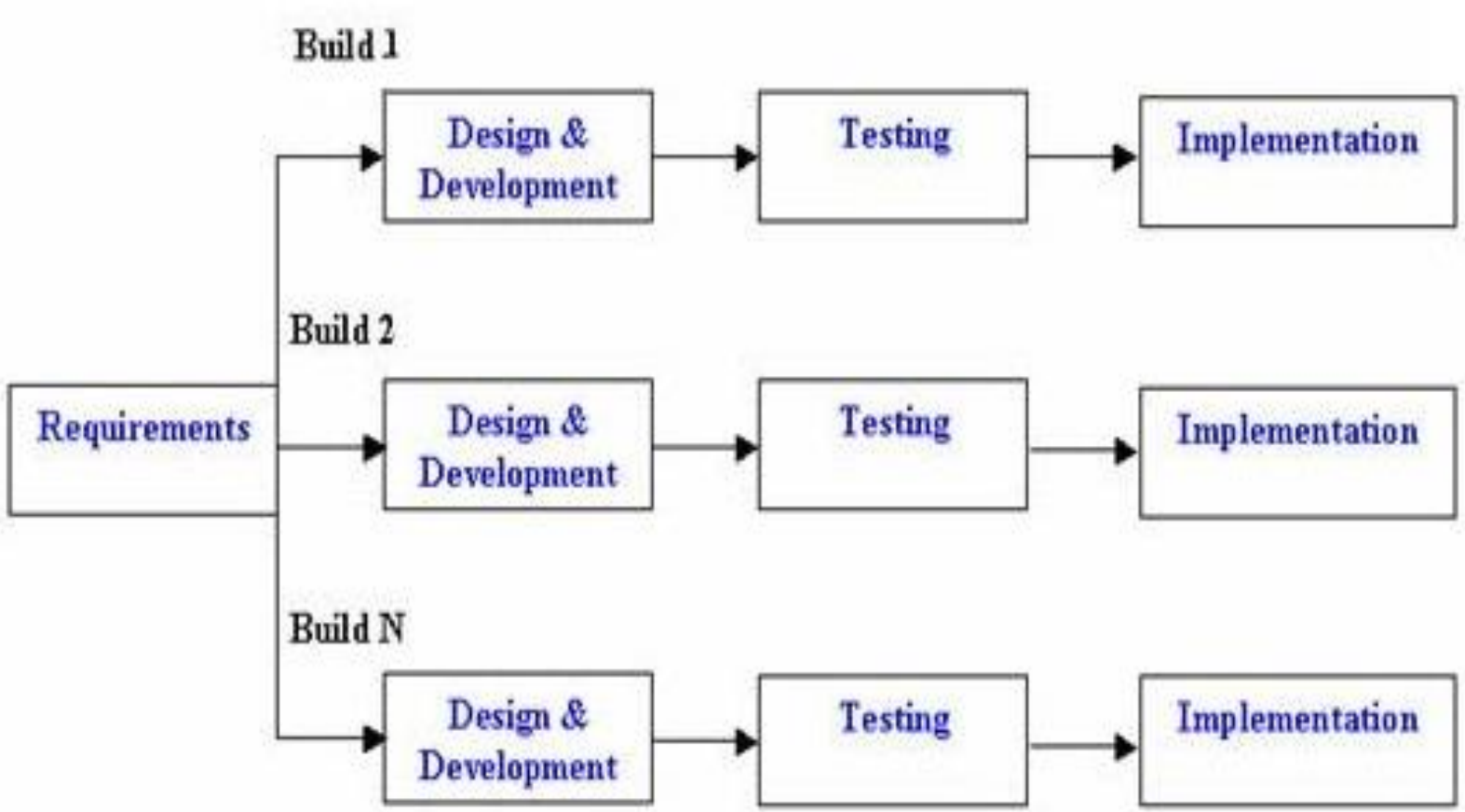
Cycles are divided up into smaller, more easily managed modules.

Each module passes through the requirements, design, implementation and testing phases.

A working version of software is produced during the first module, so you have working software early on during the software life cycle.

Each subsequent release of the module adds function to the previous release.

The process continues till the complete system is achieved.



Incremental Life Cycle Model

Advantages of Incremental model

Generates working software quickly and early during the software life cycle.

This model is more flexible – less costly to change scope and requirements.

It is easier to test and debug during a smaller iteration.

In this model customer can respond to each built.

Lowers initial delivery cost.

Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model

Needs good planning and design.

Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

Total cost is higher than waterfall.

RAD model

RAD model is Rapid Application Development model.

