Last lecture!

Software Quality

Lecture 11 by Professor Vladimir Geroimenko

Module “Software Project Management“

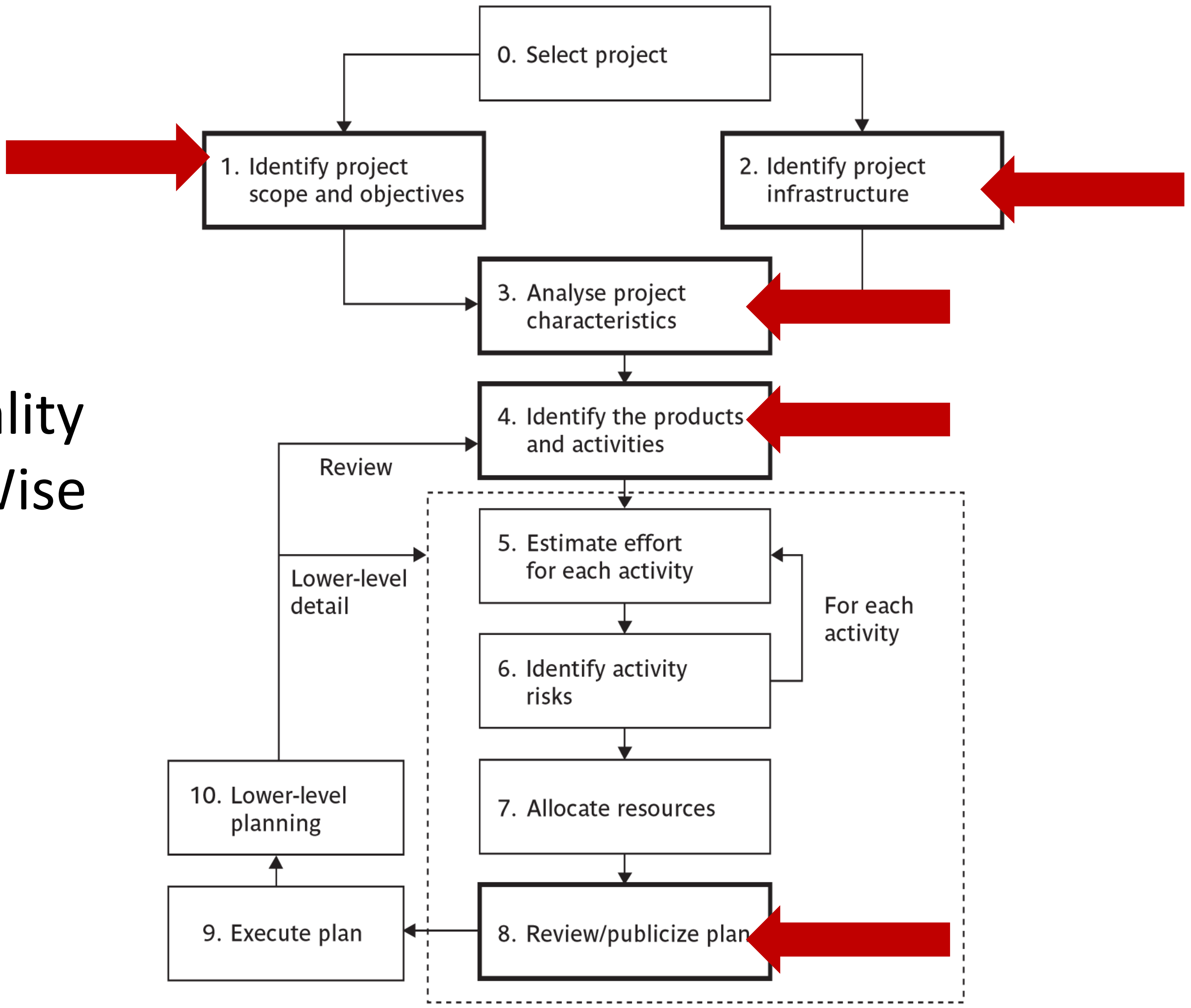
4 December 2016 - Teaching Week 11

Textbook reference: Chapter 13

Lecture Outline

- The importance of software quality
- ISO 9126 Software product quality
- Quality Management Systems
  - BS EN ISO 9001:2000
  - Capability Maturity Model
- Testing
- Quality plans

The place of software quality in the Step Wise framework



0. Select project
1. Identify project scope and objectives
2. Identify project infrastructure
3. Analyse project characteristics
4. Identify the products and activities
Review
5. Estimate effort for each activity
Lower-level detail
For each activity
6. Identify activity risks
7. Allocate resources
10. Lower-level planning
9. Execute plan
8. Review/publicize plan

The importance of software quality

- Increasing criticality of software
- The intangibility of software
- Project control concerns:
  - errors accumulate with each stage
  - errors become more expensive to remove the later they are found
  - it is difficult to control the error removal process (e.g. testing)

Quality specifications

Where there is a specific need for a quality, produce **a quality specification** that includes:

- **Definition**: description of the quality
- **Scale**: the unit of measurement
- **Test**: practical test of extent of quality
- **Minimally acceptable**: lowest acceptable value
- **Target range**: desirable value
- **Now**: value that currently applies

