Software Project Management
Fifth Edition

Chapter 13.2

Software process quality

BS EN ISO 9001:2000 and quality management systems

ISO 9001 is one of a family of standards that specify the characteristics of a good quality management system (QMS)

Can be applied to the creation of any type of product or service, not just IT and software

Does NOT set universal product/service standards

DOES specify the way in which standards are established and monitored

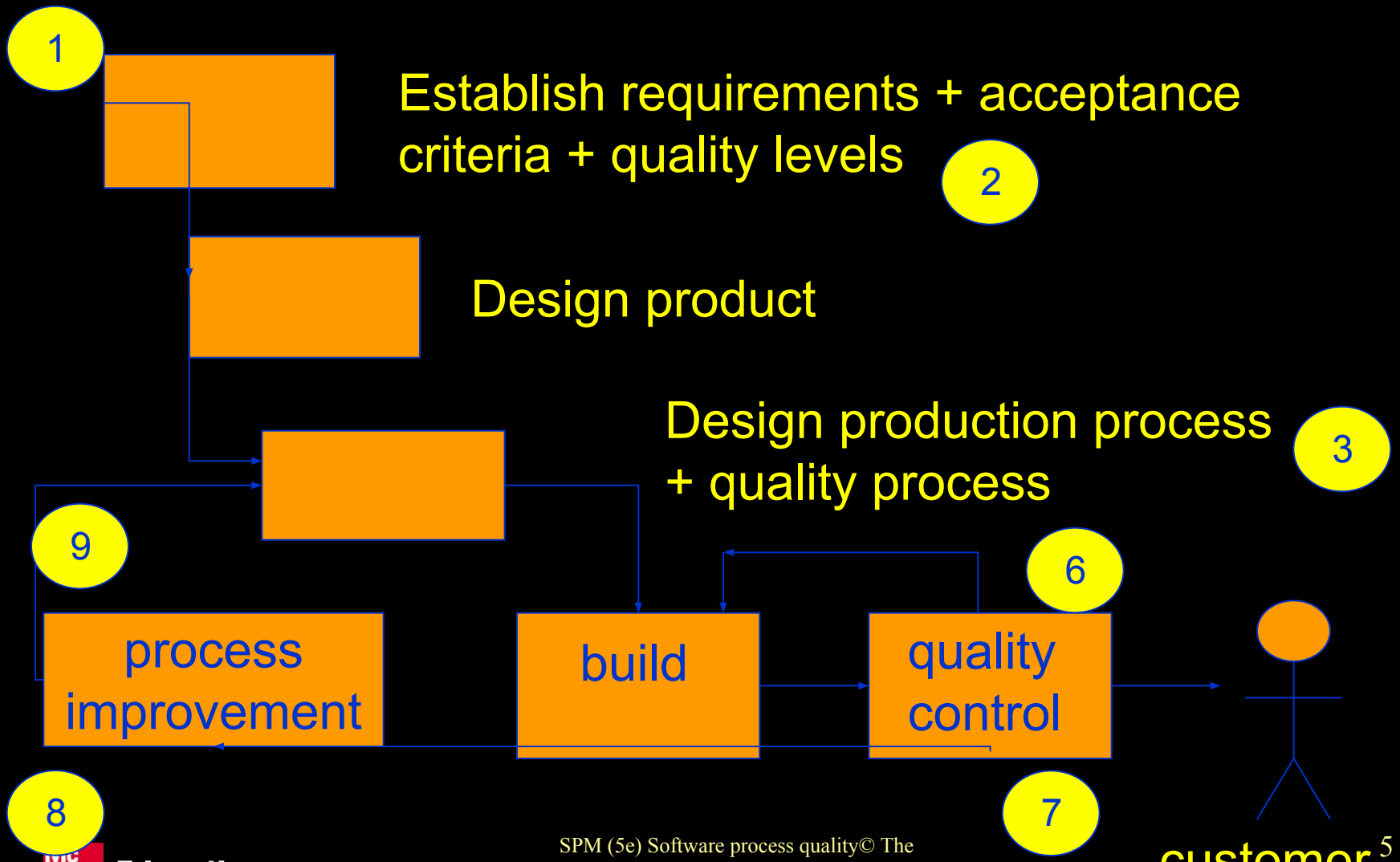
ISO 9001:2000 principles

1. Understanding the requirements of the customer
2. Leadership to provide unity of purpose and direction to achieve quality
3. Involvement of staff at all levels
4. Focus on the individual which create intermediate and deliverable products and services

ISO 9001:2000 principles

5. Focus on interrelation of processes that deliver products and services
6. Continuous process improvement
7. Decision-making based on factual evidence
8. Mutually beneficial relationships with suppliers

