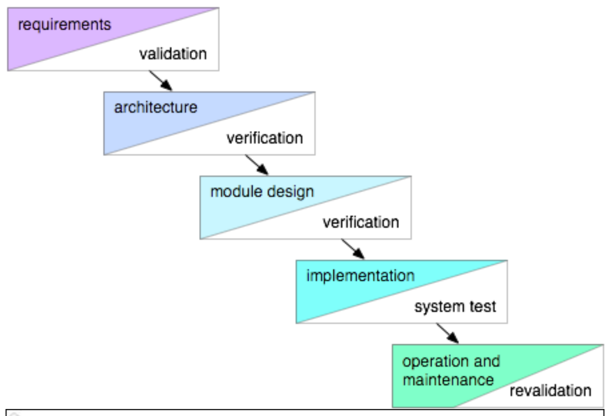
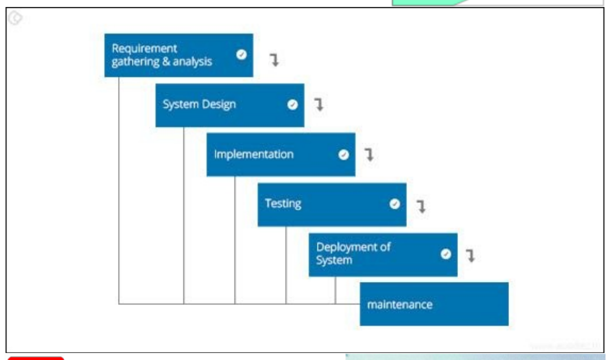
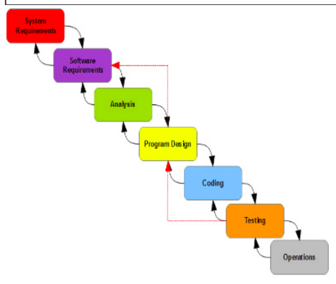
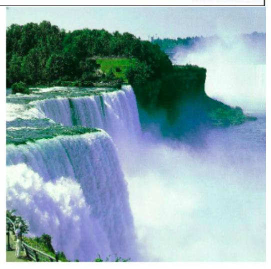
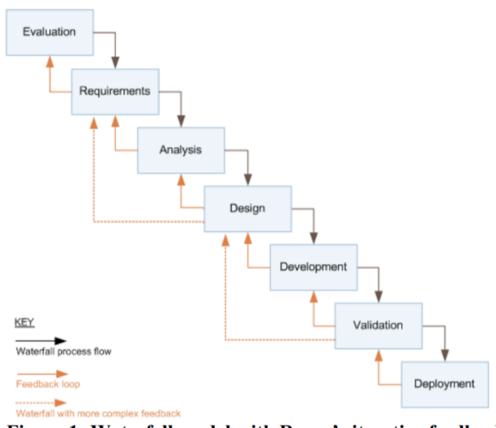
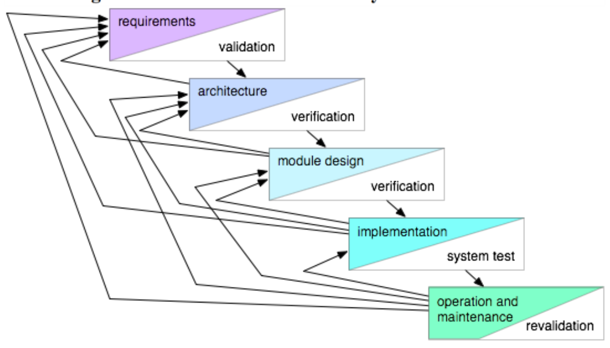
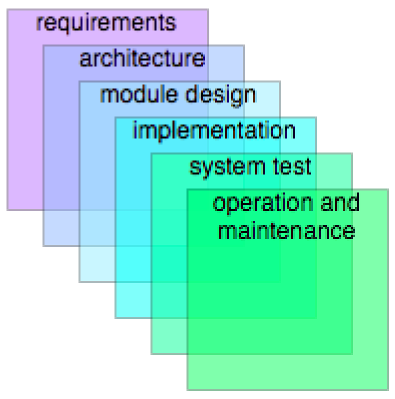
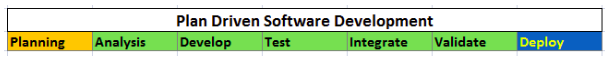
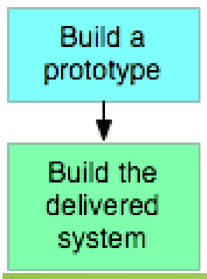
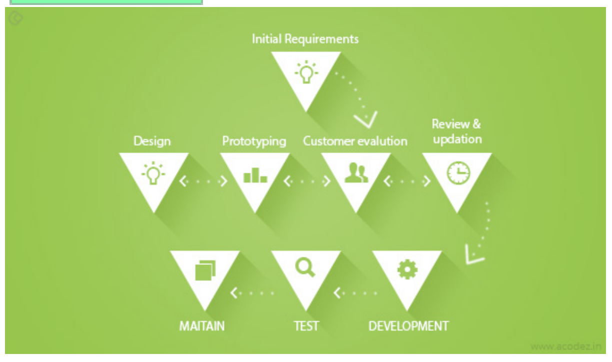
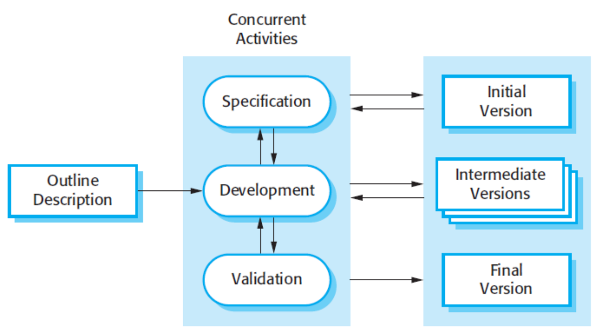
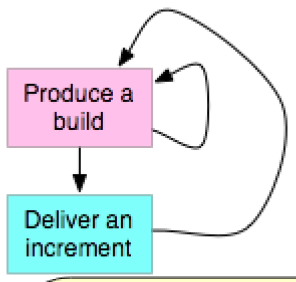
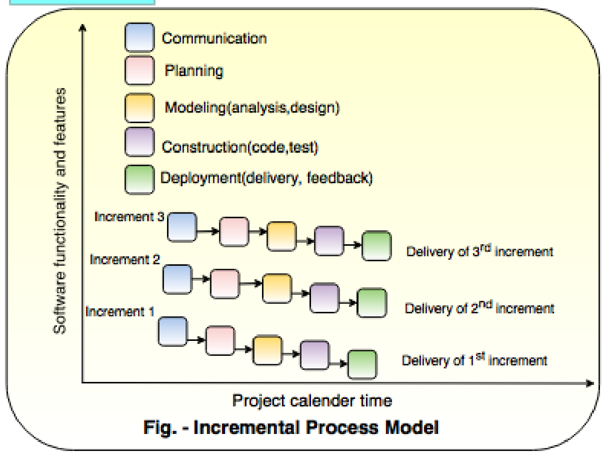
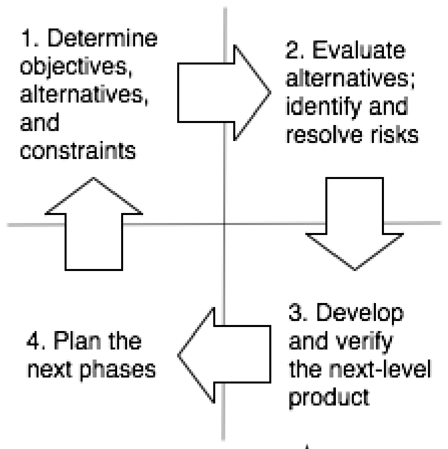
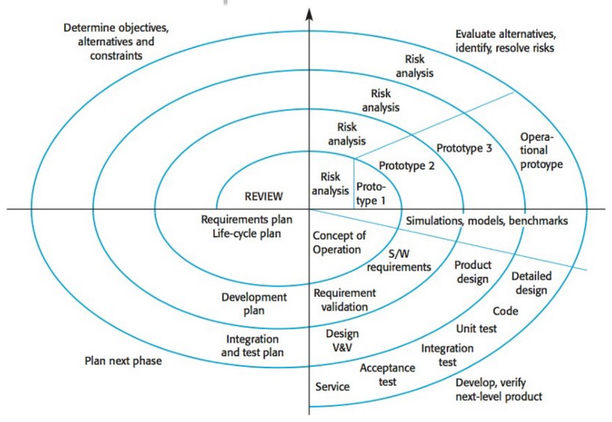
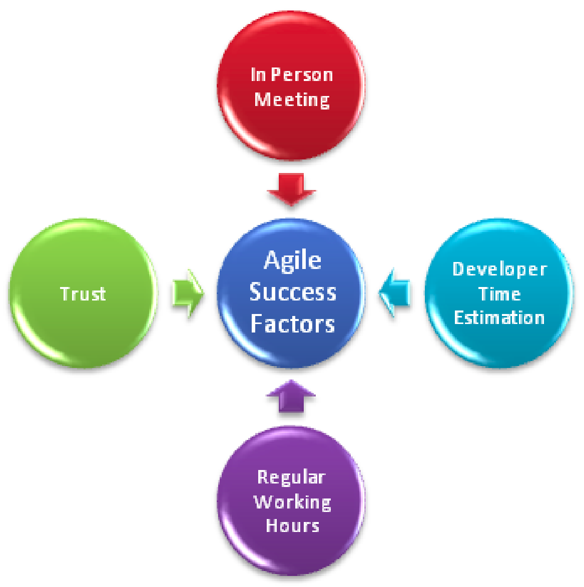
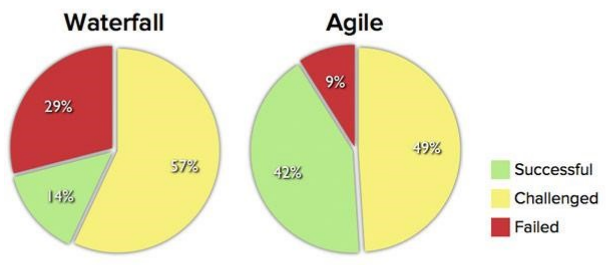
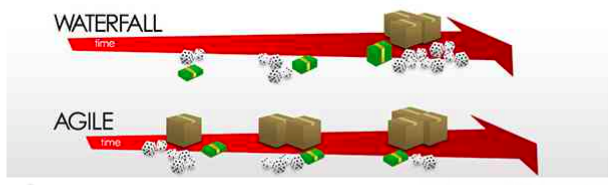
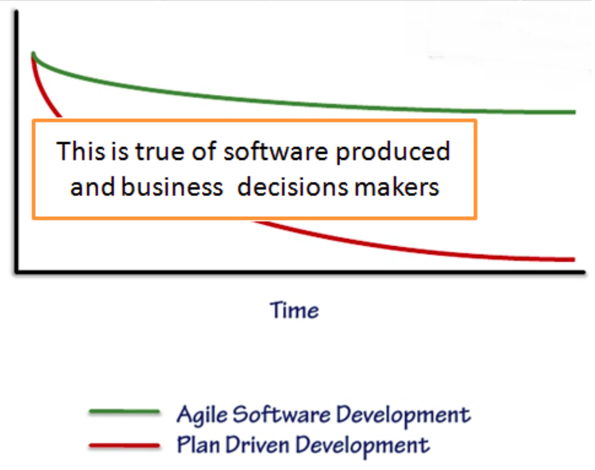
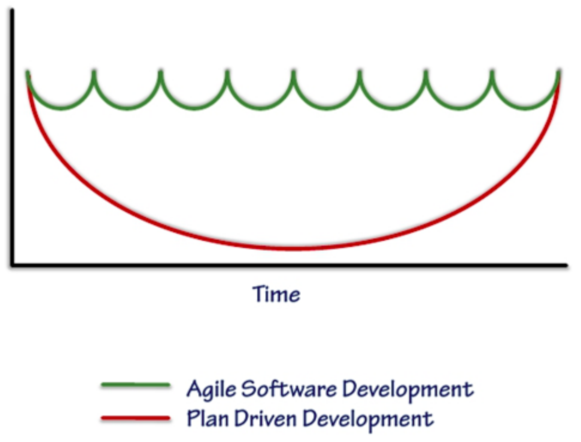
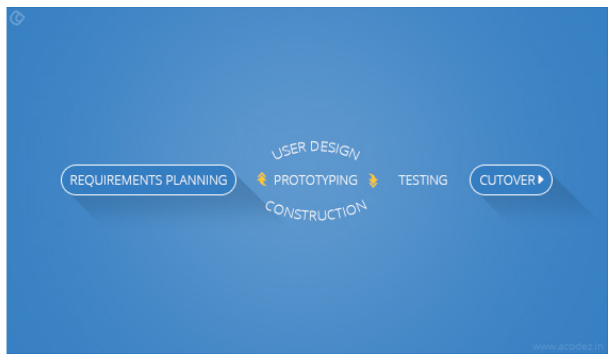
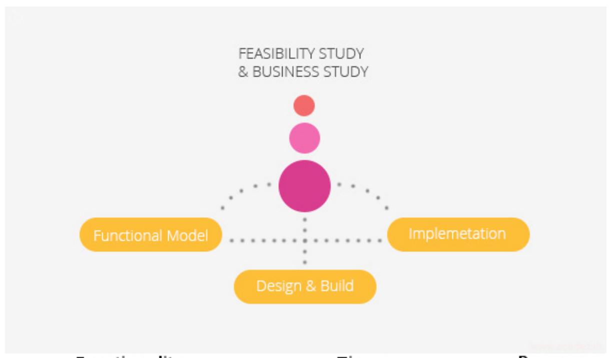
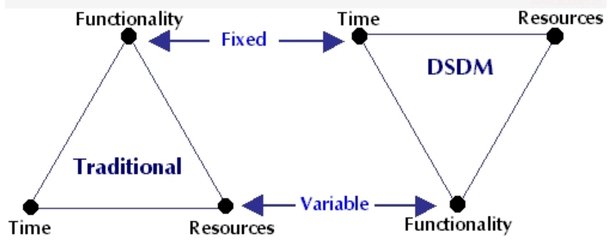
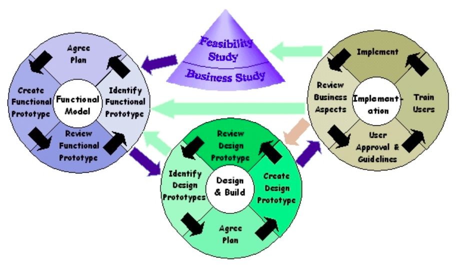
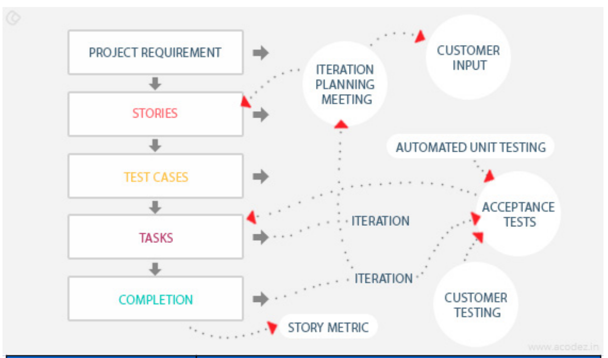
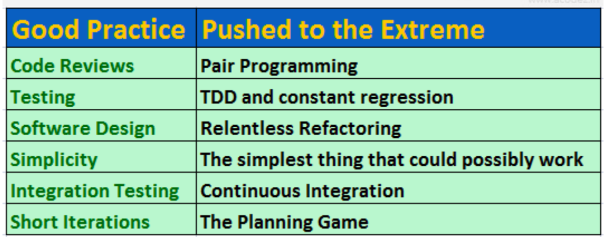
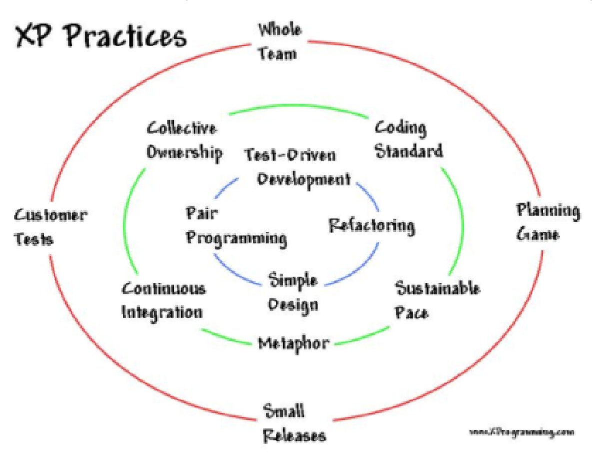
Software Process Model

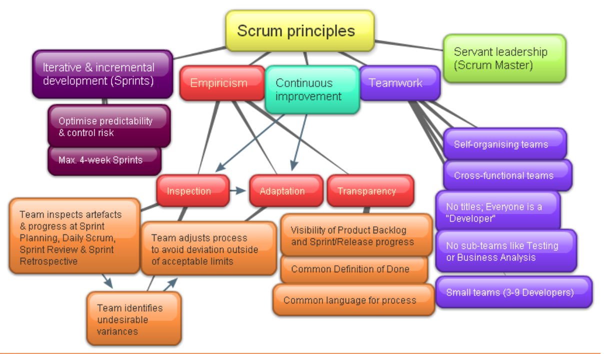
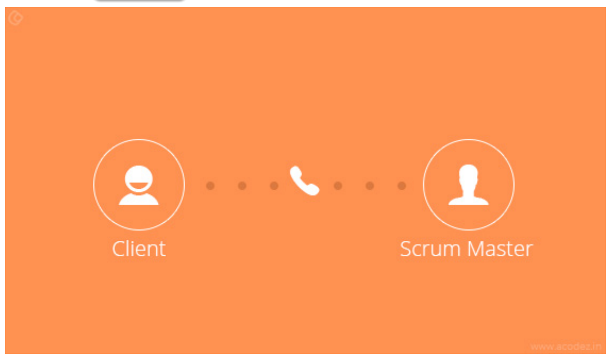
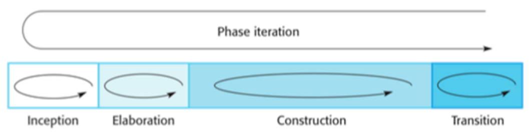
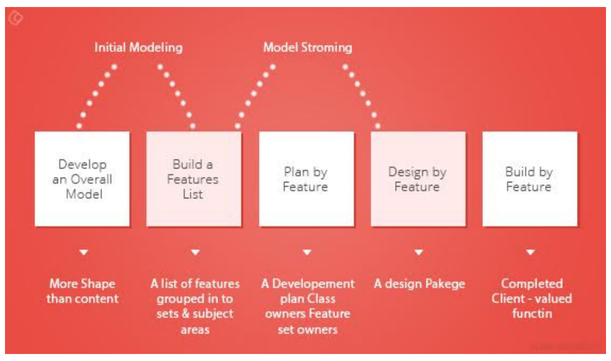
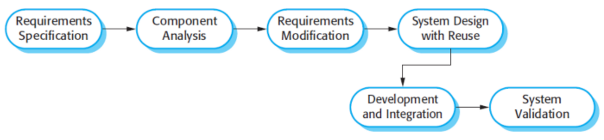