ISO standards

ISO 9126 Software product quality

Attributes of software product quality

- External qualities i.e. apparent to the user of the deliverable
- Internal qualities i.e. apparent to the developers of the deliverables and the intermediate products

ISO 14598 Procedures to carry out the **assessment** of the product qualities defined in ISO 9126

Types of quality assessment

1. During software **development**, to assist developers to build software with the required qualities
2. During software **acquisition** to allow a customer to compare and select the best quality product
3. Independent **evaluation** by assessors rating a software product for a particular community of users

ISO 9126 Quality in use

- **Effectiveness** – ability to achieve user goals with accuracy and completeness
- **Productivity** – avoids excessive use of resources in achieving user goals
- **Safety** – within reasonable levels of risk of harm to people, business, software, property, environment etc.
- **Satisfaction** – happy users

Please note: *Users* include those maintain software as well as those who operate it. ISO 9126 introduces Internal quality – External quality – Quality in use

ISO 9126 Software Qualities

functionality	does it satisfy user needs?
reliability	can the software maintain its level of performance?
usability	how easy is it to use?
efficiency	relates to the physical resources used during execution
maintainability	relates to the effort needed to make changes to the software
portability	how easy can it be moved to a new environment?

Sub-characteristics of Functionality

- Suitability
- Accuracy
- Interoperability
  - ability of software to interact with other software components
- Functionality compliance
  - degree to which software adheres to application-related standards or legal requirements e.g. audit
- Security
  - control of access to the system

Sub-characteristics of Reliability

- Maturity
  - frequency of failure due to faults - the more the software has been used, the more faults will have been removed
- Fault-tolerance
- Recoverability
  - note that this is distinguished from ‘security’
- Reliability compliance
  – complies with standards relating to reliability

Sub-characteristics of Usability

- Understandability
  - easy to understand?
- Learnability
  - easy to learn?
- Operability
  - easy to use?
- Attractiveness e.g. like entertainment/game product
- Usability compliance
  - compliance with relevant standards

Sub-characteristics of Efficiency

- Time behavior
  - e.g. response time
- Resource utilization
  - e.g. memory usage
- Efficiency compliance
  - compliance with relevant standards

Sub-characteristics of Maintainability

- “Analysability”
  - ease with which the cause of a failure can be found
- Changeability
  - how easy is software to change?
- Stability
  - low risk of modification having unexpected effects
- “Testability”
- Maintainability conformance

Sub-characteristics of Portability

- Adaptability
- “Installability”
- Co-existence
  - Capability of co-existing with other independent software products
- “Replaceability”
  - factors giving ‘upwards’ compatibility (‘downwards’ compatibility is excluded)
- Portability conformance
  - Adherence to standards that support portability

Using ISO 9126 quality standards in development mode

- Judge the importance of each quality for the application
  - for example, safety critical systems - *reliability* very important
  - real-time systems - *efficiency* important
- Select relevant external measurements within ISO 9126 framework for these qualities, for example
  - mean-time between failures for *reliability*
  - response-time for *efficiency*

Using ISO 9126 quality standards

- Map measurement onto ratings scale to show degree of user satisfaction – for example response time

response (secs)	**rating**
<2	Exceeds requirement
2-5	Target range
6-10	Minimally acceptable
>10	Unacceptable

Using ISO 9126 quality standards

- Identify the relevant internal measurements and the intermediate products in which they would appear

e.g. at software design stage the estimated execution time for a transaction could be calculated

Using ISO 9126 approach for application software selection

- Rather than map engineering measurement to qualitative rating, map it to a score
- Rate the importance of each quality in the range 1-5
- Multiply quality and importance scores – see next slide

Response (secs)	Quality score
<2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

Weighted quality scores

		Product A		Product B	
Product quality	Importance rating (a)	Quality score (b)	Weighted score (a x b)	Quality score (c)	Weighted score (a x c)
usability	3	1	3	3	9
efficiency	4	2	8	2	8
maintain-ability	2	3	6	1	2
Overall totals			**17**		**19**

How do we achieve product quality?

- **The problem**: quality attributes tend to *retrospectively* measurable
- Need to be able to examine processes by which product is created *beforehand*
- The production process is a network of *sub-processes*
- Output from one process forms the *input* to the next
- *Errors* can enter the process *at any stage*

Correction of errors

- Errors are more expensive to correct at later stages
  - need to rework more stages
  - later stages are more detailed and less able to absorb change
- Barry Boehm
  - Error typically 10 times more expensive to correct at coding stage than at requirements stage
  - 100 times more expensive at maintenance stage

For each activity, *define*:

- Entry requirements
  - these have to be in place before an activity can be started
  - example: ‘a comprehensive set of test data and expected results be prepared and independently reviewed against the system requirement before program testing can commence’

For each activity, *define*

- Implementation requirements
  - these define how the process is to be conducted
  - example ‘whenever an error is found and corrected, *all* test runs must be completed, including those previously successfully passed’

For each activity, *define*

- Exit requirements
  - an activity will not be completed until these requirements have been met
  - example: ‘the testing phase is finished only when all tests have been run in succession with no outstanding errors’

