	Bules				Git Workflows					
		Rules		GitHub Flow	Feature Branch	GitFlow	Trunk-based Development	Cactus Model		
		A) Clone					REC			
	a) Repository creation	B) GitHub-style Fork								
			A) Master	1	1	1	/	1		
			B) Develop			✓		1		
			C) Feature							
		i. Main [4]	D) Topic							
			E) Production							
			F) Release					1		
			A) Master			/		1		
			B) Develop			✓		1		
			C) Feature					1		
			D) Topic					1		
		ii. Integration	E) Production					1		
			F) Release					1		
			G) Fix					1		
			H) HotFix					1		
			I) BugFix					1		
			A) Master		/		/	1		
			B) Develop	1	/		/	1		
			C) Feature	1	1	/	/	1		
			D) Topic	1	/	1	/	1		
		iii. Change	E) Production	1	/		/	1		
	b) Branch role assignment [3]		F) Release	1	/		/	1		
			G) Fix	1	/		/	1		
			H) HotFix	1	1		/	1		
			I) BugFix	1	/		/	1		
			A) Master	1		✓	/	1		
			B) Develop							
			C) Feature							
1. Setup [1]			D) Topic							
		iv. Release	E) Production							
			F) Release			1	1	1		
			G) Fix					1		
			H) HotFix							
			I) BugFix							
			A) Master							
			B) Develop	1						
			C) Feature	1						
			D) Topic	1						
		v. Fix	E) Production	1						
			F) Release	1						
			G) Fix	1	1	✓				
			H) HotFix	1		✓				
			I) BugFix	1		✓				
			A) Central repository			REQ				
		i. Main	B) Local repository							
			C) Forked repository							
			A) Central repository			REQ				
		ii. Integration	B) Local repository							
			C) Forked repository							
1			A) Central repository							

	Types of guidelines								
Key	Key Label Description								
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)							
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)							
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)							

C) Davebopment environment El. Change Bl. Local repository REC						Git Workflows					
Integration			Rules				GitFlow				
C) Forted repository C) Forted repository			iii. Change	B) Local repository	REC	REO	REC	REC	REO		
N. Release A. Central repository C. Forest repository C. For		environment			1,20			1,20			
W. Release B) Local repository											
C) Froked repository			iv Release								
V, Fix			Tr. Holodoo								
V, Fix B) Local repository REC											
			v Fiv	<u> </u>	DEC						
A To work on a new feature			V. I IA		REC						
Integration [7] C) To prepare a release											
D) To integrate work			ii. Integration [7]								
a) Creation event [6] iii. Change [8] iii. Change [8] Diri creates a fix C) To prepare a release D) To integrate work D) To							PEO				
a) Creation event [6] B) To create a fix C) To prepare a release D) To integrate work D) To integrate wo					BEO	BEO		DEC	DEO		
a) Creation event [6] C) To prepare a release D) To integrate work C) To prepare a release D) To integrate work C) To prepare a release C) To integrate work C) To integrate					REQ	REQ	REQ	REC	REQ		
D) To Integrate work D) To Integrate work			iii. Change [8]								
A) Treation event [6]											
N. Release [9] B) To create a fix		a) Creation event [6]				 					
N. Release [9] C) To prepare a release REQ REC REQ						-					
D) To integrate work			iv. Release [9]				DEO	DEO	DEO		
A) To work on a new feature B) To create a fix							REQ	REC	REQ		
V. Fix [10] B) To create a fix											
V. Fix [10]				*	DEO		DEO				
D) To integrate work REQ			v. Fix [10]		REQ		REQ				
ii. Integration iii. Change iii. Integration iii. Change iii. Integration iii. Change iii. Ch											
ii. Integration iii. Change											
I. Main REQ			:: 1-4				REQ				
b) Parent branch [11] ii. Change ii. Integration REQ REQ REQ REQ REQ REQ			II. Integration								
Description											
iv. Release PRO REQ REQ					REQ	REQ		REQ	REQ		
b) Parent branch [11] i. Main REQ REQ REQ			III. Change								
i. Release ii. Integration REC REC iii. Change REC ii. Change REC ii. Main REC iii. Change REC iii. Change REQ iv. Release REQ iv. Release REQ REQ REQ iii. Change REQ iv. Release REQ REQ iv. Release REQ REQ iii. Change REQ iv. Release REQ REQ iv. Release REQ REQ iv. Release REQ REQ iii. Integration iii. Change REQ REQ iv. Release REQ REQ iv. Rele							PRO				
Iii. Change		b) Parent branch [11]	l					REQ	REQ		
2. Branching strategy [5] 2. Broken code is not allowed 2. Broken code is not allowed 3. Main 3.			IV. Release				REC				
2. Branching strategy [5] c) Broken code is not allowed ii. Main ii. Integration iii. Change iv. Release REQ REQ REQ REQ REQ REQ REQ RE											
V. Fix							REC				
2. Branching strategy [5] C Broken code is not allowed I. Main I. Integration II. Change II. Main II. Integration II. Change II. Main II. Integration III. Change III. Chang			v. Fix								
2. Branching strategy [5] C Broken code is not allowed I. Main											
Strategy [5] C Broken code is not allowed REQ REQ REQ	2. Branching			IV. Release							
C) Broken code is not allowed III. Change III. Chang					REQ	REQ	REQ	REQ			
A Short-lived A Short-		c) Broken code is not									
v. Fix								_			
i. Main					REQ			REQ			
ii. Integration											
d) Production-ready					REQ		REQ	REQ			
12		d) Production-ready									
v. Fix i. Main A) Short-lived E) Long-lived REC REQ						-					
i. Main					REQ			REQ			
I. Main B) Long-lived REC REQ REQ			v. Fix	Terran							
B) Long-lived REC REQ REQ ii Integration A) Short-lived			i. Main								
lii Integration							REC	REQ	REQ		
B) Long-lived REC			ii. Integration								
	1			B) Long-lived			REC				