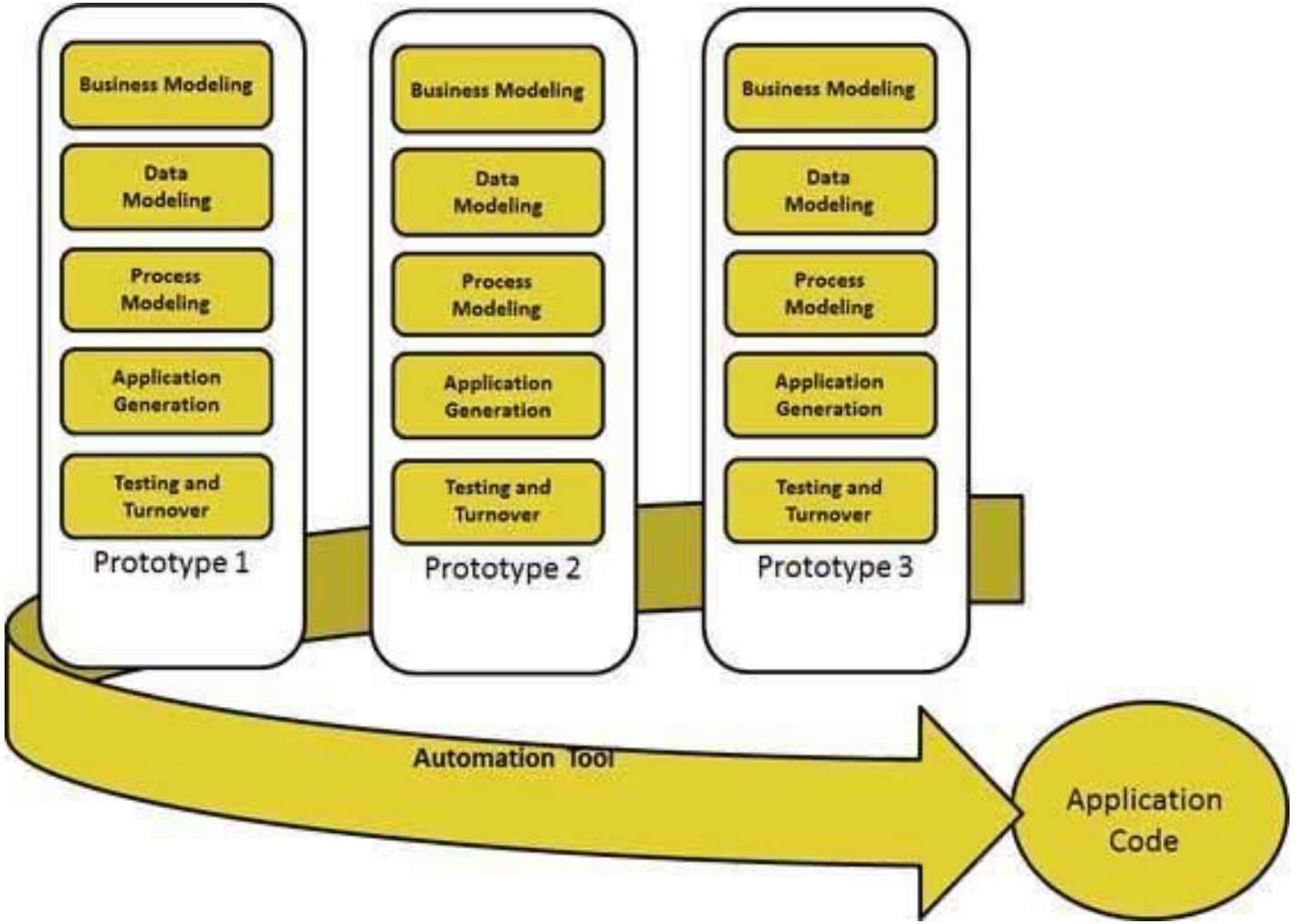
It is a type of incremental model.

In RAD model the components or functions are developed in parallel as if they were mini projects.

The developments are time boxed, delivered and then assembled into a working prototype.

This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Following image illustrates the RAD Model:



Advantages of the RAD model

Reduced development time.

Increases reusability of components

Quick initial reviews occur

Encourages customer feedback

Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model

Depends on strong team and individual performances for identifying business requirements.

Only system that can be modularized can be built using RAD

Requires highly skilled developers/designers.

High dependency on modeling skills

Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

Agile Model

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

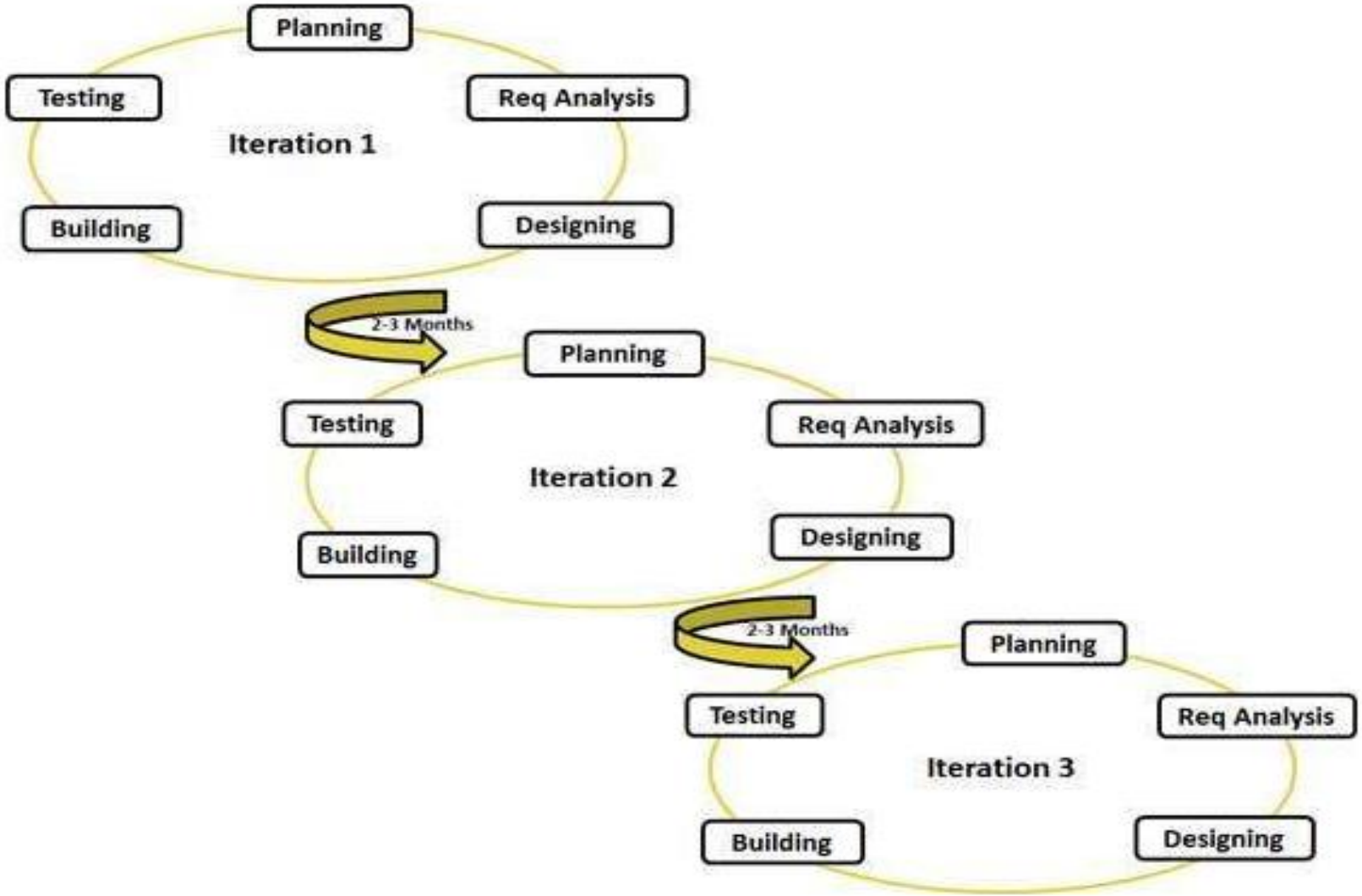
Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