Quality Management Systems: BS EN ISO 9001:2000

- 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 software
- Does **not** set universal product/service standards
- Does **specify the way** in which standards are established and monitored
- *Please note: These types of standards are concerned with the certification of the development process, not of the end-product.*

ISO 9001:2000 principles

1. Understanding the requirements of customers so that they can be met or even exceeded;
2. Leadership to provide the unity of purpose and direction needed to achieve quality objectives;
3. Involvement of staff at all levels;
4. A focus on the individual processes which create intermediate and deliverable products and services;

ISO 9001:2000 principles

5. A focus on the system of interrelated processes that create delivered products and services;
6. Continuous improvement of processes;
7. Decision-making based on factual evidence;
8. Building mutually beneficial relationships with suppliers;

ISO 9001:2000 cycles

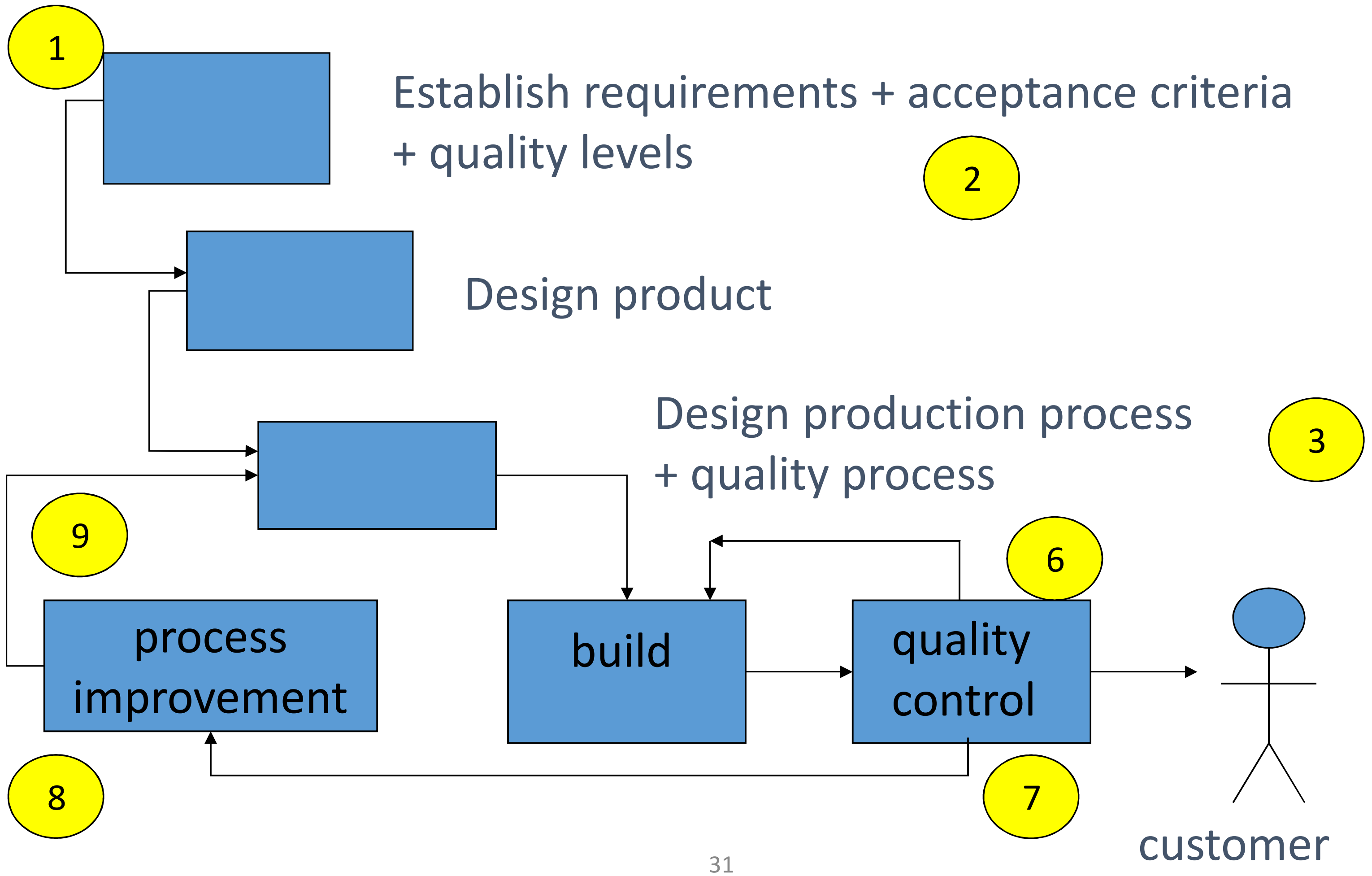
These principles are applied through cycles which involve the following activities:

1. determining the needs and expectations of the customer;
2. establishing a *quality policy*, that is, a framework which allows the organization’s objectives in relation to quality to be defined;
3. design of the *processes* which will create the products (or deliver the services) which will have the qualities implied in the organization’s quality objectives;
4. allocation of the responsibilities for meeting these requirements for each element of each process;

ISO 9001:2000 cycles

5. ensuring resources are available to execute these processes properly;
6. design of methods for measuring the effectiveness and efficiency of each process in contributing to the organization’s quality objectives;
7. gathering of measurements;
8. identification of any discrepancies between the actual measurements and the target values;
9. analysis and elimination of the causes of discrepancies.

