DBRG Evaluation

July 1, 2021

Table.1 shows how the proposed format is customised/controlled to serve in generating requirement with quality issues for each of the identified indicators in Table.3. The table lists the indicator code, an example of controlled input format, and the corresponding obtained output by our DBRG (i.e., completing the missing parameters through generating the textual requirement under the controlled format). It is also worth noting that, Prolog backtracking and matching nature helps greatly in achieving a very high level of flexibility for controlling the requirement format through the generation process.

Table 1: Example Input format of Multi-Level Indicators and the Corresponding generated Output

ID	Example of input format and Generator Output	
I1:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', ['', Head, '', 'Key access and control', '', V1, [P1]], CompText)], AL, [[], ['','','','',''], Text)]
	Out	Head = if , V1 = equals , P1 = prep('', '[True]') , CompText = "if Key access and control equals [True] ", AL = [inComp('A1', 'Act', ['', '', '', 'the regulator init timeout', '', 'shall be intialised', [...]], "the regulator init timeout shall be initialised to [True] ")], Text = "– , , if Key access and control equals [True] , the regulator init timeout shall be initialised to [True] , ."
I2:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', ['', Head, '',E1, 'is ON or equals OFF', '',[]], CompText)], AL, [[], ['','','','',''], Text)]
	Out	Head = 'provided that', E1 = 'the manage monitor interface mode', CompText = "provided that the manage monitor interface mode is ON or equals OFF ", AL = [inComp('A1', 'Act', ['', '', '', 'the Reset', '', 'shall be initialised', [...]], " the Reset shall be initialised to [True] ")], Text = "– , , provided that the manage monitor interface mode is ON or equals OFF , the Reset shall be initialised to [True] , ."
I3:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', ['', Head, '',E1, 'is ON or OFF', '',[]], CompText)], AL, [[], ['','','','',''], Text)]
	Out	Head = 'provided that', E1 = 'the manage monitor interface mode', CompText = "provided that the manage monitor interface mode is ON or OFF ", AL = [inComp('A1', 'Act', ['', '', '', 'the output monitor status', '', 'shall be initialised', [...]], "the output monitor status shall be initialised to [True] ")], Text = "– , , provided that the manage monitor interface mode is ON or OFF , the output monitor status shall be initialised to [True] , ."
I4:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', ['', Head, '','all citizens', '', 'have',[prep('', 'reference number')]], CompText)], AL, [[], ['','','','',''], Text)]
	Out	Head = if , CompText = "if all citizens have reference number ", AL = [inComp('A1', 'Act', ['', '', '', 'the output regulator status', '', 'shall be set', [...]], "the output regulator status shall be set to [True] ")], Text = "– ,, if all citizens have reference number , the output regulator status shall be set to [True] ,."
I5:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', ['', '', '','It', '', V1,[P1]], CompText)], [[], ['','','','',''], Text)]
	Out	CL = [inComp('C1', 'Cond', ['', if , '', 'the monitor status', '', is , [...]], "if the monitor status is [True]")], V1 = shall be set , P1 = prep(to , '[True]') , CompText = "It shall be set to [True] ", Text = "– , , if the monitor status is [True] , It shall be set to [True] , ."
I6:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', ['', '', '',E1, 'shall be', V1,[P1]], CompText)], [[], ['','','','',''], Text)]
	Out	CL = [inComp('C1', 'Cond', ['', if , '', 'the regulator mode', '', exceeds , [...]], "if the regulator mode exceeds the output regulator status")], E1 = 'the regulator mode', V1 = set , P1 = prep(to , '[True]') , CompText = "the regulator mode shall be set to [True] ", Text = "– , , if the regulator mode exceeds the output regulator status , the regulator mode shall be set to [True] , ."
I7:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', ['', '', '',E1, 'shall', 'be',[prep('', 'easy')]], CompText)], [[], ['','','','',''], Text)]