At the end of the iteration a working product is displayed to the customer and important stakeholders.

What is Agile?

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing

Advantages of Agile Model

1. Is a very realistic approach to software development
2. Promotes teamwork and cross training.
3. Functionality can be developed rapidly and demonstrated.
4. Resource requirements are minimum.
5. Suitable for fixed or changing requirements
6. Delivers early partial working solutions.
7. Good model for environments that change steadily.
8. Minimal rules, documentation easily employed.
9. Enables concurrent development and delivery within an overall planned context.
10. Little or no planning required
11. Easy to manage
12. Gives flexibility to developers

Disadvantages of Agile Model

Not suitable for handling complex dependencies.

More risk of sustainability, maintainability and extensibility.

An overall plan, an agile leader and agile PM practice is a must without which it will not work.

Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.

There is very high individual dependency, since there is minimum documentation generated.

Transfer of technology to new team members may be quite challenging due to lack of documentation.

(BEFORE PLANNING FOR A SET OF TESTS)

set organisational test strategy

identify people to be involved (sponsors, testers, QA, development, support, etc.)

examine the requirements or functional specifications (test basis)

set up the test organisation and infrastructure

defining test deliverables & reporting structure

HIGH LEVEL TEST PLANNING

What is the purpose of a high level test plan?

- Who does it communicate to?
- Why is it a good idea to have one?

What information should be in a high level test plan?

- What is your standard for contents of a test plan?
- Have you ever forgotten something important?
- What is not included in a test plan?

TEST PLAN 1

1 Test Plan Identifier

2 Introduction

- software items and features to be tested
- references to project authorisation, project plan, QA plan, relevant policies & standards

3 Test items

- test items including version/revision level
- how transmitted (net, disc, CD, etc.)
- references to software documentation

TEST PLAN 2

4 Features to be tested

- identify test design specification / techniques

5 Features not to be tested

- reasons for exclusion

TEST PLAN 3

6 Approach

- activities, techniques and tools
- detailed enough to estimate
- specify degree of comprehensiveness (e.g. coverage) and other completion criteria (e.g. faults)
- identify constraints (environment, staff, deadlines)

7 Item Pass/Fail Criteria

8 Suspension criteria and resumption criteria

- for all or parts of testing activities
- which activities must be repeated on resumption

TEST PLAN 4

9 Test Deliverables

- Test plan
- Test design specification
- Test case specification
- Test procedure specification
- Test item transmittal reports
- Test logs
- Test incident reports
- Test summary reports

TEST PLAN 5

10 Testing tasks

- including inter-task dependencies & special skills

11 Environment

- physical, hardware, software, tools
- mode of usage, security, office space

12 Responsibilities

- to manage, design, prepare, execute, witness, check, resolve issues, providing environment, providing the software to test

TEST PLAN 6

13 Staffing and Training Needs

14 Schedule

- test milestones in project schedule
- item transmittal milestones
- additional test milestones (environment ready)
- what resources are needed when