ISO 9001:2000 cycle

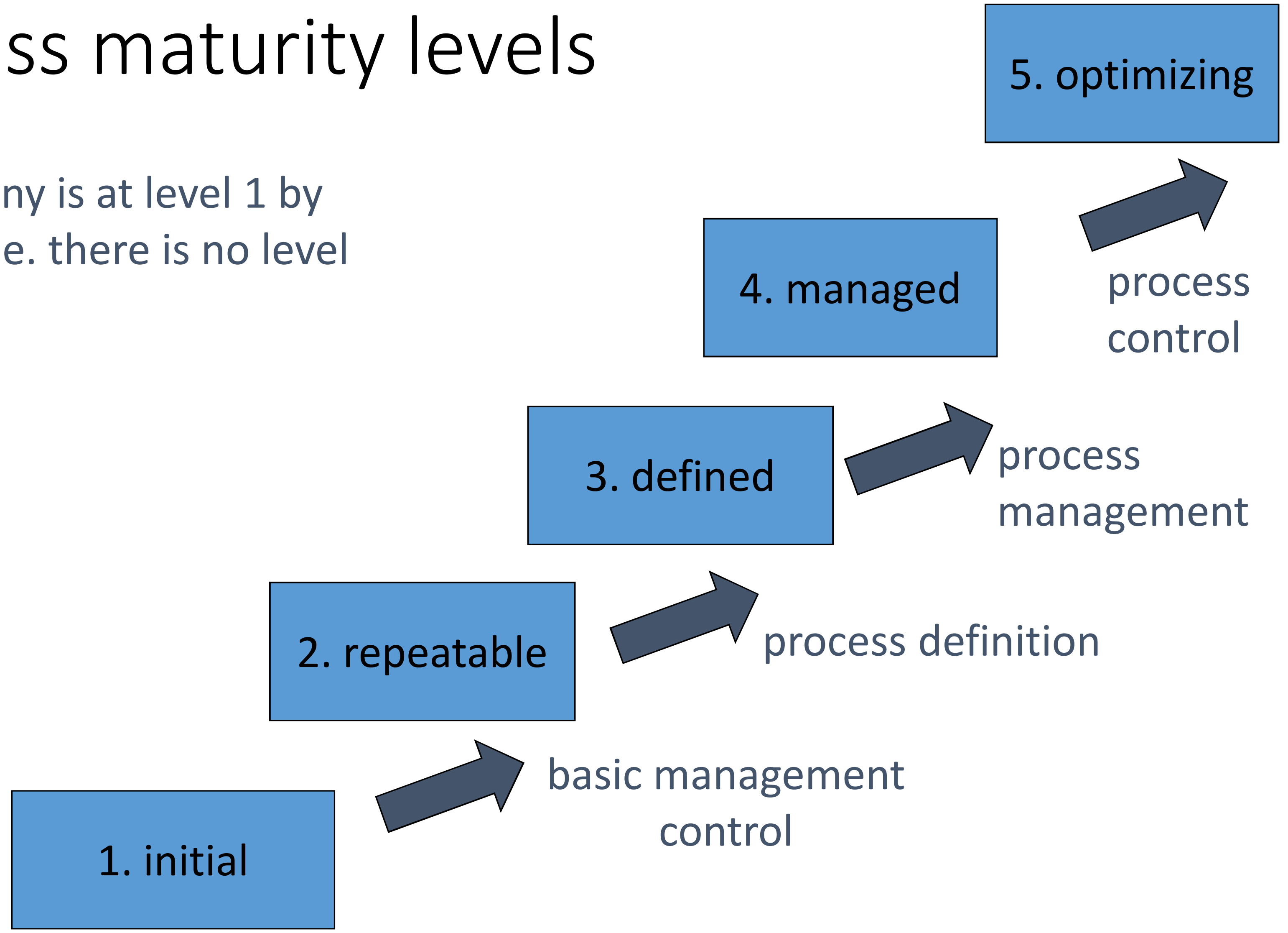


Quality Management Systems: Capability Maturity Model (CMMI)

- Created by Software Engineering Institute, Carnegie Mellon University
- CMM (Capability Maturity Model) was developed by SEI for US government to help procurement
- Book ‘Managing the software process’ by Watts S. Humphrey, Addison Wesley
- Assessment is by questionnaire and interview
- The latest version (called **CMMI = CMM Integration**) tries to set up a generic model which can be populated differently for different environments