Sr No.	Title	Objective	Pros	Cons	Images
1	Waterfall Model	(first introduced)Sequential Phases of software development;	simple to adapt; best suitable for small projects / very clear requirements; easier to test and analyze	Only matches precise needs; requirements unclear because end-user/client can't see the working software (blurry requirements); high cost for implementing creative work after adapted; hard to estimate/predict cost, deadline, limitation; can't apply to maintenance project	 requirements validation architecture verification module design verification implementation system test operation and maintenance revalidation  Requirement gathering & analysis System Design Implementation Testing Deployment of System maintenance  System Requirements Software Requirements Analysis Program Design Coding Testing Operations 
2	Royce's final model**	improvised waterfall model; feedback->design->requirement	overlapping of stage is possible according to feedback from client		 Evaluation Requirements Analysis Design Development Validation Deployment KEY Waterfall process flow Feedback loop Waterfall with more complex feedback Figure 1: Waterfall model with Royce's iterative feedback.  requirements validation architecture verification module design verification implementation system test operation and maintenance revalidation
3	The Sashimi Model	waterfall model with overlapping phases; most appropriate for medium-sized projects	implementation issue might be discovered during design and implementation	rework needed if requirement changes after design or coding	 requirements architecture module design implementation system test operation and maintenance
4	Plan Driven Development	project success directly depends on how the plan is followed and requirement stability;	less cost to requirement changes before design	costly to adapt to changes after design; highly dependent on requirement stability	 Plan Driven Software Development Planning | Analysis | Develop | Test | Integrate | Validate | Deploy

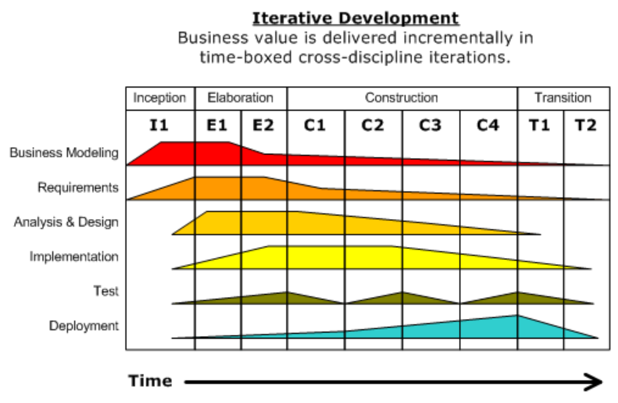
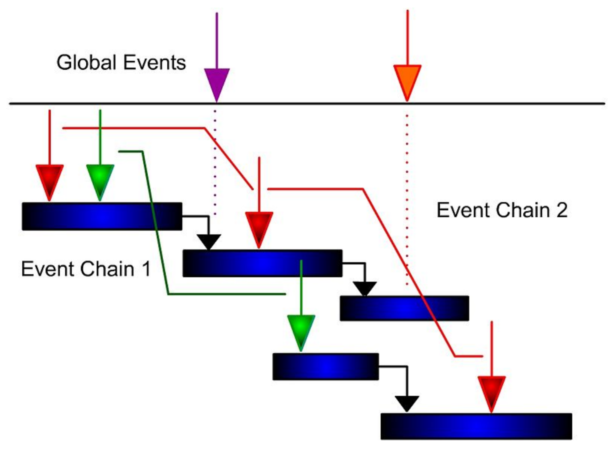
Sr No.	Title	Objective	Pros	Cons	Images
5	Prototyping	initial prototype developed to validate requirements, identify problems and solution for the problems	possible to determine clear idea about functional process of software/product; low failure risk in software functionality	over threshold involvement of client can affect development process; too many changes(patches) affect system flow(i.e. might become hard to debug after too many patches)	  Initial Requirements Design Prototyping Customer evalution Review & updation MAITAIN TEST DEVELOPMENT