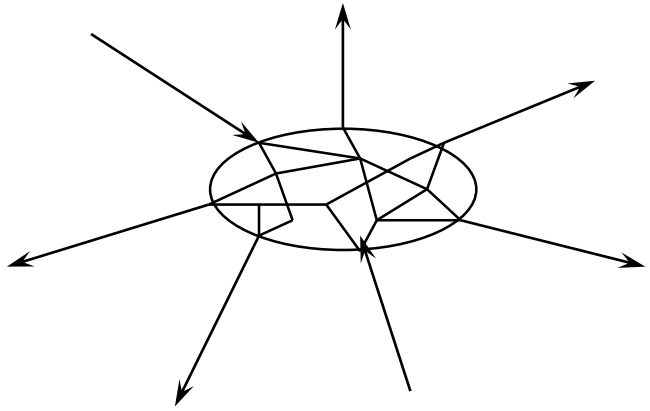
15 Risks and Contingencies

- contingency plan for each identified risk

16 Approvals

- names and when approved

COMPONENT TESTING



lowest level

tested in isolation

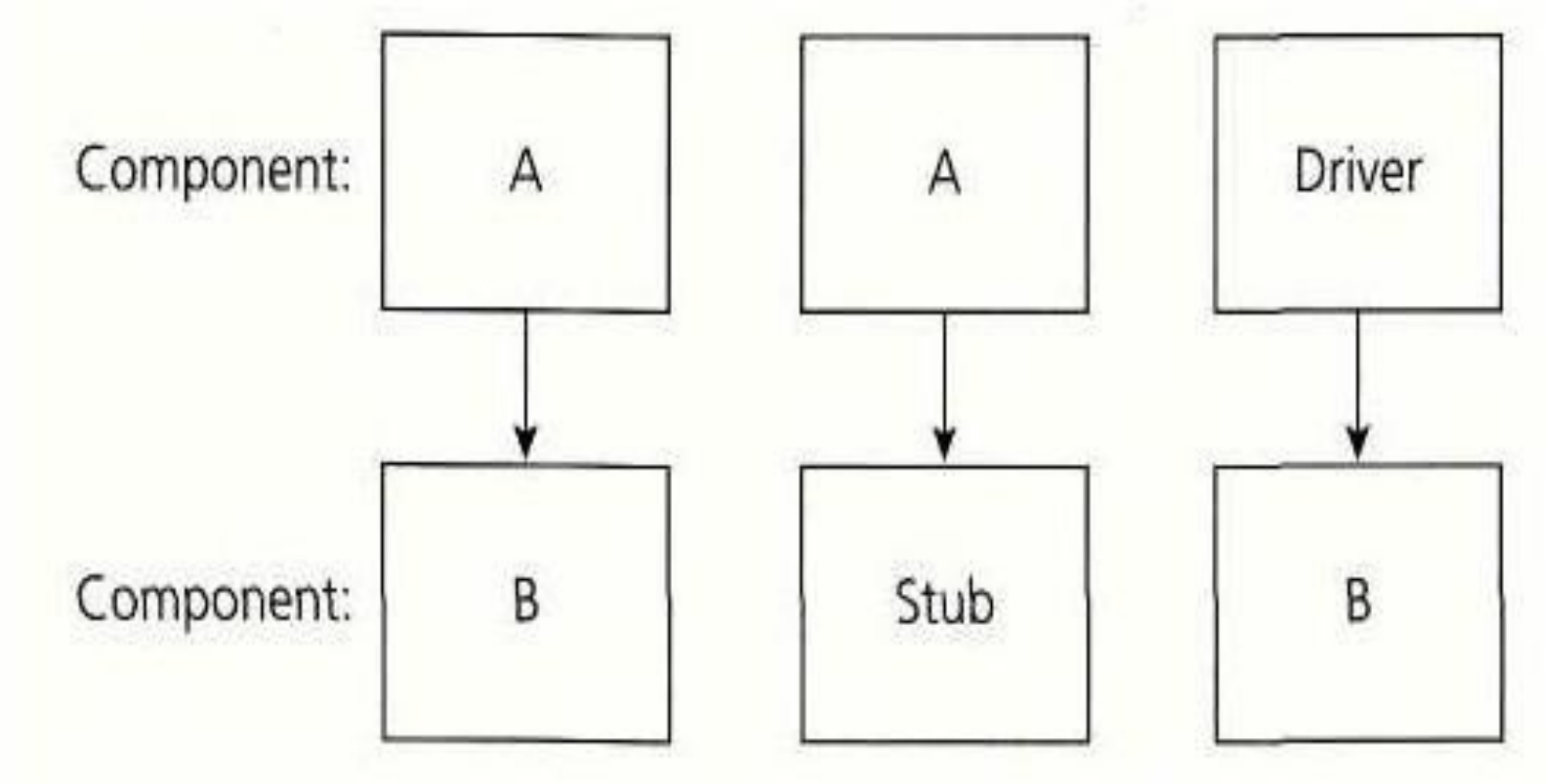
most thorough look at detail

- error handling
- interfaces

usually done by programmer

also known as unit, module, program testing

diagram of the component testing



EARLY TEST DESIGN

test design finds faults

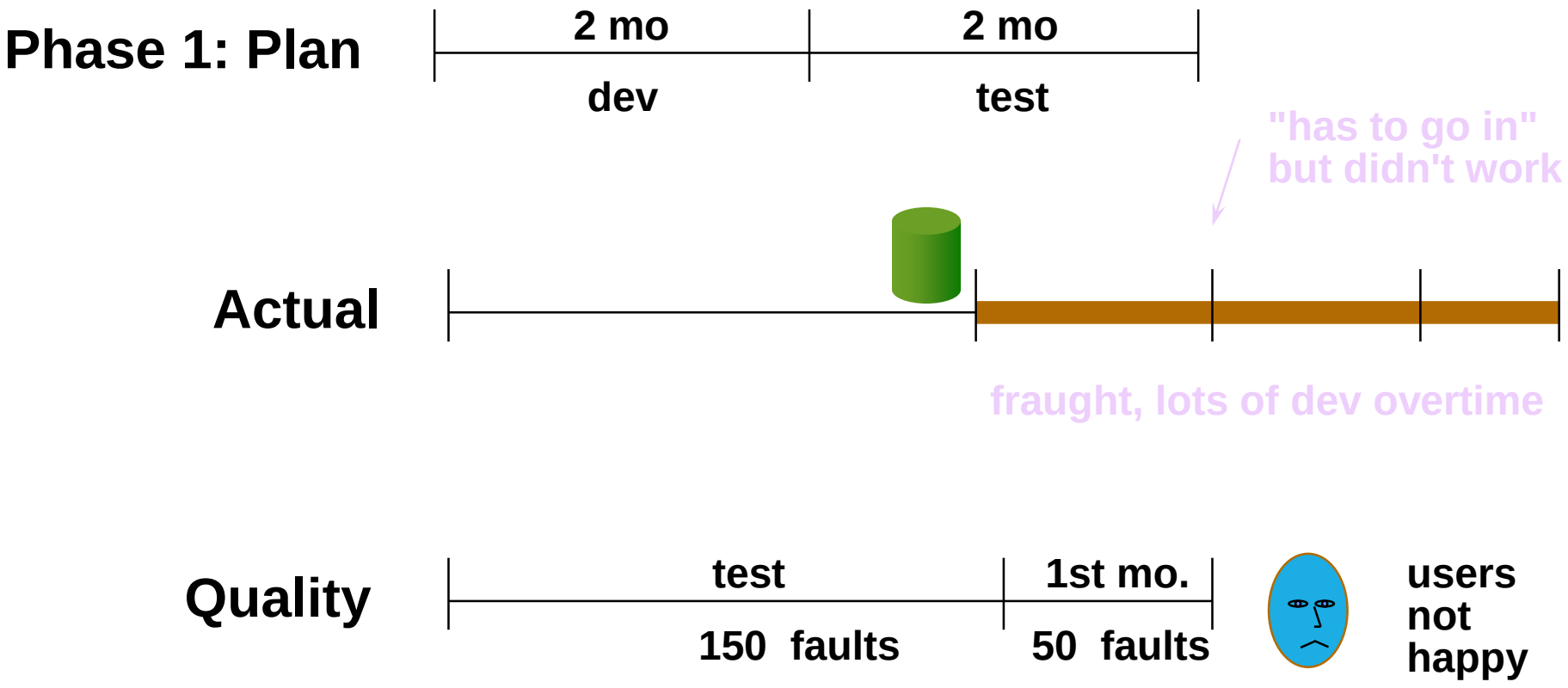
faults found early are cheaper to fix

most significant faults found first

no additional effort, re-schedule test design

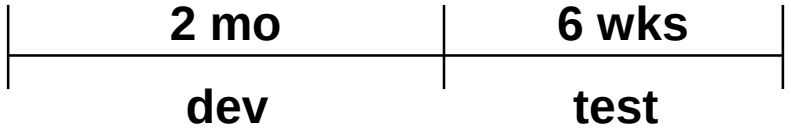