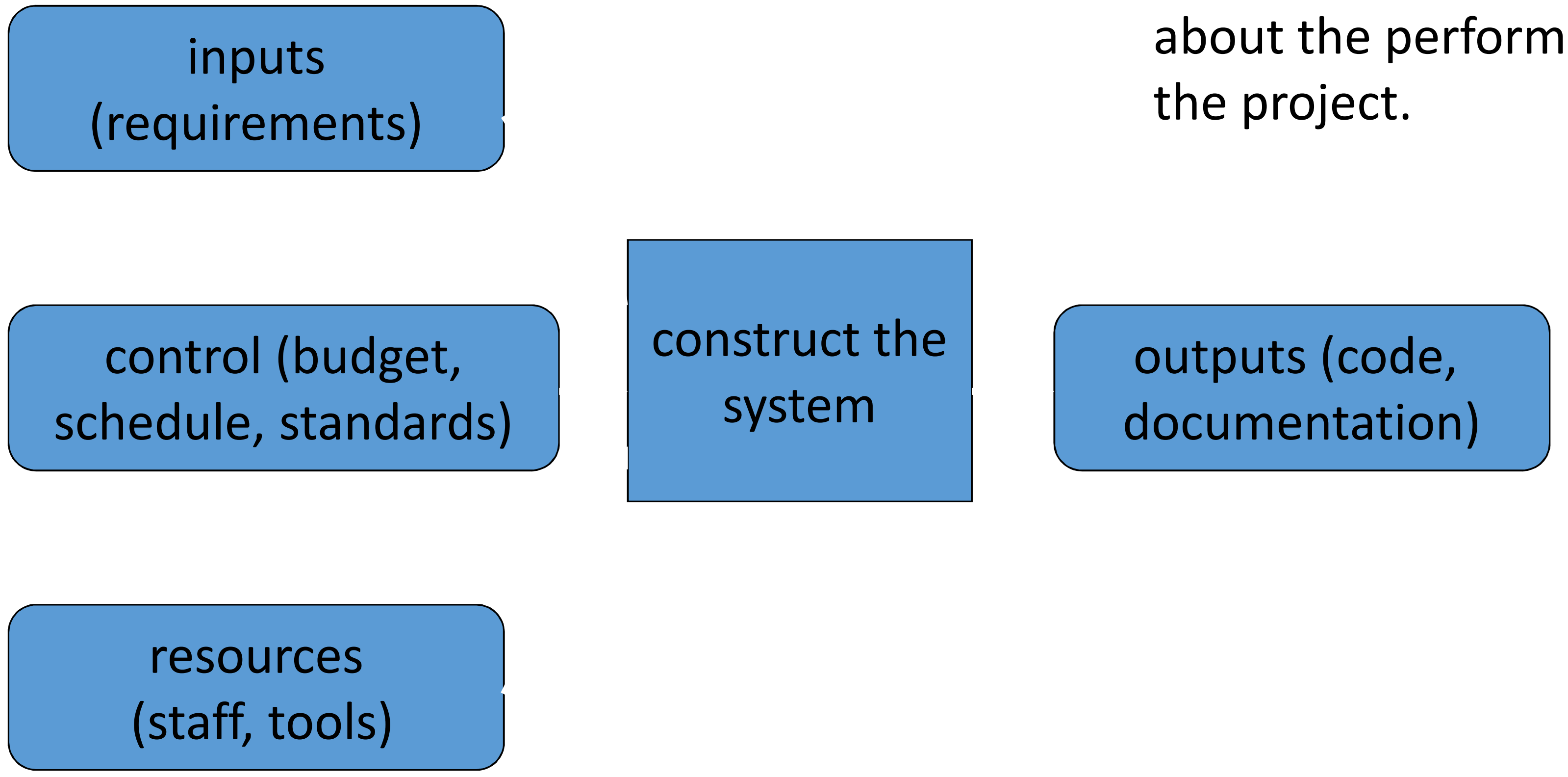
Process maturity levels

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



Level 2: A repeatable model

The idea is that the process is under ‘statistical control’. Essentially this means that you are collecting data about the performance of the project.



inputs (requirements)
control (budget, schedule, standards)
resources (staff, tools)
construct the system
outputs (code, documentation)