	Out	CL = [inComp('C1', 'Cond', ['', 'provided that', '', 'the regulator mode', '', equals, [...]], "provided that the regulator mode equals [True]")], E1 = 'the output monitor status', CompText = "the output monitor status shall be easy", Text = "– , , provided that the regulator mode equals [True] , the output monitor status shall be easy , ."
I8:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', [", Head, ", 'The previous Status', ", V1, [P1]], CompText)], AL, [[], [",",",",",], Text)]
	Out	Head = 'provided that', V1 = equals, P1 = prep('', '[True]'), CompText = "provided that The previous Status equals [True]", AL = [inComp('A1', 'Act', ['', '', '', 'the manage monitor interface mode', '', shall be set, [...]], "the manage monitor interface mode shall be set to [True] ")], Text = "– , , provided that The previous Status equals [True] , the manage monitor interface mode shall be set to [True] , ."
I9:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', [", ", ",E1, 'Could', V1,[P1]], CompText)], [[], [",",",",",], Text)]
	Out	CL = [inComp('C1', 'Cond', ['', if, '', 'the regulator init timeout', '', is, [...]], "if the regulator init timeout is [True]")], E1 = 'the monitor status', V1 = set, P1 = prep(to, '[True]'), CompText = "the monitor status could be set to [True] ", Text = "– , , if the regulator init timeout is [True] , the monitor status could be set to [True] , ."
I10:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', [", ", ",E1, ", V1,[prep(P1, 'the access')]], CompText)], [[], [",",",",",], Text)]
	Out	CL = [inComp('C1', 'Cond', ['', 'provided that', '', 'the regulator mode', '', exceeds, [...]], "provided that the regulator mode exceed the manage monitor interface mode")], E1 = 'the monitor init timeout', V1 = equals, P1 = '', CompText = "the monitor init timeout equals access", Text = "– , , provided that the regulator mode exceeds the manage monitor interface mode , the monitor init timeout equals the access , ."
I11:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', [", Head, ", 'ACT', ", V1, [P1]], CompText)], AL, [[], [",",",",",], Text)]
	Out	Head = 'provided that', V1 = equals, P1 = prep('', '[True]'), CompText = "provided that ACT equals [True]", AL = [inComp('A1', 'Act', ['', '', '', 'the regulator status', '', 'shall be initialised', [...]], "the regulator status shall be initialised to [True] ")], Text = "– , , provided that ACT equals [True] , the regulator status shall be initialised to [True] , ."
I12:	In	[inReq('R1', [[], [], CL, [inComp('A1', 'Act', [", ", ",E1, ", V1,[P1, prep(", 'as far as possible')]], CompText)], [[], [",",",",",], Text)]
	Out	CL = [inComp('C1', 'Cond', ['', if, '', 'the signal', '', exceeds, [...]], "if the signal exceeds the alarm control ")], E1 = 'the regulator init timeout', V1 = shall be set, P1 = prep(to, '[True]'), CompText = "the regulator init timeout set to [True] as far as possible", Text = "– , , if the signal exceeds the alarm control , the regulator init timeout shall be set to [True] as far as possible , ."
I13:	In	[inReq('R1', [[], [], [inComp('C1', 'Cond', [", Head, ", E1, 'does not', V1, [P1]], CompText)], AL, [[], [",",",",",], Text)]
	Out	Head = if, E1 = 'the monitor status', V1 = equal, P1 = prep('', '[True]'), CompText = "if the monitor status does not equal [True]", AL = [inComp('A1', 'Act', ['', '', '', 'the signal', '', 'shall be initialised', [...]], "the signal shall be initialised to [True] ")], Text = "– , , if the monitor status does not equal [True] , the signal shall be initialised to [True] , ."

Table.2 shows how the proposed format is customised/controlled to serve in generating requirement sentence with quality issue for each of the identified indicators in Table.3. The table presents the indicator code, an example of controlled input format, and the corresponding obtained output by our generator (i.e., completing the missing parameters through generating the textual requirement under the controlled format).

Table 2: Example Input format of Single-Level Indicators and the Corresponding generated Output