ISO 9001:2000 cycle



The need to improve

can everything be improved at one?

no, must tackle the most important things first

'poor companies are poor at changing'

some later improvements build on earlier ones

but there are problems

- *improvement takes up time and money*
- *'improvement' may simply be more bureaucracy!*

Capability maturity model (CMM)

created by Software Engineering Institute, Carnegie Mellon University

CMM developed by SEI for US government to help procurement

Watts S. Humphrey ‘Managing the software process’ Addison Wesley

Assessment is by questionnaire and interview

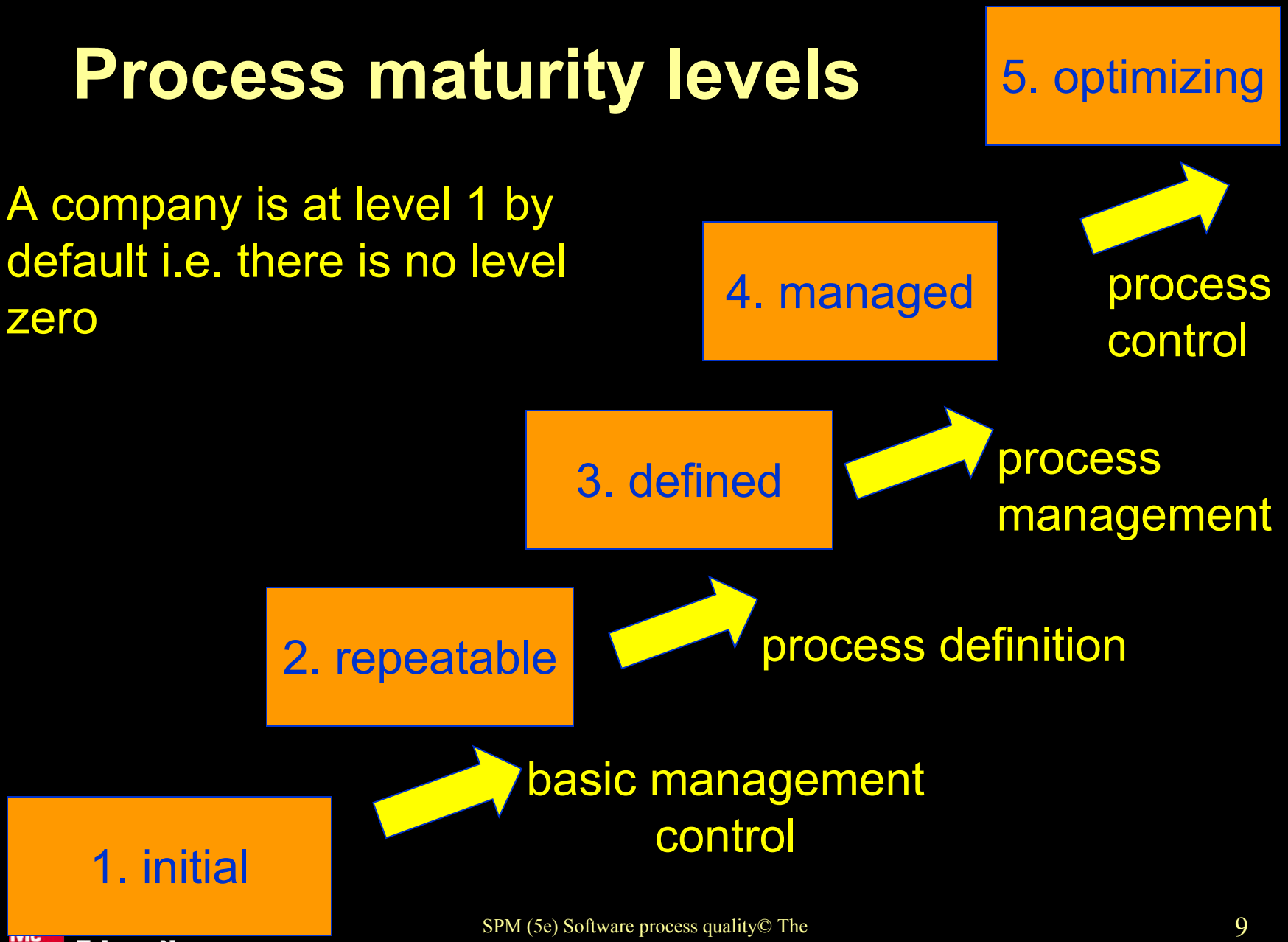
Capability maturity model 2

Different versions have been developed for different environments e.g. software engineering

New version CMMI tries to set up a generic model which can be populated differently for different environments