6	Incremental Development	evolving software from initial implementation to operational software with several versions of user feedback	user can validate system with respect to requirements delivery at early stages	later stages might become costlier in adapting changes as previous/existing system modules becomes degraded; due addition of new features, system architecture might face issue which may not exist in earlier stages	 Concurrent Activities Specification Development Validation Outline Description Initial Version Intermediate Versions Final Version  Communication Planning Modeling(analysis,design) Construction(code,test) Deployment(delivery, feedback) Increment 3 Increment 2 Increment 1 Delivery of 3rd increment Delivery of 2nd increment Delivery of 1st increment Software functionality and features Project calender time Fig. - Incremental Process Model
7	Spiral	cyclical model; producing prototype in each cycle for testing and risk analysis until software is operational (as per requirements/desire)	risk factors are reduced; suitable for large/complex projects; additional feature can be introduced	costly model for development; final operational may starve as cycle may continue; project can be compromised if failed to evaluate risk analysis properly	 1. Determine objectives, alternatives, and constraints 2. Evaluate alternatives; identify and resolve risks 3. Develop and verify the next-level product 4. Plan the next phases Determine objectives, alternatives and constraints Evaluate alternatives, identify, resolve risks Risk analysis Risk analysis Risk analysis Risk analysis Prototype 3 Prototype 2 Proto-type 1 Opera-tional protoype REVIEW Requirements plan Life-cycle plan Concept of Operation Simulations, models, benchmarks S/W requirements Product design Detailed design Development plan Requirement validation Code Integration and test plan Design V&V Unit test Integration test Acceptance test Service Plan next phase Develop, verify next-level product

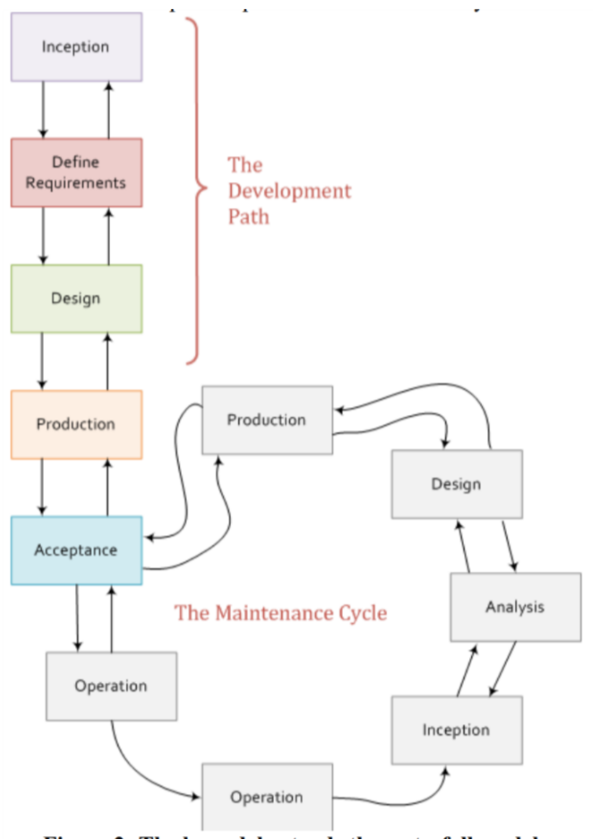
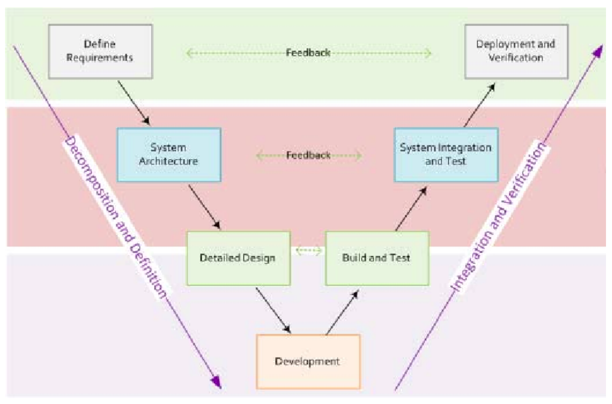
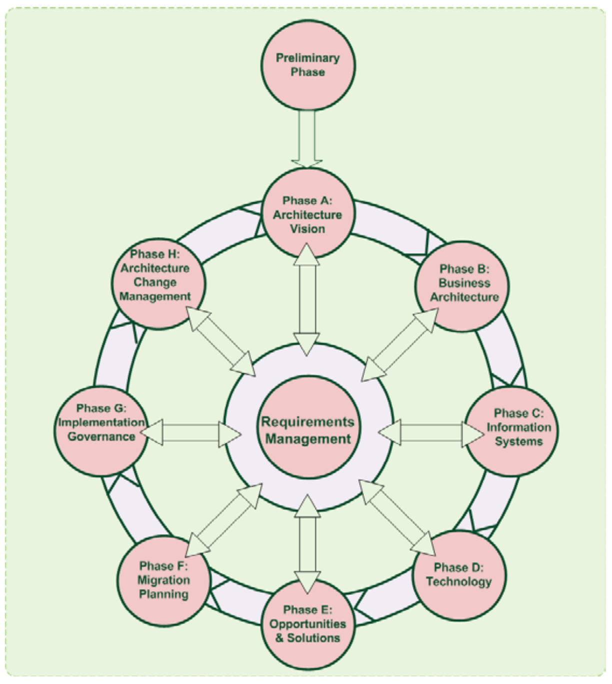
Sr No.	Title	Objective	Pros	Cons	Images
8	Agile development	fast catching up and flexible; targets tech industry, people and collaboration within; focuses on adapting changes and fast turnaround	suitable for creative projects; easy to adapt creative ideas in existing framework/design; transparency maintains	only focus on software work impact on documentation efficiency; path can be diverged as outcome seem unclear	 In Person Meeting Trust Agile Success Factors Developer Time Estimation Regular Working Hours Source: The CHAOS Manifesto, The Standish Group, 2012.  