Early test design helps to build quality, stops fault multiplication

EXPERIENCE REPORT: PHASE 1

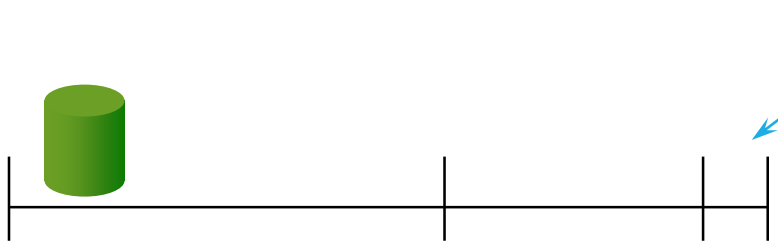


EXPERIENCE REPORT: PHASE 2

Phase 2: Plan



Actual



on time

smooth, not much for dev to do

Quality

happy users!

"TESTING IS EXPENSIVE"

Compared to what?

What is the cost of NOT testing, or of faults missed that should have been found in test?

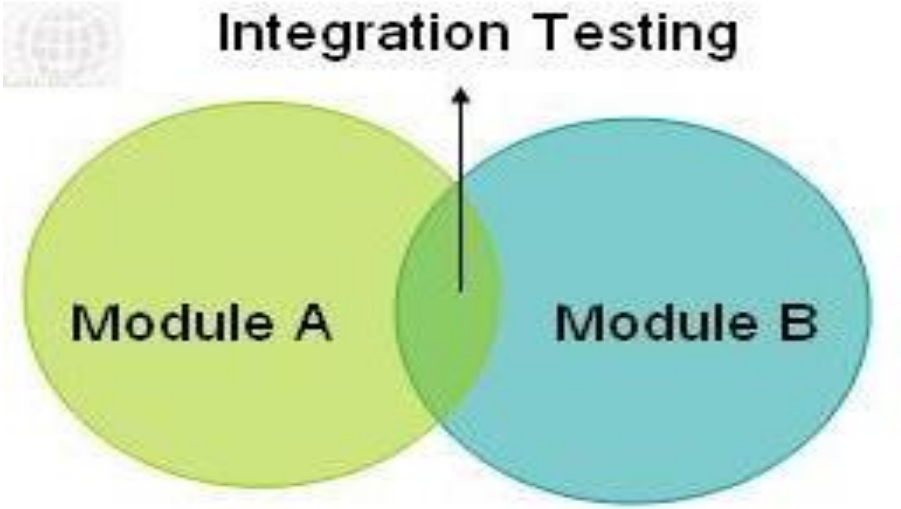
- Cost to fix faults goes higher the later the fault is found
- Poor quality software costs more to use
  - users take more time to understand what to do
  - users make more mistakes in using it
  - morale suffers
  - => lower productivity

Do you know what it costs your organisation?

