Name	When to use	Point of View	Requirements method	Schedule definition	Build	Testing	Delivery approach	Closing approach
DMAI-Ban **Blend of DMAIC and Kanban**	-Customer somewhat involved -Process driven -Team members are specialized, only used in certain stages or deliverables -Stories can be paused as needed without causing significant domino effect	Organization efficiency	-Requirements and development of the scope follows the defined DMAIC process - Define, Measure, Analyze; they are written in statistical language	-Team defines the lifecycle stages -Scheduling based on team capacity and area of expertise while considering team WIP limits and customer expectations -Tasks are determined at a constant flow	-Scope prioritized by the team based on order of precedence, importance, availability and expertise of team resources -Constant flow of work pulled by the team	-Discrete Testing lifecyle stage identified. -Continual testing once scope is developed/built. -Items requiring modification return to Develop/Build lifecycle stage	-Deliverables released throughout the project as defined by the lifecycle stages -Control measures are defined and built as a lifecycle stage to be deployed along with or after the applicable scope	-Stabilization can be a discrete lifecycle stage -Defects return back to build stage -Control measures exist and are monitored in "monitor" lifecycle stage. -Documentation, lessons learned, solution acceptance and project closure all included within lifecycle stages, either as one stage or discrete. -A "Done" stage contains all completed work.
DMAI-Fall **Blend of DMAIC and Waterfall**	-Intermittent customer involvement -Process driven -Can include traditional vendor RFP -May only have one big bang delivery	Organization efficiency	-Requirements and development of the scope follows the defined DMAIC process - Define, Measure, Analyze; they are written in statistical language	-Sequential task planning, considering dependencies, team allocations and other standard waterfall planning -managed in traditional project management schedule or spreadsheet	-Traditional waterfall: Design/Build/Develop	-Will have defined test plan to include different testing types -Full end to end testing done prior to acceptance	-Big Bang delivery, which may have several smaller delivery points after for supporting functionality -Measures of Control are also defined	-Control measures exist and are monitored. -Stabilization, documentation, standard closing activities done at end of the project -Defects addressed and prioritized in closing phase
DMAI-Scrum **Blend of DMAIC and Scrum**	-Very involved customer -Process driven -Higher number of unknowns -Deliverables can be deployed independent of one another	Organization efficiency	-Requirements and development of the scope follows the defined DMAIC process - Define, Measure, Analyze; they are written in statistical language	-Scope time-boxed for controlled releases in Sprints with an overall release plan, providing customer with end-date/delivery timeframes	-Team collaborates together on all aspects of creating the scoped deliverables -Product owner (customer) prioritizes the stories -Creation of control measures if no control Sprint	-Team defines "how to demo" information to test each item contained in the timebox -Continual testing as stories are developed/built per defined acceptance criteria -Items requiring modification move to future Sprint	-Iterative demos and deployments -Customer feedback/refactoring anticipated -Control measures are deployed along with the applicable scope	-Control measures exist and are monitored in subsequent Sprints post-deployment -Defects move to future sprints -Documentation during build Sprint or during later documentation Sprint -Standard closing activies done as closing Sprints at the end of the project
Kan-Fall **Blend of Kanban and Waterfall**	-Team members are specialized, only used in certain stages -Customer somewhat involved -Some amount of unknowns -Stories may be paused as needed without causing significant domino effect - Infrastructure and development blend	Users (does not include "why")	-Requirements are written as user stories. -User stories are written from the business users perspective and DO NOT include "why". -Business forecast driven and are written in business user language	-Team defines the lifecycle stages most scope items need to follow. -Rough estimates and timing identified based on relative sizing, team WIP limits and customer expectations	-Lifecycle stages include Design, Build, Develop -Constant flow of work pulled by the team in planned sequence	-Will have defined test plan to include different testing types -Full end to end testing done prior to acceptance -Stories typically DO NOT include "how to demo"/test	-Big Bang delivery, which may have several smaller delivery points after for supporting functionality	-Stabilization, documentation, standard closing activities done at end of the project -Defects addressed and prioritized in closing phase
Kan-Scrum **Blend of Kanban and Scrum**	-Team members are specialized, only used in certain stages -Customer is heavily involved -Higher number of unknowns -Stories can be paused as needed without causing significant domino effect	Users (does not include "why")	-Requirements are written as user stories. -User stories are written from the business users perspective and DO NOT include "why" -Business forecast driven and are written in business user language	-Scope time-boxed for controlled releases in Sprints with an overall release plan, providing customer with end-date/delivery timeframes	-Team collaborates together on all aspects of creating the scoped deliverables -Product owner (customer) prioritizes the stories	-Team defines "how to demo" information to test each item contained in the timebox -Continual testing as stories are developed/built per defined acceptance criteria -Items requiring modification move to next Sprint	-Iterative demos and deployments -Customer feedback/refactoring anticipated	-Defects move to future sprints - Documentation during build Sprint or during later documentation Sprint - Standard closing activies done as closing Sprints at the end of the project

