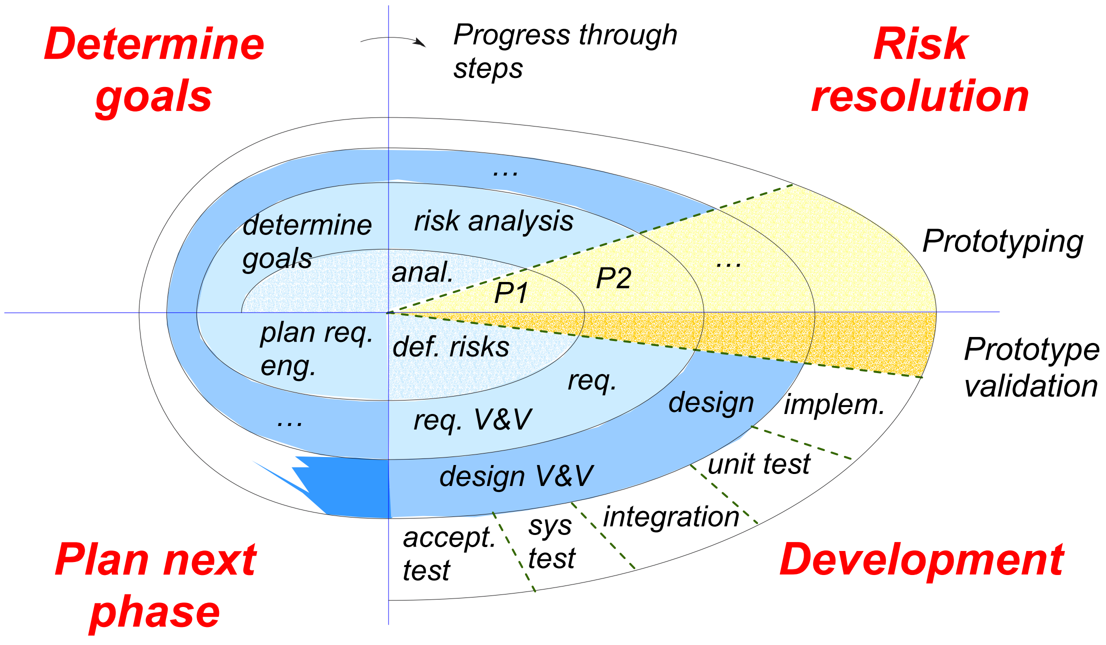
Software Engineering (400071) – written exam
Schema solution

Q1: Life cycle models

a) **Waterfall (**strict order, never come back; each phase's output is input for the next phase); **evolutionary prototyping** (subset of requirements to implement, interact with user), **incremental development** (identify all requirements at first then implement increments).
b) See slides
c) **Waterfall (**each phase one spiral); **evolutionary** ("II Risk resolution" means to produce a prototype/feasibility study, and then validate the prototype, **incremental development** (each spiral covers waterfall model in III Development).
d) Motivate your choice as based on the problem description!! E.g., for the EES throw-away prototyping could provide the first version of the GUI to test usability with end-users; waterfall model could be used as requirements are well understood and system is not likely to undergo fast changes.



Q2: Requirements engineering

a) Functional requirements:

☑ Moscow List:

Order	Name UC	Motivation
Must have	Vote (elector)	…
Must have	Authenticate (all users)	…
Must have	Access results (final, system manager)	…
Must have	Calculate results (final)	Application logic
Should have	Monitor progress (in progress, system manager)	
Should have	Calculate progress (statistics)	Application logic
Could have	-	-
Wont have	Calculate progress – know who did vote	Conflict!!!

☑ UCD for Must have's (…)

b) NFR: usability (GUI is very important); stability (not run the risk to invalidate elections); hard persistency (cannot loose data)

c) Domain model: should cover Vote (what, time, place), User (generalities, ID, type, PID, status of election); Referendum (description, data); Status (options=<party, representative>, electorsPool, state={in-progress, final}, statistics).

d) UC description

Basic c.o.e.	Vote
Description	1. Elector performs "Authenticate" 2. System shows referendum options 3. Elector makes his/her choice 4. System displays the choice and asks for confirmation 5. If "yes" the vote is stored and the Elector is logged out (UC stops) 6. If "no" all data is cleared and the UC is repeated from step 2.
Alternative c.o.e.	Elector is voting and the system stops
	1.-4. (the same) 5. Elector answers "yes": the vote is stored (system stops before Elector is logged out) 6. When the system is restarted, a rollback/commit procedure is executed (e.g. all locks are verified and if votes were stored, the logged in users are logged out; otherwise they are invited to vote again).
Exceptional c.o.e.	Elector forgot his/her PID
	1. Elector performs "Authenticate" (not successful) 2. UC stops

Q3: Software architecture/design patterns

a) Architectural patterns are macro architectural solutions to recurring problems; design ones are micro architectural solutions instead (never covering the whole system). The general pattern structure covers (name, context, problem, solution, variants).

b) E.g., deployment view or component view. Components are e.g., GUI, EES logic, DB users, DB referendum.

c) E.g. 3-tier architectural pattern. It naturally separates the tiers; it supports scalability and persistency.

Q4: Software design

a) Models the solution decisions, how to support requirements
b) Difference with requirements engineering (what-how; different stakeholders; different objectives).
c) One among abstraction, information hiding, cohesion, coupling, etc. (see slides).

Q5: Testing

a) Verification (local, to check correctness); validation (against requirements, to check that what provided fulfills customer needs)