Process maturity levels

A company is at level 1 by default i.e. there is no level zero



Key process areas

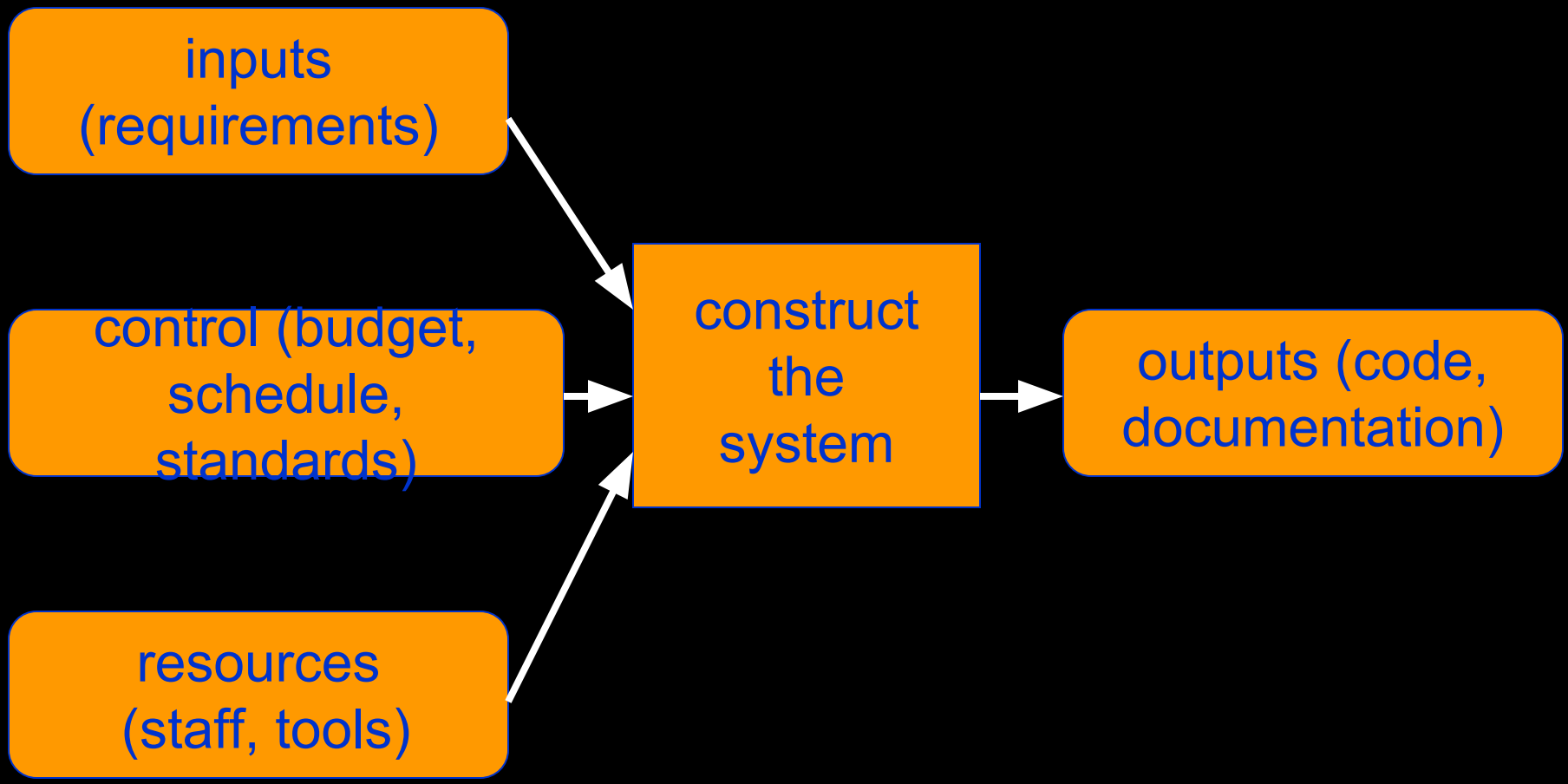
The KPAs of a level indicate the areas that an organization at the lower maturity level needs to focus to reach this level.

KPAs provide a way for an organization to gradually improve its quality of over several stages.

KPAs for each stage has been carefully designed such that one stage enhances the capability already built up.

- Trying to focus on some higher level KPAs without achieving the lower level KPAs would be counterproductive.

