Lifecycle models comparison (Roman Nesterov)

Model	Advantages	Disadvantages	Situations to be most appropriate
Waterfall	straightforward to exploit	inflexible, slow, costly	mainframe-based/transaction-oriented
	the rigidity of the model makes the management easier through the production of deliverables at each phase	problems often remains undiscovered until the testing stage	large, expensive and complicated with clear objectives and solutions
	the boundaries of phases are clear	difficult to respond to changing environment	requirements are defined unambiguously and remain stable
		the gap between users and developers is promoted	
Code-and-Fix	less management expertise needed	the process can't be assessed having no practical means to implement this analysis	small projects with distinct requirements
	less project planning	maintenance problems	
	early process (due to fast coding)	costly due to the cycles of accomplishing user requirements	
		a lot of unplanned actions can take place	
Evolutionary	customers are involved in the process of requirements formulation	difficult tracing	
	requirements can be changed	wrong system structure influence	
	prototypes reflect the real situation unambiguously	use of special techniques being incompatible	
		producing documents is costly	
Incremental	reuse of knowledge gained at the previous development stages	interfaces is required since some modules can be ready earlier than the others	large projects with changing requirements
	risks can be easily identified at the early project stages	procrastination of difficult problems rather than resolving them to demonstrate success to the company management	event-driven systems
	progress can be easily measured and presented to the stakeholders		leading-edge applications
	incremental changes monitoring		
Spiral	risk avoidance enhancement	limited reuse due to the high customization	real-time, safety-critical systems
	can incorporate different models	controls for moving between cycles are to be developed	implementation has priority over functionality
	on a given iteration the most appropriate model can be selected in accordance with the risk assessments	can degenerate to waterfall model	
		cycles have no firmly defined deadlines	
Rapid protityping	incorrect user requirements risk is reduced	the possibility of an unstable prototype to become a final product	small-to-medium scale projects
	respond fast to fluctuating requirements	projects terms are often difficult to measure	applications are highly interactive with clearly defined user groups
	progress can be easily measured	can degenerate to code-and-fix model	
	product can be quickly marketed		
Agile	high rate of customer satisfaction due to the regular software releases	difficult to estimate the effort needed at the beginning stage	highly changing and fluctuating requirements together with the plans that are also flexible
	close cooperation between stakeholders	documentation can be poor	
	regular and fast adaptation to changing environment	the project can easily be taken off main track in case customer has no clear vision of it	
	frequent delivery of stable software		
COTS	fast and not expensive	functionality limitations	
	all basic functionality included	licensing problems and additional licensing burden	
	easy to run and exploit		