9	Rapid application development (RAD)	primary focus on providing quick results; a better development process with support of other development approaches	effortless development process; quickly reviewable; ensures client inputs and feedback	depends on overall team performance; requires person with highly adequate knowledge and skills; not suitable for small budget project	 REQUIREMENTS PLANNING USER DESIGN PROTOTYPING CONSTRUCTION TESTING CUTOVER

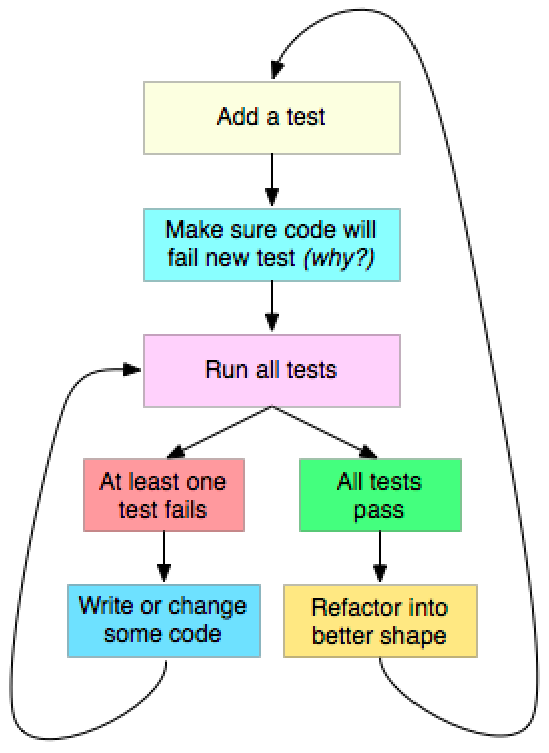
Sr No.	Title	Objective	Pros	Cons	Images
10	Dynamic Software Development Method (DSDM)	derived from RAD; iterative + incremental; focuses on end-user/client involvement; primary focus on system development in deadline and allocated budget	quick delivery of module(functionality); easy access to end user; determine project outcome earlier	expensive implementation; not possible to adapt by small organization;	 FEASIBILITY STUDY & BUSINESS STUDY Functional Model Implemetation Design & Build  Functionality Fixed Time Resources DSDM Traditional Time Resources Variable Functionality The DSDM Development Process  Feasibility Study Business Study Agree Plan Create Functional Prototype Functional Model Identify Functional Prototype Review Functional Prototype Implement Review Business Aspects Implementation Train Users User Approval & Guidelines Review Design Prototype Identify Design Prototypes Design & Build Create Design Prototype Agree Plan
11	Continuous integration	developers work as community and integrate/merge code in shared repository; build/merged changes are tested by automated software	increase workflow process and saves time and effort(code break); bugs and defects can be detected earlier; reduces manual testing effort	some typo mistake can be reveal to end user if it is deployed without manual verification	
12	Continuous Deployment	Continuous integration with automated deployment if codebase passes all test cases	No need of manual deployment; bugs and defects can be detected earlier; quick feedback of system for business	some typo mistake reveal to end user as soon as it is deployed	
13	Extreme Programming (XP)	accentuate teamwork (pair programming); responsive to changing requirements (continuous integration); frequent releases in short development cycles; code refactor, Planning; regular(daily) communication	improve productivity; new requirements can be easily adopted; focuses on user feedback and involvement	efficiency depends on involved people; future possibilities and outcome are unknown; requires regular communication/meeting raises cost	 PROJECT REQUIREMENT STORIES TEST CASES TASKS COMPLETION ITERATION PLANNING MEETING CUSTOMER INPUT AUTOMATED UNIT TESTING ACCEPTANCE TESTS ITERATION ITERATION CUSTOMER TESTING STORY METRIC   XP Practices Whole Team Collective Ownership Test-Driven Development Coding Standard Customer Tests Pair Programming Refactoring Planning Game Continuous Integration Simple Design Sustainable Pace Metaphor Small Releases www.XProgramming.com

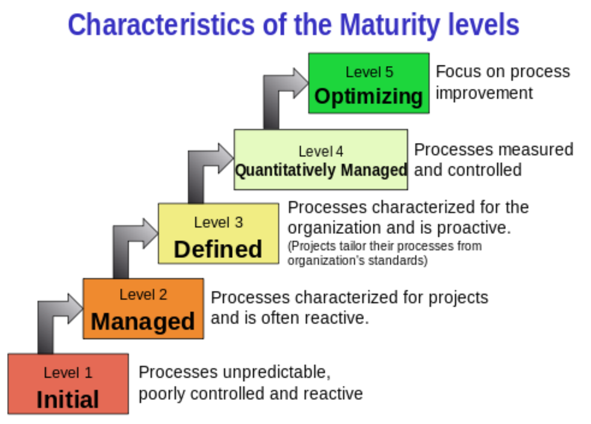
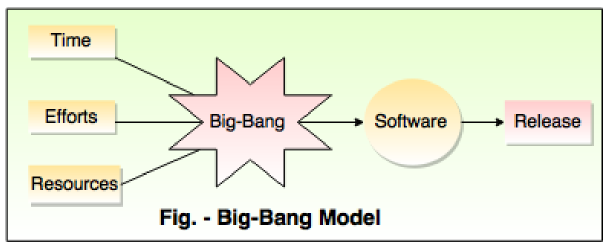
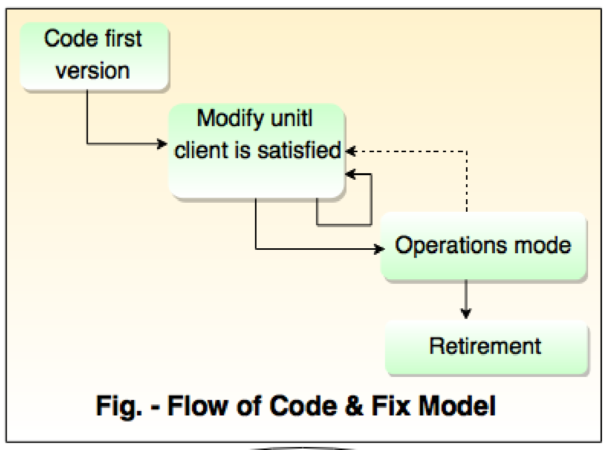
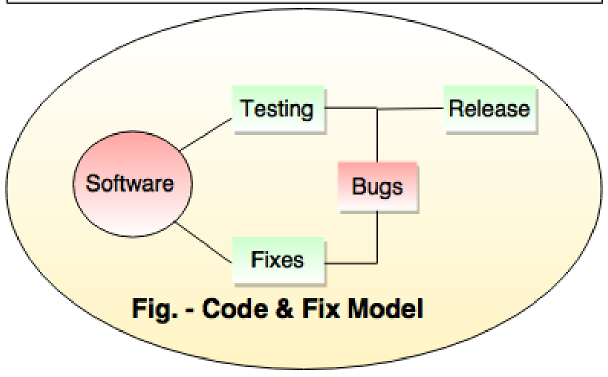
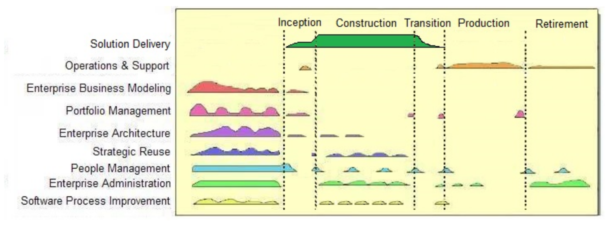
Good Practice	Pushed to the Extreme
Code Reviews	Pair Programming
Testing	TDD and constant regression
Software Design	Relentless Refactoring
Simplicity	The simplest thing that could possibly work
Integration Testing	Continuous Integration
Short Iterations	The Planning Game

Sr No.	Title	Objective	Pros	Cons	Images
14	Scrum	incremental+agile; focuses on short or quick deadline delivery of product;	rapid feedback and quick response to changes; good productivity with adapting changes of requirements and feature; with every iterative feedback/meet/testing non-desire feature/bugs can be fixed easily	as regular changes adapted and integrated project might not be well documented; require more experience of project handling; time frames might extended frequently to adapt changes	 Scrum principles Iterative & incremental development (Sprints) Optimise predictability & control risk Max. 4-week Sprints Empiricism Continuous improvement Teamwork Servant leadership (Scrum Master) Inspection Adaptation Transparency Team inspects artefacts & progress at Sprint Planning, Daily Scrum, Sprint Review & Sprint Retrospective Team adjusts process to avoid deviation outside of acceptable limits Team identifies undesirable variances Visibility of Product Backlog and Sprint/Release progress Common Definition of Done Common language for process Self-organising teams Cross-functional teams No titles; Everyone is a "Developer" No sub-teams like Testing or Business Analysis Small teams (3-9 Developers) 