A repeatable model

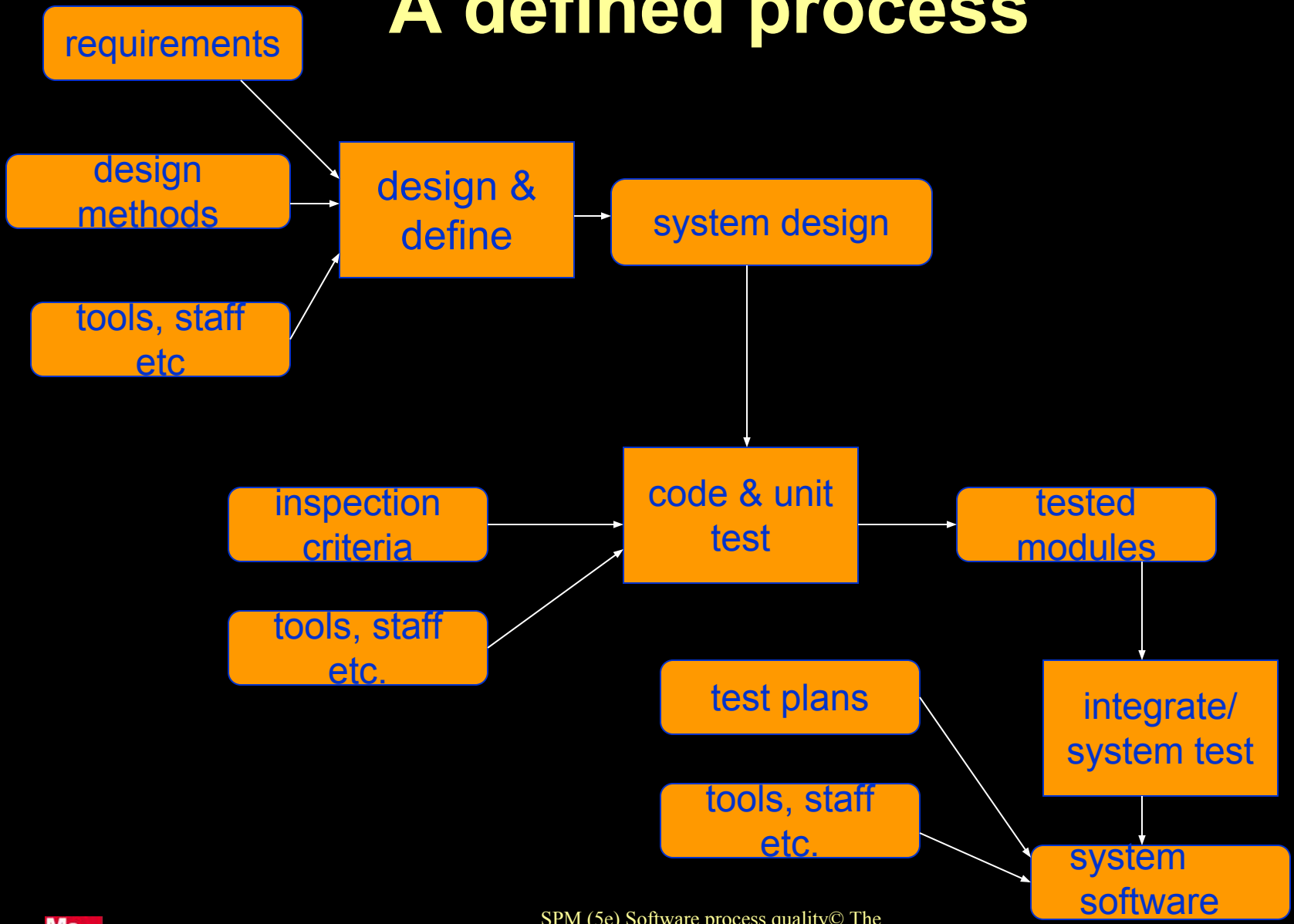


Repeatable model KPAs

To move to this level concentrate on:

- Configuration management
- Quality assurance
- Sub-contract management
- Project planning
- Project tracking and oversight
- Measurement and analysis