Level 2: Repeatable model KPAs

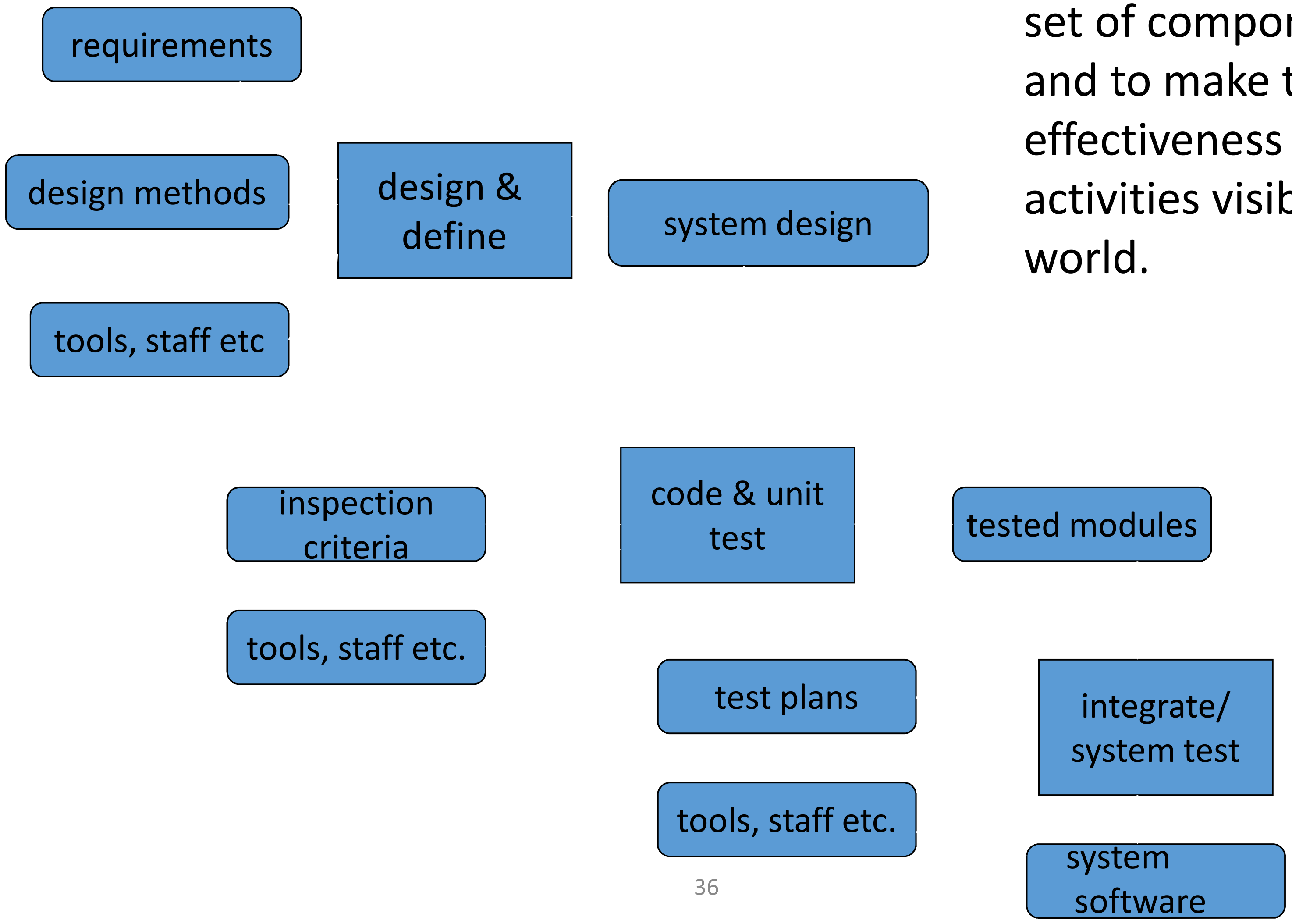
To move to this level concentrate on:

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

KPAs =

Key Process areas.

Level 3: A defined process



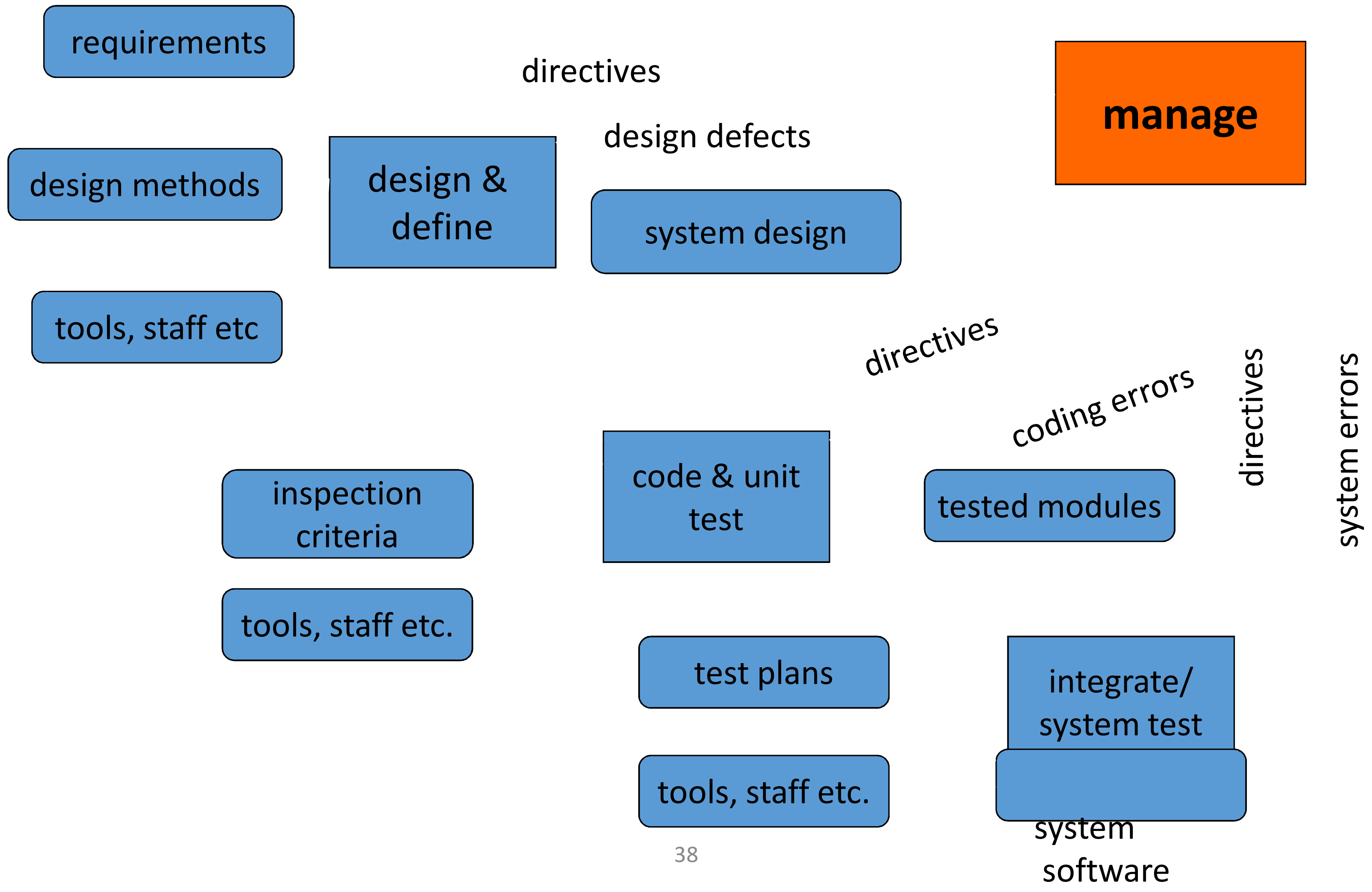
At this level an attempt is made to breakdown the project into a set of component sub-activities and to make the progress and effectiveness of these sub-activities visible to the wider world.