INTEGRATION TESTING

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

Also after integrating two different components together we do the integration testing. As displayed in the image below when two different modules 'Module A' and 'Module B' are integrated then the integration testing is done.



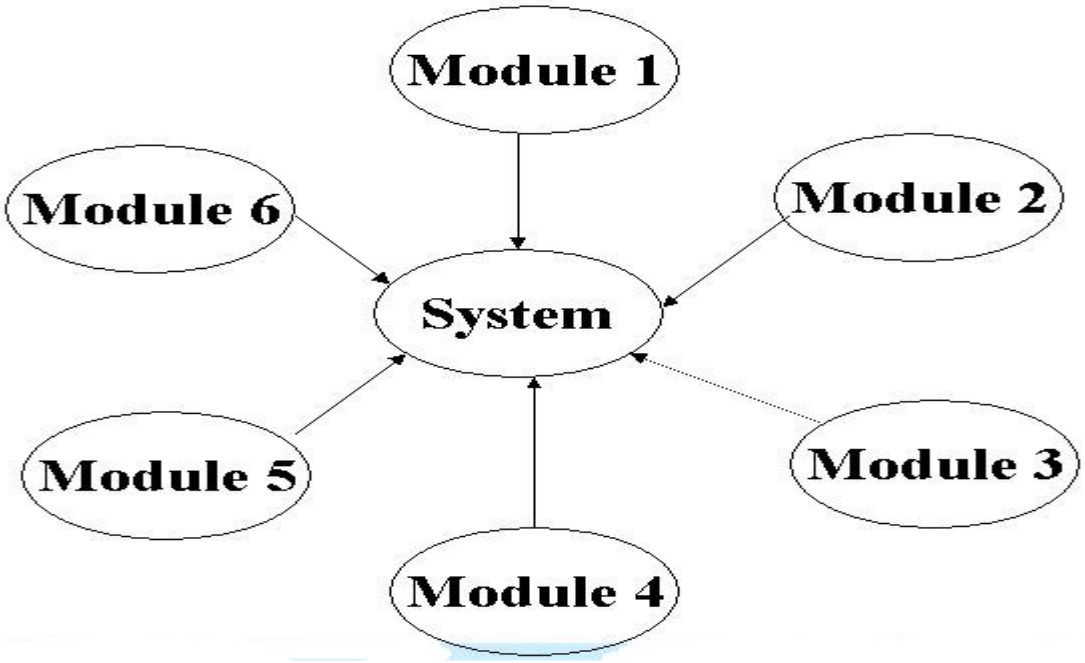
integration testing techniques:

1. **Big Bang integration testing**
2. **Top-down integration testing**
3. **Bottom-up integration testing:**

BIG-BANG INTEGRATION

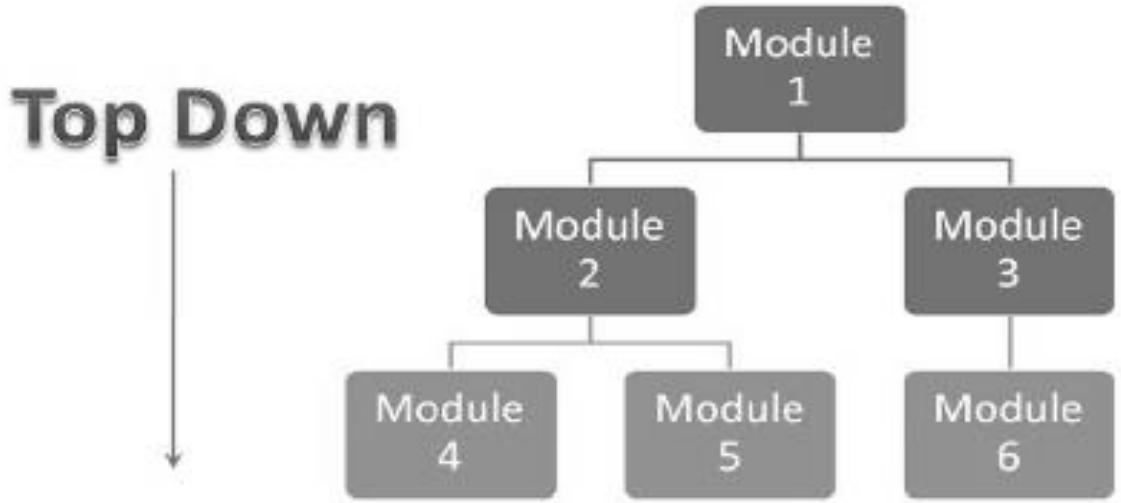
In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.

As per the below image all the modules from 'Module 1' to 'Module 6' are integrated simultaneously then the testing is carried out.