A defined process

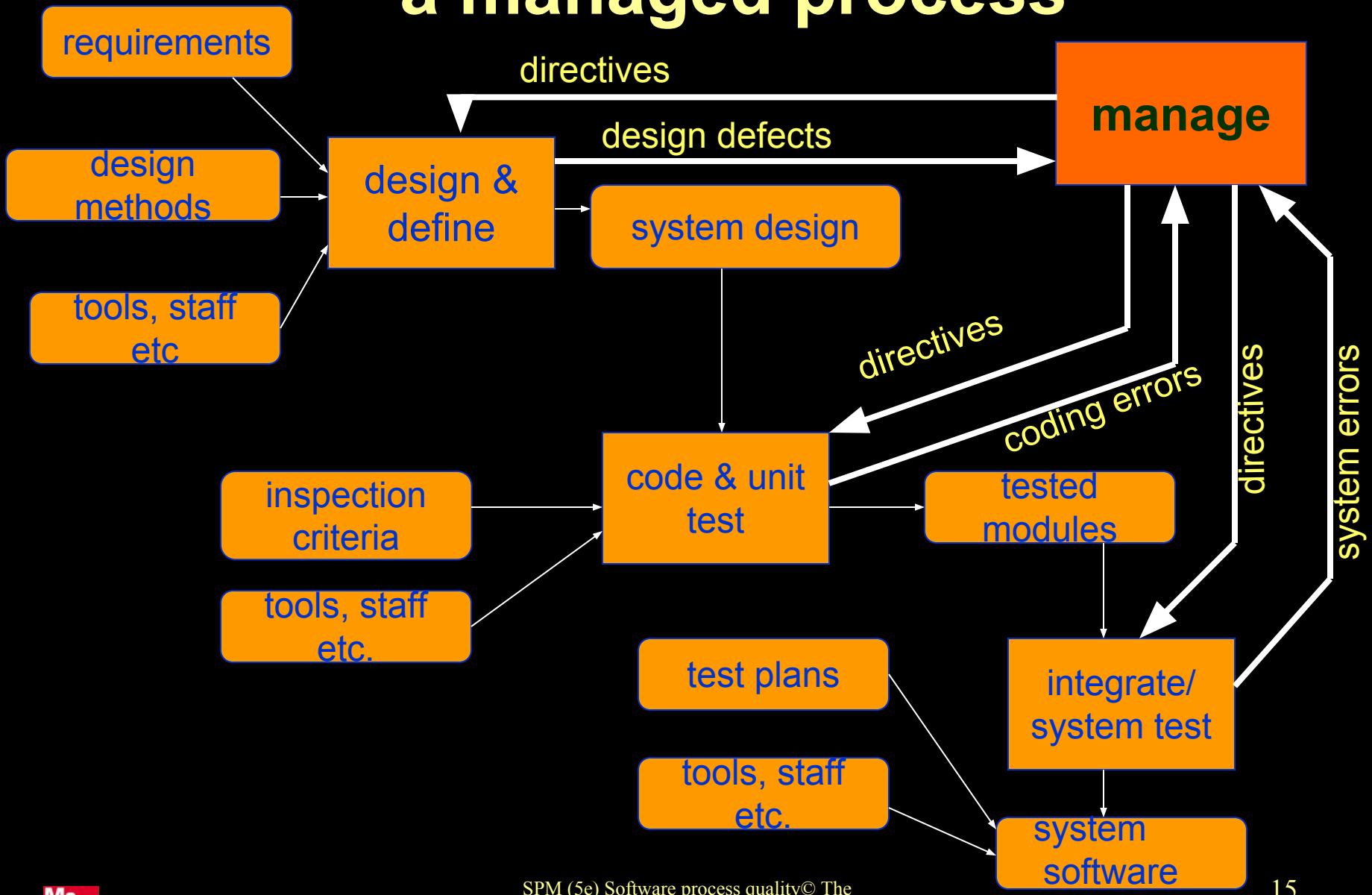


Repeatable to defined KPAs

Concentrate on

- Requirements development and technical solution
- Verification and validation
- Product integration
- Risk management
- Organizational training
- Organizational process focus (function)
- Decision analysis and resolution
- Process definition
- Integrated project management