15	Iterative Development	building small modules of all featured project; build project quickly for user feedback	flexibility for changes; potential bugs/issues can be fixed at earlier stages; progress can be easily tracked	might require more amount of resource; hard to determine deadline; require trained and skilled person for risk analysis	 Phase iteration Inception Elaboration Construction Transition
16	Feature Driven Development	focused on serving large number of teams; suitable for large project	successive progress in large projects; supports continuous update in project; end result always better then initial ones	not suitable for small projects/single person; documentation barely maintained; doesn't guarantee deadline; much more dependable on leading developer	 Initial Modeling Model Stroming Develop an Overall Model Build a Features List Plan by Feature Design by Feature Build by Feature More Shape than content A list of features grouped in to sets & subject areas A Developement plan Class owners Feature set owners A design Pakege Completed Client - valued functin
17	Reuse-oriented Software Engineering	utilizing existing codebase similar to requirement;	system delivery can be quicker;	limited scopes; dependency issue may rise	 Requirements Specification Component Analysis Requirements Modification System Design with Reuse Development and Integration System Validation
18	Critical Chain / Path	focuses to solve resource issues and design for teams of people with flexible skills; determine core element and deadline on its basis	easy to collaborate; tracking and managing resources and developer is easier	each stage plan may extend timeline hence might not suite to small projects	
19	Projects integrating Sustainable Methods (PRiSM)	focuses to account environmental factors; developed by GPM Global; used at large-scale construction(real-estate) projects	shows client about seriousness of developer related to environmental factors	can't work in isolation	
20	Projects IN Controlled Environments (PRINCE2)	de facto standard(convention) used by the UK government since 1996; process oriented; each stage with it's own plan;	heavily documented; suitable for corporate entities	it'll take amount of time to adapt changes (due to heavily documented)	

Sr No.	**Title**	**Objective**	**Pros**	**Cons**	**Images**
21	Lean Development (LD)	empowering team to produce great results with delivering ton of value while producing little project waste*;	low cost budget; good for short deadline and resource; team is motivated to produce perfect feature	depends on decision making (how quickly good decision are made); documentation needs to be precise and understandable	
22	Rational Unified Process (RUP)	Invented by Rational, a division of IBM; iterative waterfall (because it inherits best part of waterfall); 4 phases per project; 9 disciplines per phase[waterfall strategy to moving to next phase]	best of waterflow with iterative approach	too heavily relies on user feedback	 Iterative Development Business value is delivered incrementally in time-boxed cross-discipline iterations. Inception Elaboration Construction Transition I1 E1 E2 C1 C2 C3 C4 T1 T2 Business Modeling Requirements Analysis & Design Implementation Test Deployment Time
23	Event Chain Methodology	relationship between event and task and how they affect each other; focuses on planning for potential risks; handles impact of external events on project;	allows to examine relationship between tasks and external pressures	sometimes beneficial external events might be considered as threat by project manager	
24	Kanban	inherited from lean development; steady stream of product delivery; continuous workflow	easy to track time; concentrates on most important feature; increases speed of development process; constant system improvements	major variation in user requirements; length projects effectively don't get much benefits	
25	Chaos model	attempt towards fixing deadline management issues while fixing bugs and odds of system; problems are divided into levels; upper level -> users' need & bottom level -> define technical resource	flexible & iterative ;	high complexity to use and implement	

Sr No.	Title	Objective	Pros	Cons	Images
26	B-model	Extension to waterfall model; maintenance cycle added to water fall model;	ensures constant improvements of system in development stages		 Inception Define Requirements Design Production Acceptance Operation The Development Path Production Design Analysis Inception Operation The Maintenance Cycle Figure 2: The b-model extends the waterfall model.