Name	When to use	Point of View	Requirements method	Schedule definition	Build	Testing	Delivery approach	Closing approach
Scrum-Ban **Blend of Scrum and Kanban**	-Team members are specialized, only used in certain stages -Customer heavily involved -Higher number of unknowns -Stories can be paused as needed without causing significant domino effect	Users (includes "why")	-Requirements are written as user stories -User stories are written from the users perspective and include "why" -Stories include acceptance criteria -Priorities are set by customer	-Team defines the lifecycle stages -Scheduling based on team capacity and area of expertise while considering team WIP limits and customer expectations -Tasks are determined at a constant flow	-Led by the customers, user stories are prioritized based on order of precedence, importance, availability and expertise of team resources -Constant flow of work pulled by the team	-Discrete Testing lifecyle stage identified -Continual testing as stories are developed/built per defined acceptance criteria -Items requiring modification return to Develop/Build lifecycle stage	-Deliverables released throughout the project as defined by the lifecycle stages	-Stabilization can be a discrete lifecycle stage -Defects return back to build stage -Documentation, lessons learned, solution acceptance and project closure all included within lifecycle stages, either as one stage or discrete. -A "Done" stage contains all completed work.
Scrum-Fall **Blend of Scrum and Waterfall**	-Vendor with RFP in user stories -Customer somewhat involved -Only one big bang delivery -Fewer number of unknowns -Infrastructure and development blend	Users (includes "why")	-Requirements are written as user stories -User stories are written from the users perspective and include "why" -Stories include acceptance criteria	-Stories organized in a release plan specifically outlined on a timeline and may include Sprint time boxing	-Traditional waterfall: Design/Build/Develop potentially done in Sprints	-Will have defined test plan to include different testing types -Full end to end testing done prior to acceptance -Stories typically include "how to demo"/test	-Big Bang delivery, which may have several smaller delivery points after for supporting functionality	-Stabilization, documentation, standard closing activities done at end of the project -Defects addressed and prioritized in closing phase
Water-Ban **Blend of Waterfall and Kanban**	-Traditional vendor RFP -Team members are specialized, only used in certain stages -Customer not as heavily involved	Functional Specification	-Requirements are known upfront and written as functional and sometimes technical specifications -Requirements are translated to user stories and DO NOT include "why"-	-Team defines the lifecycle stages -Scheduling based on team capacity and area of expertise while considering team WIP limits and customer expectations -Traditional methods of defining tasks up front over a defined timeline	-Deliverables prioritized by the team based on order of precedence, importance, availability and expertise of team resources -Constant flow of work pulled by the team	- Discrete Testing lifecyle stage identified. - Continual testing once scope is developed/built. - Items requiring modification return to Develop/Build lifecycle stage	-Deliverables released throughout the project as defined by the lifecycle stages -Control measures are defined and built as a lifecycle stage to be deployed along with or after the applicable scope	-Stabilization can be a discrete lifecycle stage -Defects return back to build stage -Control measures exist and are monitored in "monitor" lifecycle stage. -Documentation, lessons learned, solution acceptance and project closure all included within lifecycle stages, either as one stage or discrete. -A "Done" stage contains all completed work.
Water-Scrum **Blend of Waterfall and Scrum**	-May include traditional vendor RFP -High level requirements defined -Details for implementation have a higher level of unknowns -Deliverables can be deployed independent of one another	Functional Specification	-Requirements are known upfront and written as functional and sometimes technical specifications -Requirements are translated into user stories including the "why" and acceptance criteria -Priorities are set by customer	-Scope time-boxed for controlled releases in Sprints with an overall release plan, providing customer with end-date/delivery timeframes	-Team collaborates together on all aspects of creating the scoped deliverables -Product owner (customer) prioritizes the stories	-Team defines "how to demo" information to test each item contained in the timebox -Continual testing as stories are developed/built per defined acceptance criteria -Items requiring modification move to future Sprint	-Iterative demos and deployments -Customer feedback/refactoring anticipated	-Defects move to future sprints - Documentation during build Sprint or during later documentation Sprint - Standard closing activies done as closing Sprints at the end of the project