a managed process

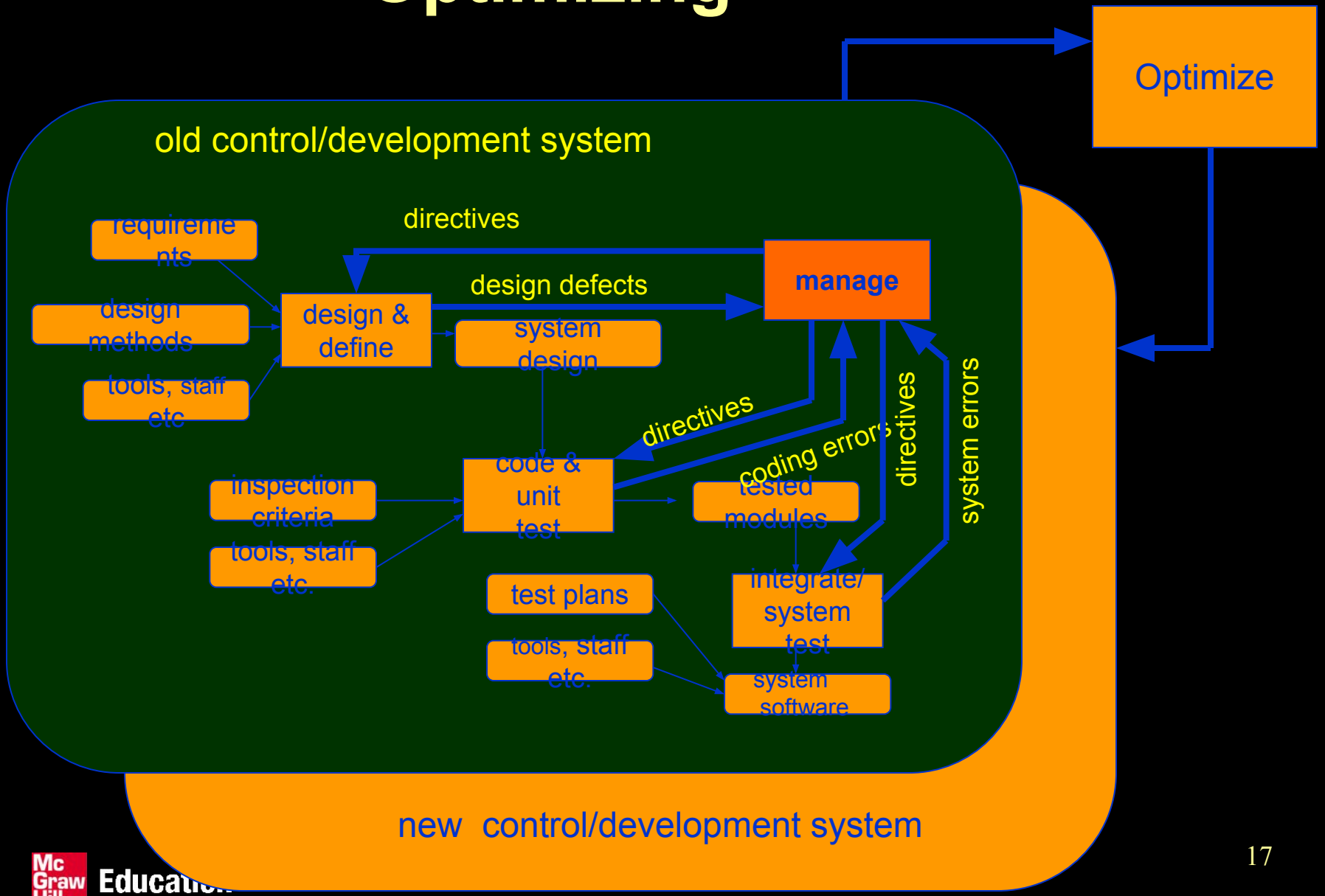


Defined to managed KPAs

Concentrate on:

- Organizational process performance
- Quantitative project management

Optimizing


Optimize
old control/development system
directives
requirements
design methods
tools, staff etc
design & define
design defects
manage
system design
directives
coding errors
directives
system errors
code & unit test
inspection criteria
tools, staff etc.
tested modules
test plans
tools, staff etc.
integrate/ system test
system software
new control/development system

Managing to optimizing: KPAs

concentrate on:

- Causal analysis and resolution
- Organizational innovation and deployment

CMMI (Capability Maturity Model Integration)

CMMI is the successor of the Capability Maturity Model (CMM).

After CMMI was first released in 1990:

- It became popular in many other domains
- Human Resource Management (HRM).: people management (PCMM)
- software acquisition (SA-CMM)
- systems engineering (SE-CMM)

Some questions about CMMI

suitable only for large organizations?

- e.g. need for special quality assurance and process improvement groups

defining processes may not be easy with new technology

- how can we plan when we've not used the the development method before?

higher CMM levels easier with maintenance environments?

can you jump levels? (HP level 5 in India)

ISO/IEC 15504 IT process assessment

To provide guidance on the assessment of software development processes

Process Reference Model: Needs a defined set of processes that represent good practice to be the benchmark

ISO 12207 is the default reference model

Could use others in specific environments

ISO 15504 performance attributes

CMMI level	ISO 15504
	0. incomplete
initial	1.1.process performance – achieves defined outcome
repeatable	2.1 process management – it is planned and monitored
	2.2 work product management – control of work products

ISO 15504 performance attributes - contd

CMMI	ISO 15504
Defined	3.1. Process definition
	3.2. Process deployment
Managed	4.1. Process measurement
	4.2. Process control
Optimizing	5.1. Process innovation
	5.2. Process optimization

Process Reference Model

A defined standard approach to development

Reflects recognized good practice

A benchmark against which the processes to be assessed can be judged

ISO 12207 is the default model

IS0 15504 Process Assessment

For each process in the relevant Process Reference Model
- For each set of attribute level criteria
- Assess whether:
  - N: not achieved 0-15%
  - P: partially achieved >15%-50%
  - L: largely achieved >50%-85%
  - F: fully achieved >85%