27	V-model (vee model)	developed by NASA; molded waterfall model in V shape; allows evolution of user requirements	can utilize for large projects; quality assurance; the requirements & design are verifiable in a SMART (Specific, Measurable, Achievable, Realistic and Timebound) manner; symmetric across two legs	user friendly is non verifiable; limited scope to backed technologies and system	 Define Requirements Feedback Deployment and Verification System Architecture Feedback System Integration and Test Detailed Design Build and Test Development Decomposition and Definition Integration and Verification
28	vee+ model	variation of V-model; adds user involvement; adds user involvement, risks to z-axis of v-model	can identify odds during integration phase; odds are resolved as errors/accepted as design feature	suitable for system with backend and frontend (no business logic)	
29	vee++ model	adds intersecting processes to vee+ model; a decomposition analysis and resolution process is added	utilization scope is not limited and hence suitable for all kind of system		
30	Wheel-and-spoke Model	based on spiral model; initially small teams; wheel represent development cycle; best suitable for multiple projects with identical architecture/feature	negligible initial risk; scalable from small team and task to larger team depending on requirements; core code remains same hence code logic and quality will rise	best suitable for only backed and frontend	 Preliminary Phase Phase A: Architecture Vision Phase B: Business Architecture Phase C: Information Systems Phase D: Technology Phase E: Opportunities & Solutions Phase F: Migration Planning Phase G: Implementation Governance Phase H: Architecture Change Management Requirements Management

Sr No.	Title	Objective	Pros	Cons	Images
31	Joint Application Development (JAD)	collaborative development with customer/involving in design & development phases through workshops(JAD sessions)	feasible for any system development; speed up development and delivery; improves quality of final system	not cost effective (due to JAD sessions); lack of deadline; time commitment needs may not meet if multiple workshops relative to project size	
32	Six Sigma	management framework designed to be driven by data; developed my Motorola	ensures product is 99.99966% error free; suggests improvements before defects even appear	extremely rigid; limits creativity of developer	
33	Test Driven Development (TDD)	all iteration defined by a new test; expects new test fail so that new features implemented and fixed so that it works with old features	improved stability of system; system is well tested and documented	it takes more effort and time while writing test; require code refactor continuously; only few member of team uses TDD	 Add a test Make sure code will fail new test *(why?)* Run all tests At least one test fails All tests pass Write or change some code Refactor into better shape
34	Acceptance Test Driven Development (ATDD)	involves members from different point of view in team to collaborate and write acceptance test; acceptance test act as form of requirements (customer -> what to solve? | Developer -> how to solve | testing -> what about..)	provides interfaces specific to functional testing (without testing through actual UI)	rely on some specific tools	
35	Behaviour Driven Development (BDD)	extension to TDD and ATDD with specification by example;	also suitable and more precise at organizational level; target BDD tool (i.e. FitNesse) support generating technical and end-user document automatically; low cost and risk; eay to undo		

Sr No.	Title	Objective	Pros	Cons	Images
36	Capability Maturity Model (CMM)	developed by Software Engineering Institute (SEI), R&D center sponsored by the U.S. Dept. of Defense (DoD); similar to ISO 9001; framework establishment for continuous process improvement;	five-level evolutionary organized and systematic path; optimize development, acquire, and maintain large-scale software-reliant systems	costly process; not suitable for small organization; require leader with adequate field knowledge	
37	Capability Maturity Model Integration (CMMI)	meet challenges to trending global business; focused on improving usability of maturity models by integrating various different models into single framework	trending process model to satisfy new technology/functionality; Appraisals of organizations using a CMMI model	high cost (In March 2018 CMMI 2.0 introduced at minimum price of US$150.00)	 Characteristics of the Maturity levels Level 5 Optimizing Focus on process improvement Level 4 Quantitatively Managed Processes measured and controlled Level 3 Defined Processes characterized for the organization and is proactive. (Projects tailor their processes from organization's standards) Level 2 Managed Processes characterized for projects and is often reactive. Level 1 Initial Processes unpredictable, poorly controlled and reactive
38	Big-Bang Model	no predefined process is followed; it only requires some planning and need not to follow formal development;	simple; low budget; suitable for small team and project; suitable for academic project, students or new comers	risk is very high; not suitable for long and complex project; the output developed prototype/product does not guarantee initial requirements to be met	 Time Efforts Resources Big-Bang Software Release Fig. - Big-Bang Model
39	Code and Fix Model	it adds testing to big bang model; testing team/developer test the prototype and sends for fixes until it meets the requirement	no burden of project planning; suitable for small project;	Quality assessment is difficult; timeframe for release are not clear; hard to adopt requirement changes	 Code first version Modify unitl client is satisfied Operations mode Retirement Fig. - Flow of Code & Fix Model  Fig. - Code & Fix Model
40	Enterprise Unified Process (EUP)	extended variant of UP(Unified Process) to fulfill lack of production and retirement of software system; embedded to life cycle of organization itself	growing model(growing to Disciplined agile delivery (DAD)); higher quality; high adaptability; seen as end-user point of view		

Sr No.	Title	Objective	Pros	Cons	Images
40	Disciplined agile delivery (DAD) - process decision framework	incremental and iterative delivery of product; grown from Enterprise Unified Process (EUP); developed after 2013 with adopting best from many development process like agile/scrum/lean/etc.;	2nd generation framework; end-to-end method for agile delivery; hybrid agile; high scalability; goal driven; enterprise level; provides more affilated approach to agile software development; fills the gaps ignored by scrum; capable at enterprise level scale		

Summary

I am going to pursue Software as a Service for web portal, management of product sale, purchase and inventory. Such a product has a very large scope to develop in future and because it's target many field for generic product the system needs continuous testing development and adapting new requirements along with upcoming technologies hence the best suitable model for this project is Disciplined agile delivery (DAD)

18MCEC10 - Gahan M. Saraiya
Student, M. Tech, Nirma University