Level 3: 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

Level 4: A managed process

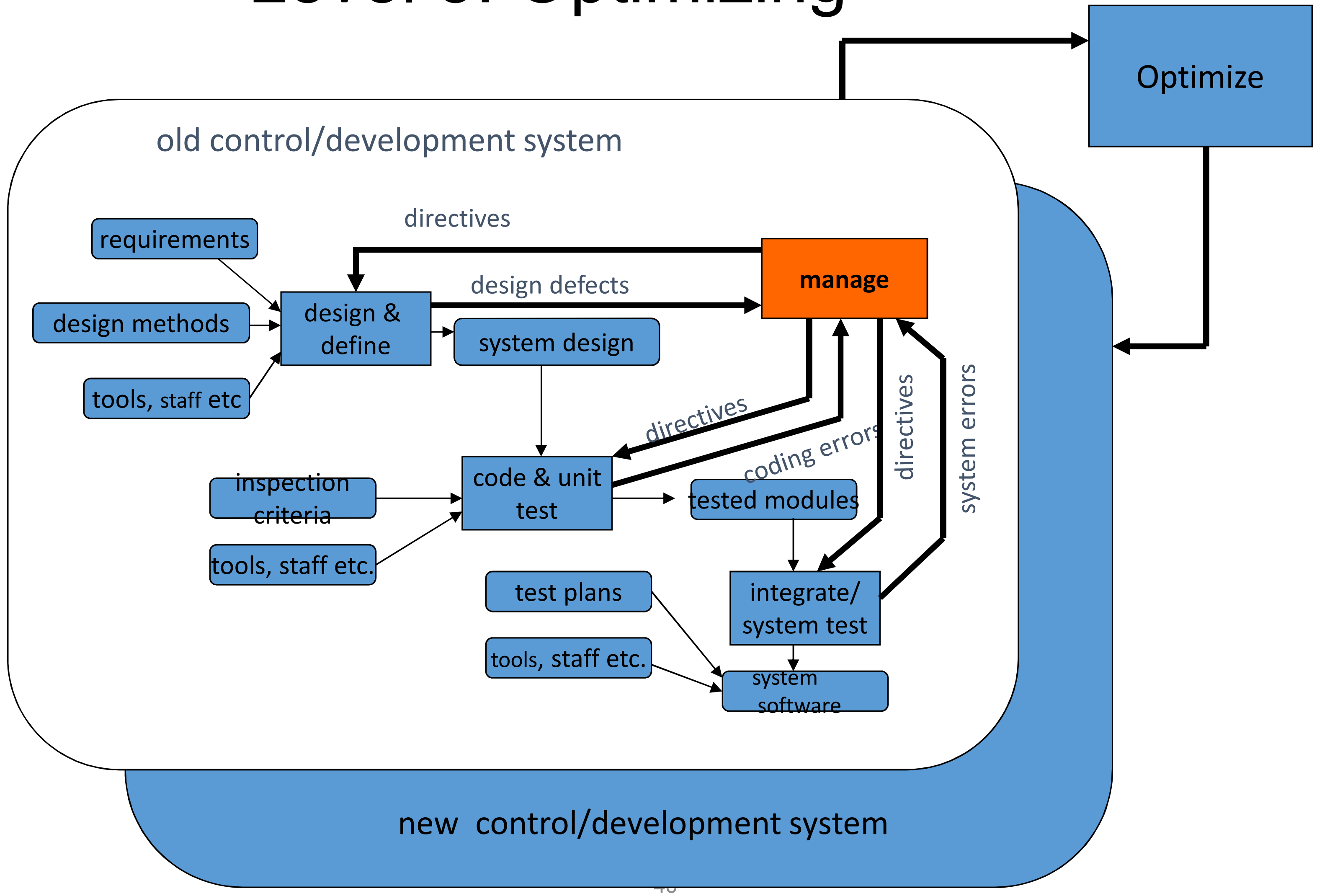


Level 4: Defined to managed KPAs

Concentrate on:

- Organizational process performance
- Quantitative project management

Level 5: Optimizing



Level 5: Managing to optimizing KPAs

Concentrate on:

- Causal analysis and resolution
- Organizational innovation and deployment

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 development method before?
- Higher CMM levels easier with maintenance environments?
- Can you jump levels?