	Types of guidelines								
Key	Key Label Description								
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)							
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)							
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)							

				Git Workflows						
	Rules		GitHub Flow	Feature Branch	GitFlow	Trunk-based Development	Cactus Model			
		A) Short-lived		REC	REC	REC	REC			
e) Lifetime	iii. Change	B) Long-lived								
		A) Short-lived			REC	REC				
	iv. Release	B) Long-lived			ILLO	TLEO				
		A) Short-lived			REC					
	v. Fix	B) Long-lived			ILLO					
		A) Integration completed								
	i. Main	B) Additional tasks completed								
		A) Integration completed								
	ii. Integration	B) Additional tasks completed								
		A) Integration completed			REC	REC	REQ			
f) Delete branch after [13]	iii. Change				REC	REC	REQ			
[10]		B) Additional tasks completed								
	iv. Release	A) Integration completed			REC					
		B) Additional tasks completed			REC					
	v. Fix	A) Integration completed			REC					
		B) Additional tasks completed								
		A) End of working day								
		B) Every <n> weeks</n>								
		C) When work completed								
	i. Main	D) After rebase								
		E) After code review completed								
		F) After tests passing in Dev Env.								
		G) After tests passing in Test/QA/Stage								
		H) After tests passing in Production								
		I) To update the target branch								
		J) As frequently as possible					REC			
		A) End of working day								
		B) Every <n> weeks</n>								
		C) When work completed								
		D) After rebase								
		E) After code review completed								
	ii. Integration	F) After tests passing in Dev Env.								
		G) After tests passing in Test/QA/Stage								
		H) After tests passing in Production								
		To update the target branch								
		J) As frequently as possible								
		A) End of working day								
		B) Every <n> weeks</n>								
		C) When work completed			REQ					
		D) After rebase								
a) Code integration	iii. Change	E) After complete code review	REC	REC		REQ	REC			
event [15]	iiii Gridingo	F) After tests passing in Dev Env.	REC			REQ				
		G) After tests passing in Test/QA/Stage								
		H) After tests passing in Production	REC							
		I) To update the target branch								
		J) As frequently as possible	REQ			REC				
		A) End of working day								
		B) Every <n> weeks</n>								
		C) When work completed								
		D) After rebase								
1	1	E) After code review completed		+	REQ					

	Types of guidelines								
Key	Key Label Description								
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)							
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)							
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)							