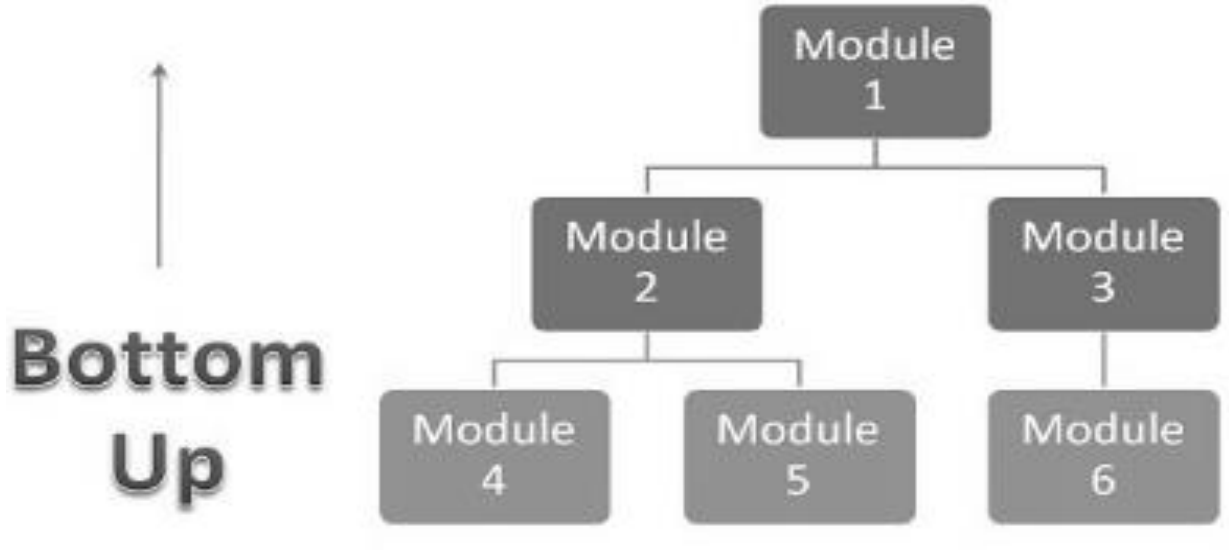
Top-down integration testing

Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs. Below is the diagram of 'Top down Approach':



Bottom-up integration testing

Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. Below is the image of 'Bottom up approach':



SYSTEM TESTING

In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.

It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.

System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.

System testing is carried out by specialists testers or independent testers.

System testing should investigate both functional and non-functional requirements of the testing.

FUNCTIONAL SYSTEM TESTING

Functional requirements

- a requirement that specifies a function that a system or system component must perform

Functional specification

- the document that describes in detail the characteristics of the product with regard to its intended capability

REQUIREMENTS-BASED TESTING

Uses specification of requirements as the basis for identifying tests

- table of contents of the requirements spec provides an initial test inventory of test conditions
- for each section / paragraph / topic / functional area,
  - risk analysis to identify most important / critical
  - decide how deeply to test each functional area

BUSINESS PROCESS-BASED TESTING

Expected user profiles

- what will be used most often?
- what is critical to the business?

Business scenarios

- typical business transactions (birth to death)

Use cases

- prepared cases based on real situations

NON-FUNCTIONAL SYSTEM TESTING

different types of non-functional system tests:

- usability
- security
- documentation
- storage
- volume

- configuration / installation
- reliability / qualities
- back-up / recovery
- performance, load, stress

PERFORMANCE TESTS

Timing Tests

- response and service times
- database back-up times

Capacity & Volume Tests

- maximum amount or processing rate
- number of records on the system
- graceful degradation

Endurance Tests (24-hr operation?)

- robustness of the system
- memory allocation

MULTI-USER TESTS

Concurrency Tests

- small numbers, large benefits
- detect record locking problems

Load Tests

- the measurement of system behaviour under realistic multi-user load

Stress Tests

- go beyond limits for the system - know what will happen
- particular relevance for e-commerce

USABILITY TESTS

messages tailored and meaningful to (real) users?

coherent and consistent interface?

sufficient redundancy of critical information?

feedback (wait messages)?

clear mappings (how to escape)?

Who should design / perform these tests?

SECURITY TESTS

passwords

encryption

hardware permission devices

levels of access to information

authorisation

covert channels

physical security

CONFIGURATION AND INSTALLATION

Configuration Tests

- different hardware or software environment
- configuration of the system itself
- upgrade paths - may conflict

Installation Tests

- distribution (CD, network, etc.) and timings
- physical aspects: electromagnetic fields, heat, humidity, motion, chemicals, power supplies
- uninstall (removing installation)

RELIABILITY / QUALITIES

Reliability

- "system will be reliable" - how to test this?
- "2 failures per year over ten years"
- Mean Time Between Failures (MTBF)
- reliability growth models

Other Qualities

- maintainability, portability, adaptability, etc.

BACK-UP AND RECOVERY

Back-ups

- computer functions
- manual procedures (where are tapes stored)

Recovery

- real test of back-up
- manual procedures unfamiliar
- should be regularly rehearsed
- documentation should be detailed, clear and thorough

DOCUMENTATION TESTING

Documentation review

- check for accuracy against other documents
- gain consensus about content
- documentation exists, in right format

Documentation tests

- is it usable? does it work?
- user manual
- maintenance documentation

USER ACCEPTANCE TESTING

Final stage of validation

- customer (user) should perform or be closely involved
- customer can perform any test they wish, usually based on their business processes
- final user sign-off

Approach

- mixture of scripted and unscripted testing
- 'Model Office' concept sometimes used

