MODELLO	TIPO	CARATTERISTICHE	PRO	CONTRO
Code & Fix	Non vero modello	Scrittura codice e poi fix dei bug	se LOC < 1500	impraticabile per progetti grandi
Waterfall	Plan Driven	primo storico modello, fasi a cascata, ognuna produce un deliverable.FASI: req. - design - implem. - verif. - maintenance ...	enfasi su analisi dei req., pospone l'implementazione a dopo aver capito i bisogni del cliente, ancora usato (anche se raramente) se i req. sono chiari e stabili	lineare, rigido, monolitico, no feedback tra fasi, no //, unica data di consegna --> sw obsoleto, molta carta
Waterfall con prototipazione	Plan Driven	... + codice usa e getta x fornire ai clienti una base per meglio definire i requisiti	mitigati i contro del waterfall	vedi waterfall
Waterfall con feedback e iterazioni	Plan Driven	... + feedback tra fasi e possibilità di "tornare indietro" in più iterazioni	mitigati i contro del waterfall	vedi waterfall
V-MODEL	Plan Driven	iterazioni esplicite: PARTE SX (guidano le fasi a DX) e PARTE DX (testing, su cui è posta enfasi). Se Test non da buon esito --> SX	gia in //, usato x Safety Critical	Rimangono ancora problemi generali: i proj reali si conformano raramente allo schema sequenziale, il cliente non sa esattamente cosa vuole all'inizio, req. non congelabili, unico rilascio, errore all'inizio --> comunque problema
PROTOTYPING	Evolutivo	prototipo usa e getta o che si estende nel progetto reale. il prototipo è ridotto (- funz., no sicurezza, - usab., - effic.) e spesso ad alto livello	raffinare req., rilevazioni errori precocemente	spesso si butta il prototipo (tempo e psicologia svilupatori) o, se no, se si tiene rimangono scelte non ottimali
ITERAT.-INCREM.	Evolutivo	sviluppo di varie release (dopo la prima release si procede in parallelo) con feedback degli utenti --> spesso usate insieme	/	/
INCREM. PURO	Evolutivo	a ogni release una nuova funzionalità, molto importante la pianificazione, criteri su cosa fare prima: funzioni critiche e importanti per gli utenti	/	/
ITERAT. PURO	Evolutivo	alla prima release ci sono già tutte le funzionalità, poi raffinate	release secondo feedback, più facile prevedere le risorse perché ogni fase è più piccola, problemi individuati presto, risposta veloce al mercato	/
A SPIRALE	Evolutivo (META)	evolutivo, risk driven (scelte basate sulla stima dei rischi). 4 FASI: planning, risk analysis, engineering, customer eval (a spirale)	adatto x sistemi complessi, valutazione rischio	necessita competenze x stima rischi, serve personalizzazione (è meta), no panacea, posso cmq trovare problemi
UP	Plan Driven (META)	proviene da 3 modelli. caratteristiche: oop, uml, use case, iterativo (più iterazioni, ognuna con 4 fasi terminanti con MILESTONE, ognuna divisà in attività). FASI: inception, elaboration, construction, transition	Pro dello spirale, supportato da tool	/
COMPONENT BASED	Evolutivo?	approccio "lego" con riutilizzo sw propri o da store con componenti. Lo sviluppo inizia con requisiti e identificaz. componenti (lo ho? devo svilupparlo?). Magari ho A ma al cliente serve B: si contratta	- sw da sviluppare, (- costi, - rischi, consegne + veloci)?	contrattare, integrazione non sempre facile, spesso componenti modificati in-opera dai produttori
XP	Agile	/	/	/