				Git Workflows					
		Rules		GitHub Flow	Feature Branch	GitFlow	Trunk-based Development	Cactus Model	
		iv. Release	F) After tests passing in Dev Env.						
			G) After tests passing in Test/QA/Stage						
			H) After tests passing in Production						
			I) To update the target branch		1				
			J) As frequently as possible						
			A) End of working day						
			B) Every <n> weeks</n>						
			C) When work completed			REQ			
			D) After rebase						
			E) After code review completed	REC					
		v. Fix	F) After tests passing in Dev Env.	REC					
			G) After tests passing in Test/QA/Stage						
			H) After tests passing in Production	REC					
			I) To update the target branch						
			J) As frequently as possible						
		:: l=t====#==	A) Upstream						
		ii. Integration	B) Downstream						
		Ob	A) Upstream				REC		
	b) Main integrated into	iii. Change	B) Downstream					REQ	
		iv. Release	A) Upstream						
		IV. Release	B) Downstream						
		v. Fix	A) Upstream						
		V. FIX	B) Downstream						
		i. Main	A) Upstream						
		i. ividiii	B) Downstream						
		iii. Change	A) Upstream						
	c) Integration		B) Downstream						
	integrated into [16]	iv. Release	A) Upstream						
			B) Downstream						
		v. Fix	A) Upstream						
			B) Downstream						
		i. Main	A) Upstream	REQ	REQ		REQ	REQ	
			B) Downstream						
3. Code		ii. Integration	A) Upstream			REQ			
integration			B) Downstream		DEO				
strategy [14]	d) Change integrated into [17]	iii. Change	A) Upstream B) Downstream		REC				
			A) Upstream	REQ					
		iv. Release	B) Downstream	REQ					
			A) Upstream						
		v. Fix	B) Downstream						
			A) Upstream			REQ		REC	
		i. Main	B) Downstream			TLEG		INLO	
			A) Upstream			REQ			
	e) Release integrated	ii. Integration	B) Downstream						
	into [18]		A) Upstream						
		iii. Change	B) Downstream		+ +				
			A) Upstream						
		v. Fix	B) Downstream						
			A) Upstream	REQ		REQ			
		i. Main	B) Downstream						
1	1		•				-		

	Types of guidelines							
Key	Key Label Description							
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)						
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)						
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)						

				Git Workflows					
	Rules		GitHub Flow	Feature Branch	GitFlow	Trunk-based Development			
		A) Upstream			REQ		_		
f) Fix integrated into	ii. Integration	B) Downstream							
[19]		A) Upstream	REQ						
	iii. Change	B) Downstream	1.24						
		A) Upstream			REC				
	iv. Release	B) Downstream							
		A) fast-forward merge							
		B) non-fast forward merge							
		C) rebase							
	i. Main	D) cherry-pick							
		E) fetch with rebase and merge							
		F) fetch and merge							
		A) fast-forward merge							
		B) non-fast forward merge							
		C) rebase							
	ii. Integration	D) cherry-pick							
		E) fetch with rebase and merge							
		F) fetch and merge							
	iii. Change	A) fast-forward merge		1		REC			
		B) non-fast forward merge			REQ	REC			
g) Code integration		C) rebase		REC		1.20			
mechanism [20]		D) cherry-pick		1.20					
		E) fetch with rebase and merge							
		F) fetch and merge		1					
		A) fast-forward merge		1					
		B) non-fast forward merge			REQ				
		C) rebase							
	iv. Release	D) cherry-pick		+ +		REC			
		E) fetch with rebase and merge				1,20			
		F) fetch and merge							
		A) fast-forward merge							
		B) non-fast forward merge			REQ				
		C) rebase		1					
	v. Fix	D) cherry-pick							
		E) fetch with rebase and merge		1					
		F) fetch and merge		1					
	A) force	i , i etan eme merge							
h) Merge options [21]	B) force-with-lease								
, . 5	C) no options								
	,	A) Staged							
	i. Main	B) At central repository							
		C) Through pull/merge request							
		A) Staged							
	ii. Integration	B) At central repository		 		1			
		C) Through pull/merge request		+ +					
i) Codo rovious		A) Staged		REC		REC			
i) Code review approach before	iii. Change	B) At central repository				1.20			
integrating from [22]		C) Through pull/merge request	REC	REC					
		A) Staged	INLO	1,50		+			
	iv. Release	B) At central repository		+ +		1			
	· · · · · · · · · · · · · · · · · · ·	12,7 tt ochtrar republitory	1			_1			