ISO 15504 performance indicators

This is just an example of how indicators for each level *might* be identified

1.Performance

Descriptions of maximum and minimum expected input values exist

2.1 Performance management

A plan of how expected input variable ranges are to be obtained exists which is up to date

ISO 15504 performance indicators 2

2.2 Work product management

There are minutes of a meeting where the input requirements document was reviewed and corrections were mandated

3.1 Process definition

A written procedure for input requirements gathering exists

3.2 Process deployment

A control document exists that is signed as each part of the procedure is completed

ISO 15504 performance indicators 2

4.1. Process measurement

Collected measurement data can be collected e.g. number of changes resulting from review

4.2. Process control

Memos relating to management actions taken in the light of the above

ISO 15504 performance indicators 3

5.1 Process innovation

Existence of some kind of 'lessons learnt' report at the end of project

5.2. Process optimization

Existence of documents assessing the feasibility of suggested process improvements and which show consultation with relevant stakeholders

Techniques to improve quality -Inspections

when a piece of work is completed, copies are distributed to co-workers

time is spent individually going through the work noting defects

a meeting is held where the work is then discussed

a list of defects requiring re-work is produced

Inspections - advantages of approach

an effective way of removing superficial errors from a piece of software

motivates the software developer to produce better structured and self-descriptive code

spreads good programming practice

enhances team-spirit

the main problem maintaining the commitment of participants

‘Clean-room’ software development

Ideas associated with Harlan Mills at IBM

- Three separate teams:
- Specification team – documents user requirements and usage profiles (how much use each function will have)
- Development team – develops code but does not test it. Uses mathematical verification techniques
- Certification team – tests code. Statistical model used to decide when to stop

Formal methods

- Use of mathematical notations such as VDM and Z to produce unambiguous specifications
- Can prove correctness of software mathematically (cf. geometric proofs of Pythagoras' theorem)
- Newer approach use Object Constraint Language (OCL) to add detail to UML models
- Aspiration is to be able to generate applications directly from UML+OCL without manual coding – Model Driven Architectures (MDA)

Verification versus Validation

Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase;

whereas validation is the process of determining whether a fully developed software conforms to its requirements specification.

Verification is carried out during the development process to check if the development activities are being carried out correctly,

whereas validation is carried out towards the end of the development process to check if the right product as required by the customer has been developed.

Testing: the V-process model

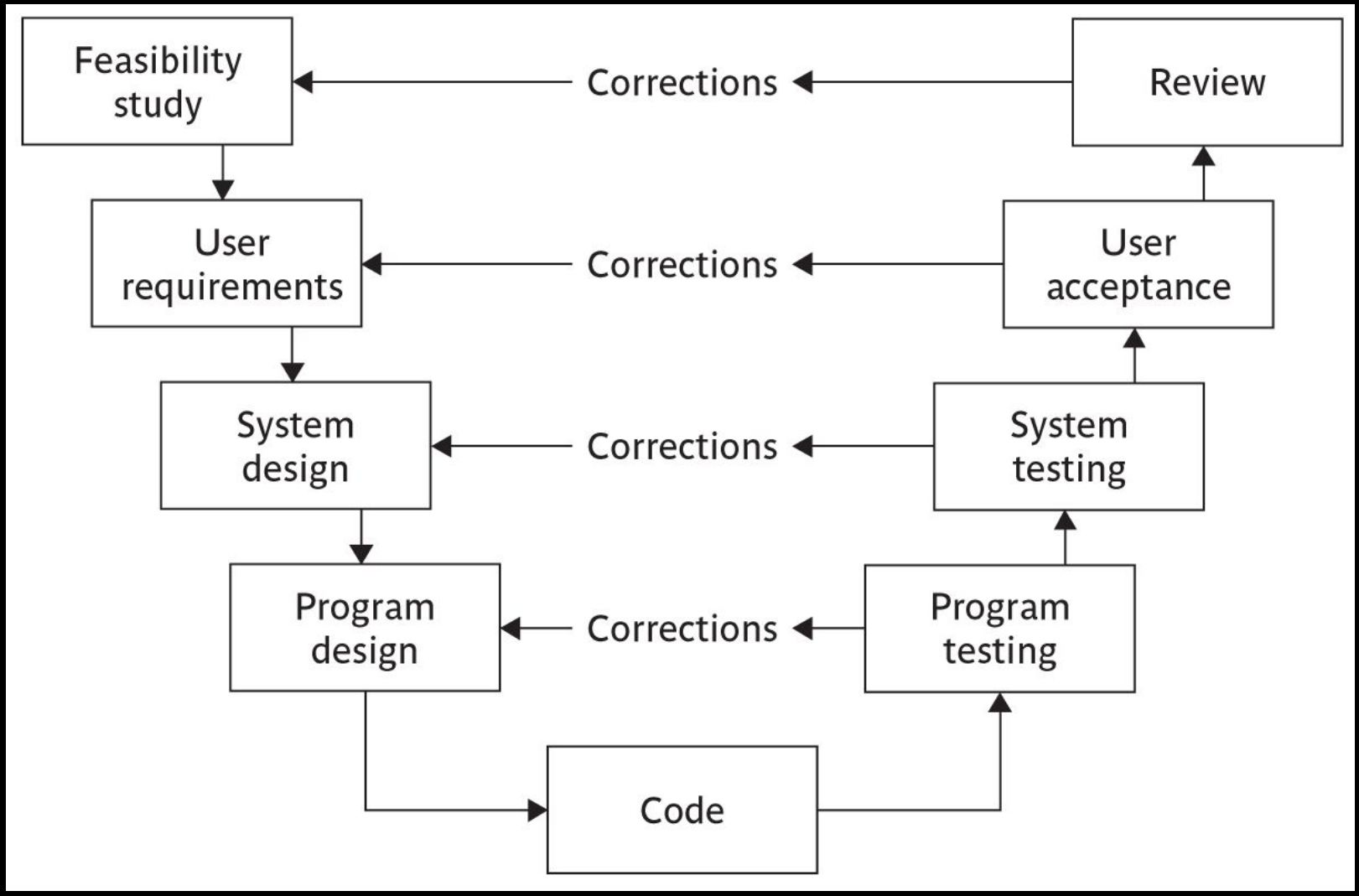
This shown diagrammatically on the next slide