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

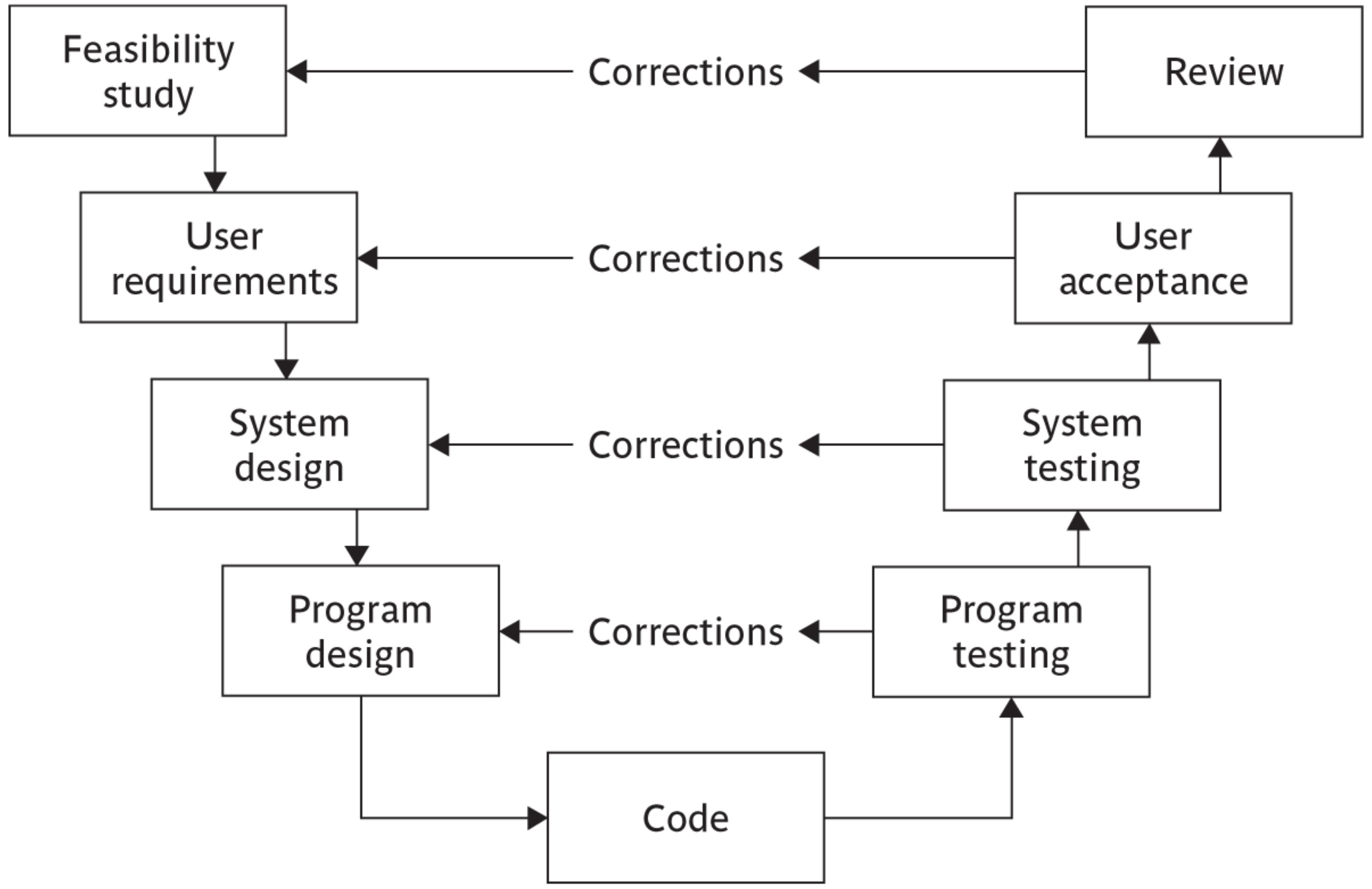
Three separate teams:

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

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

Black box testing

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

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

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?

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

1. Scope of plan
2. References to other documents
3. Quality management, including organization, tasks, and responsibilities
4. Documentation to be produced
5. Standards, practices and conventions
6. Reviews and audits

More contents of a quality plan

7. Testing
8. Problem reporting and corrective action
9. Tools, techniques, and methodologies
10. Code, media and supplier control
11. Records collection, maintenance and retention
12. Training
13. Risk management

Conclusion

- Quality is an important part of Software Project Management
- Software quality is important to both software users and developers
- Quality by itself is a vague concept and practical quality requirements have to be carefully defined
- Most the qualities that apparent to the users of software can only be tested for when the system is completed
- Some quality-enhancing techniques concentrate on testing **products**, while others try to evaluate the quality of the **development processes** used.

Thank you for your attention

Any questions, please?