	Types of guidelines								
Key	Key Label Description								
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)							
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)							
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)							

		Rules				Git Workflows	5	
	Rules			GitHub Flow	Feature Branch	GitFlow	Trunk-based Development	Cactus Model
			A) Staged					
		v. Fix	B) At central repository					
			C) Through pull/merge request	REC				
	a) Commit message guidelines [24]	A) GitHub convention						
		i. Main	A) semantic versioning (semver)					
		ii. Integration	A) semantic versioning (semver)					
		ii. iiitegration	B) develop			REQ		
		iii. Change	A) semantic versioning (semver)					
1.	[25]	iv. Release	A) semantic versioning (semver)			REC		
 Development 		IV. Release	B) release-*			REQ		
conventions		v. Fix	A) semantic versioning (semver)			REC		
[23]		V. FIX	B) hotfix-* bugfix-* fix-*			REQ		
		i. Main				REQ		
		ii. Integration						
	c) Tags used [26]	iii. Change						
		iv. Release						REC
		v. Fix						
	d) Version bump used	[27]	·			REQ		

	Types of guidelines								
Key	Key Label Description								
REQ	Requirement	Guideline enforced by the workflow, non-optional (MUST, SHALL)							
REC	Recommendation	Suggested guideline that if ignored does not affect the main purpose of the worflow, optional (SHOULD)							
PRO	Prohibition	Guideline that is explicitly forbidden (MUST NOT, SHALL NOT)							

- [1] What steps are needed to prepare the project for the development process from the contributor point of view? (e.g., how to setup the initial working copy to start contributing to the project?)
- [2] What mechanism (clone or GitHub fork) is used to create the working copy for a contributor to the project?
- [3] In the workflow, which branch assumes the role of each these types of branches?
- [4] Contains all the changes associated with the last stable release (other names: Master, Trunk)
- [5] Which conventions are followed to decide on how branches should be used to align with the conventions, requirements and objectives of the development project? (e. g., what are the types of branches used along the development process?)
- [6] When a contributor should create a new branch?
- [7] Allows the integration of multiple changes that are not yet ready to be released (other names: Develop)
- [8] All work associated with the development of a new feature (other names: Feature, topic)
- [9] Once the changes are completed, tested and integrated they can be included in this branch, which contains code that is ready for production (other names: Production)
- [10] Include development work to correct detected bugs and other emergency fixes (other names: HotFix, BugFix)
- [11] From which branch the contributors should branch off?
- [12] Should the branch be at all times ready for production (i.e., containing only code that compiles, with all tests passing, and fully integrated)?
- [13] Under what condition(s) should the branch be removed?
- [14] What approach or mechanism is followed to merge branches back? (e.g., what triggers a merge operation in the development process and which branches participate in it?)
- [15] At which point a branch should be merged back (e.g., when the work is ready for production, when a rebase has been applied, when the work in the feature is completed, every <n> number of weeks)?
- [16] Which branch should a Working branch be merged back to?
- [17] Which branch should a Change branch be merged back to?

- [18] Which branch should a Release branch be merged back to?
- [19] Which branch should a Fix branch be merged back to?
- [20] What mechanism (e.g., fast-forward merge, non-fast-forward merge, rebase, cherry-pick, fetch with rebase and merge, fetch and merge) should be used for merging a branch back?
- [21] How changes that are intended to be merged should be pushed to the remote repository (e.g., with force option, with force-with-lease option, without options)?
- [22] Should pull requests be used before merging back a branch?
- [23] What additional guidelines are requested from contributors to the development project that are not specifically related to any of the previous categories? (e.g., what naming conventions are followed for the final release of the software product?)
- [24] Which naming convention (e.g., GitHub or other) should be followed for commit messages?
- [25] Which naming convention should be followed for a branch?
- [26] Are tags used to identify a branch?
- [27] Is version bump used to mark the creation of a new version?