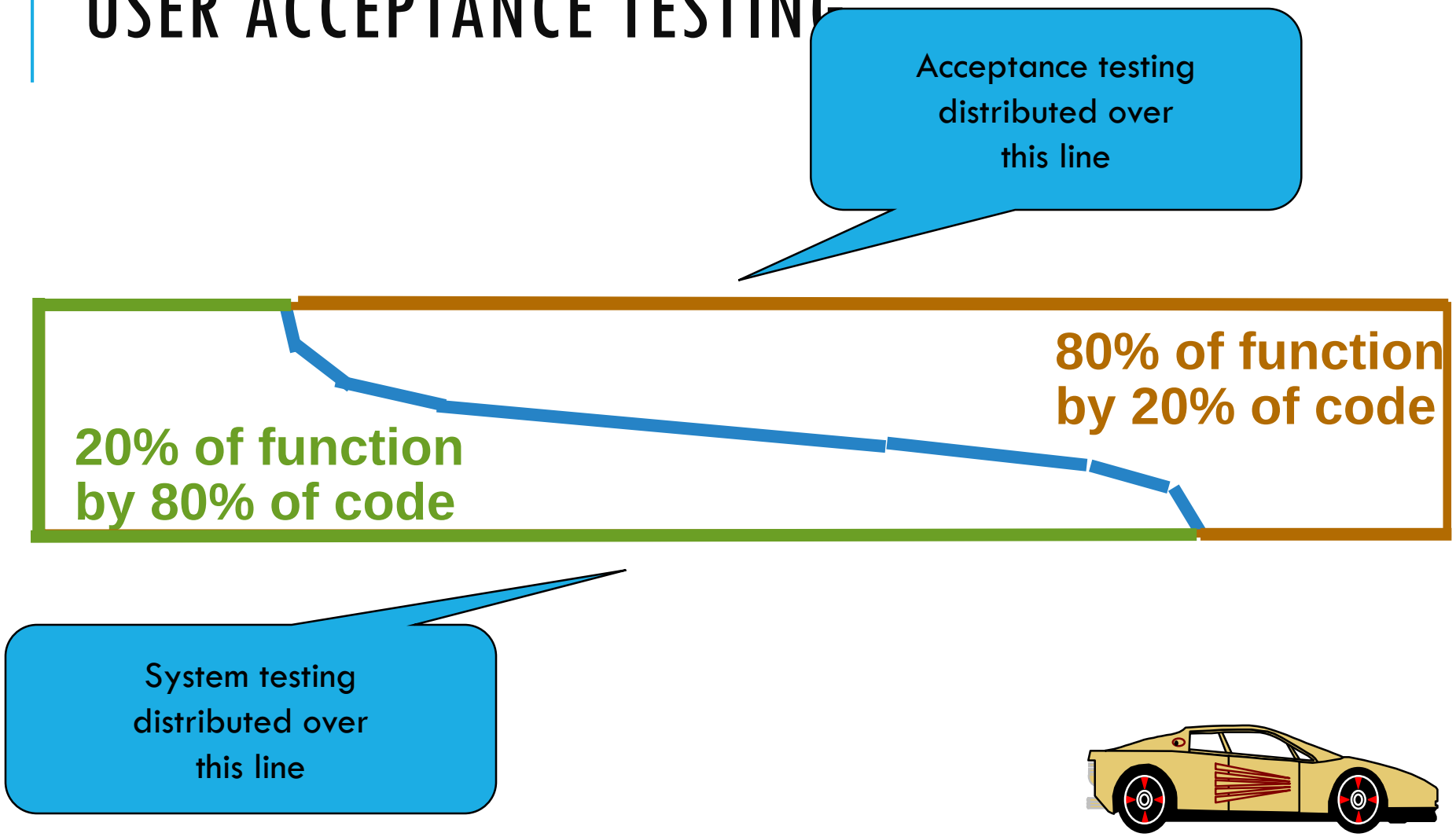
WHY CUSTOMER / USER INVOLVEMENT

Users know:

- what really happens in business situations
- complexity of business relationships
- how users would do their work using the system
- variants to standard tasks (e.g. country-specific)
- examples of real cases
- how to identify sensible work-arounds

Benefit: detailed understanding of the new system

USER ACCEPTANCE TESTING



CONTRACT ACCEPTANCE TESTING

Contract to supply a software system

- agreed at contract definition stage
- acceptance criteria defined and agreed
- may not have kept up to date with changes

Contract acceptance testing is against the contract and any documented agreed changes

- not what the users wish they had asked for!
- this system, not wish system

ALPHA AND BETA TESTS: SIMILARITIES

Testing by [potential] customers or representatives of your market

- not suitable for bespoke software

When software is stable

Use the product in a realistic way in its operational environment

Give comments back on the product

- faults found
- how the product meets their expectations
- improvement / enhancement suggestions?

ALPHA AND BETA TESTS: DIFFERENCES

Alpha testing

- simulated or actual operational testing at an in-house site not otherwise involved with the software developers (i.e. developers' site)

Beta testing

operational testing at a site not otherwise involved with the software developers (i.e. testers' site, their own location)

ACCEPTANCE TESTING MOTTO

If you don't have patience to test the system

the system will surely test your patience

MAINTENANCE TESTING

Testing to preserve quality:

- different sequence
  - development testing executed bottom-up
  - maintenance testing executed top-down
  - different test data (live profile)
- breadth tests to establish overall confidence
- depth tests to investigate changes and critical areas
- predominantly regression testing

WHAT TO TEST IN MAINTENANCE TESTING

Test any new or changed code

Impact analysis

- what could this change have an impact on?
- how important is a fault in the impacted area?
- test what has been affected, but how much?
  - most important affected areas?
  - areas most likely to be affected?
  - whole system?

The answer: “It depends”

POOR OR MISSING SPECIFICATIONS

Consider what the system should do

- talk with users

Document your assumptions

- ensure other people have the opportunity to review them

Improve the current situation

- document what you do know and find out

Track cost of working with poor specifications

- to make business case for better specifications

WHAT SHOULD THE SYSTEM DO?

Alternatives

- the way the system works now must be right (except for the specific change) - use existing system as the baseline for regression tests
- look in user manuals or guides (if they exist)
- ask the experts - the current users

Without a specification, you cannot really test, only explore. You can validate, but not verify.