

PART I**#3 project management**

- problems related to control of: resources, time, product and risk
- what to keep in mind when planning or controlling projects
- general understanding of what information you would find in PERT charts, Gantt charts and work breakdown structure charts
- how to run good meetings and why these steps are necessary
- risk management calculations and their meaning
- how to identify and respond to risks
- metrics related to project management
- responsibilities of a project manager

#22 software measurement

- measurement and concept of control and prediction
- project control measurements:
- estimating – factors that influence man-months
- software control measurements:
- reliability
- complexity

#4 software lifecycles

- for waterfall, prototype, spiral, incremental:
- differences between them
- benefits and drawbacks
- meaning of the V-model

#20 software maintenance and reuse

- types of programs and why we need to distinguish between them (for testing and for maintenance)
- types of maintenance and their handling/history
- general understanding of what happens to software as it's maintained
- difficulties in maintaining systems
- what needs to be in place for maintenance to be effective/efficient
- reuse and why it's important in maintenance as well as in development
- how to keep software healthy

#21 process modeling and improvement

- what is process modeling and why it needs to be documented
- how to improve the process over time
- general issues in process improvement
- capability maturity model and its classifications

PART II**#8 testing**

- difference between verification and validation – and the tests done for each
- timing of tests during the project
- characteristics of good testing practices
- relationship between partitioning and testing
- blackbox and whitebox testing – differences and reasons for
- integration testing and system testing – what are we trying to achieve
- large system testing and automation
- making sure your testing practices are effective

#9 reviews and inspections

- types of reviews, inspections and walkthroughs
- reviewing specifications, code, design, plans, etc.
- benefits and constraints of formal inspections
- structure and control of inspections

#10 formal verification

- purpose and definition of first order propositional logic
- pre- and post-conditions – how are they used?
- what is a correctness proof?
- Hoare's notation and what it generally means
- when and how to use formal program proofs

#11 debugging and exceptions

- difficulties in debugging, and general wisdom
- scientific method of debugging
- instrumentation of testing tools and firewalls – general understanding
- exception handling options and their meanings
- change management – control and documentation methods

PART III**#14 requirements analysis**

- the requirements engineering process and why it is difficult
- the need to focus on the “what” before we address the “how”
- differences between functional and non-functional requirements?
- techniques for eliciting requirements: differences, reasons?
- requirements engineering methods
- modeling and modeling methods

#18 specifications

- reasons/uses for software specifications
- characteristics of good specifications, with examples
- traceability and its importance

#5 decomposition and abstraction

- reasons for decomposition and abstraction – esp. related to analysis and design
- the three methods of abstraction and their limits and options
- characteristics of good decomposition

#6 procedural abstractions

- benefits of procedural abstractions
- what needs to be known about a procedure?
- total vs partial procedures – why do we care?
- desirable properties of procedural abstractions

#7 data abstraction

- benefits of data abstraction and information hiding
- how to document data abstractions
- some capabilities of O-O programming languages

#13 software design representation

- for each view point:
 - what they represent
 - their uses
 - an example of two

#15 structured modeling

- components of DFD's
- DFD hierarchies (context, level 1, etc.) and how they break down
- ability to modify diagrams
- general methodology (but not the subtle differences between DFD variants)
- uses, advantages and disadvantages

#16 object-oriented modeling

- modeling primitives
- criteria for selecting objects
- key principles
- ability to read diagrams
- general understanding, but not details of variants

17 formal modeling

- general uses of formal methods/ analysis

#19 software architectures

- for the six architectural styles:
- general description of diagram
- uses and disadvantages
- not KWIC nor description languages

#12 software quality

- definition of software quality – by IT professional and by users
- relationship of measurement to concepts of quality
- coupling and cohesion as measure of potential quality

WHAT'S NOT ON THE EXAM?

- history (names, dates...)
- details from assignments

WHERE IS THE EMPHASIS?

PART I and PART II