It is an extension of the waterfall approach

For each development stage there is a testing stage

The testing associated with different stages serves different purposes e.g. system testing tests that components work together correctly, user acceptance testing that users can use system to carry out their work

Testing: the V-process model



Black box versus glass box test

Glass box testing

- The tester is aware of the internal structure of the code; can test each path; can assess percentage test coverage of the tests e.g. proportion of code that has been executed

Black box testing

- The tester is not aware of internal structure; concerned with degree to which it meets user requirements

Levels of testing

Unit testing
Integration testing
System testing

Testing activities

Test planning

Test suite design

Test case execution and result checking

Test reporting:

Debugging:

Error correction:

Defect retesting

Regression testing

Test closure:

Test plans

- Specify test environment
  - In many cases, especially with software that controls equipment, a special test system will need to be set up
- Usage profile
  - failures in operational system more likely in the more heavily used components
  - Faults in less used parts can lie hidden for a long time
  - Testing heavily used components more thoroughly tends to reduce number of operational failures

Management of testing

The tester executes test cases and may as a result find discrepancies between actual results and expected results – **issues**

Issue resolution – could be:

- a mistake by tester
- a fault – needs correction
- a fault – may decide not to correct: **off-specification**
- a change – software works as specified, but specification wrong: submit to change control

Decision to stop testing

The problem: impossible to know there are no more errors in code

Need to estimate how many errors are likely to be left

Bug seeding – insert (or leave) known bugs in code

Estimate of bugs left =

(total errors found)/(seeded errors found) x (total seeded errors)

Alternative method of error estimation

Have two independent testers, A and B

N_1 = valid errors found by A

N_2 = valid errors found by B

N_{12} = number of cases where same error found by A and B

Estimate = $(N_1 \times N_2)/ N_{12}$

Example: A finds 30 errors, B finds 20 errors. 15 are common to A and B. How many errors are there likely to be?

Test automation

Other than reducing human effort and time in this otherwise time and effort-intensive work,

- Test automation also significantly improves the thoroughness of testing.

A large number of tools are at present available both in the public domain as well as from commercial sources.

Types of Testing Tools

Capture and playback
Automated test script
Random input test
Model-based test

Software reliability

The reliability of a software product essentially denotes its *trustworthiness* or *dependability*.

Reliability of a software product usually keeps on improving with time during the testing and operational phases as defects are identified and repaired.

A reliability growth model (RGM) models how the reliability of a software product improves as failures are reported and bugs are corrected.

- An RGM can be used to determine when during the testing phase a given reliability level will be attained, so that testing can be stopped

Quality plans

quality standards and procedures should be documented in an organization's *quality manual*

for each separate project, the quality needs should be assessed

select the level of quality assurance needed for the project and document in a *quality plan*

typical contents of a quality plan

scope of plan

references to other documents

quality management, including organization, tasks, and responsibilities

documentation to be produced

standards, practices and conventions

reviews and audits

more contents of a quality plan

testing
problem reporting and corrective action
tools, techniques, and methodologies
code, media and supplier control
records collection, maintenance and retention
training
risk management