ID	Example of input format and Generator Output	
$I'1$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], [",", ", ", ", "], Req1_Text), inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E3, V3, Rel3, 'no', 1)], [",",",",","] , Req2_Text)]
	Out	E1 = 'the defined position', V1 = V2, V2 = V3, V3 = '[True]', Rel1 = Rel3, Rel3 = equals, E2 = 'the sailing request', Req1_Text = "– , , if the defined position equals [True] , the sailing request equals [True] , .", E3 = 'the sailing termination', Req2_Text = "– , , if the defined position equals [True] , the sailing termination equals [True] , ."
$I'2$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('cond', E4, V4, Rel1, 'no', 2), inComp('act', E2, V2, Rel1, 'no', 1)], [",", '(1 or 2)', ", "] , Req1_Text), inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E3, V3, Rel3, 'no', 1)], [",",",","] , Req2_Text)]
	Out	E1 = 'the IDC inhibitor', V1 = '[False]', Rel1 = Rel3, Rel3 = equals, E4 = 'the brake pedal', V4 = V2, V2 = V3, V3 = '[True]', E2 = 'the sailing inhibitor', Req1_Text = "– , , if the IDC inhibitor equals [False] , or if the brake pedal equals [True] , the sailing inhibitor equals [True] , .", E3 = 'the transmission error', Req2_Text = "– , , if the IDC inhibitor equals [False] , the transmission error equals [True] , ."
$I'3$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('cond', E4, V4, Rel1, 'no', 2), inComp('act', E2, V2, Rel1, 'no', 1)], [",", '(1 or 2)', ", "] , Req1_Text), inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('cond', E4, V4, Rel1, 'no', 2), inComp('act', E3, V3, Rel3, 'no', 1)], [",",'(1 and 2)',","] , Req2_Text)]
	Out	E1 = 'the defined position', V1 = '[False]', Rel1 = Rel3, Rel3 = equals, E4 = 'the vehicle speed setpoint', V4 = V2, V2 = V3, V3 = '[True]', E2 = 'the sailing inhibitor', Req1_Text = "– , , if the defined position equals [False] , or if the vehicle speed setpoint equals [True] , the sailing inhibitor equals [True] , .", E3 = 'the sailing termination', Req2_Text = "– , , if the defined position equals [False] , and if the vehicle speed setpoint equals [True] , the sailing termination equals [True] , ."
$I'4$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], [",", ", ", ", "] , Req1_Text), inReq([inComp('cond', E2, V2, Rel1, 'no', 1), inComp('act', E1, V1, Rel1, 'no', 1)], [",",",","] , Req2_Text)]
	Out	E1 = 'the transmission error', V1 = V2, V2 = '[True]', Rel1 = equals, E2 = 'the LSC', Req1_Text = "– , , if the transmission error equals [True] , the LSC equals [True] , .", Req2_Text = "– , , if the LSC equals [True] , the transmission error equals [True] , ."
$I'5$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], [",", ", ", ", "] , Req1_Text), inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], [",",",","] , Req2_Text)]
	Out	E1 = 'the defined position', V1 = V2, V2 = '[True]', Rel1 = equals, E2 = 'the LSC', Req1_Text = "– , , if the defined position equals [True] , the LSC equals [True] , .", Req2_Text = "– , , if the defined position equals [True] , the LSC equals [True] , . "
$I'6$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('cond', E4, V4, Rel1, 'no', 2), inComp('act', E2, V2, Rel1, 'no', 1)], [",", '(1 or 2)', ", "] , Req1_Text), inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], [",",",","] , Req2_Text)]

	Out	E1 = 'the vehicle speed setpoint', V1 = '[False]', Rel1 = equals, E4 = E2, E2 = 'the sailing request', V4 = V2, V2 = '[True]', Req1_Text = "– , , if the vehicle speed setpoint equals [False] , or if the sailing request equals [True] , the sailing request equals [True] , .", Req2_Text = "– , , if the vehicle speed setpoint equals [False] , the sailing request equals [True] , ."
$I'7$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, '', Rel2, 'no', 1)], ['','', '', '', ''] , Req_Text)]
	Out	E1 = E2, E2 = 'the brake fault state', V1 = '[True]', Rel1 = equals, Rel2 = shall be set, Req_Text = "– , , if the brake fault state equals [True] , the brake fault state shall be set , ."
$I'8$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], ['','', '', '', ''] , Req1_Text), inReq([inComp('act', E1, V1, Rel1, 'no', 1)], ['','','','',''] , Req2_Text), inReq([inComp('cond', E3, V3, Rel3, 'no', 1), inComp('act', E4, V4, Rel4, 'no', 1)], ['','','','',''] , Req3_Text)]
	Out	E1 = 'the NoSailing requests', V1 = V2, V2 = V3, V3 = V4, V4 = '[True]', Rel1 = Rel3, Rel3 = Rel4, Rel4 = equals, E2 = 'the standstill request flag', Req1_Text = "– , , if the NoSailing requests equals [True] , the standstill request flag equals [True] , .", Req2_Text = "– , , , the NoSailing requests equals [True] , .", E3 = 'the sailing inhibitor', E4 = 'the vehicle speed quality factor', Req3_Text = "– , , if the sailing inhibitor equals [True] , the vehicle speed quality factor equals [True] , ."
$I'9$:	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], ['','', '', '', ''] , Req1_Text), inReq([inComp('act', E1, V1, Rel1, 'no', 1)], ['','','','',''] , Req2_Text)]
	Out	E1 = 'the engine error', V1 = V2, V2 = '[True]', Rel1 = equals, E2 = 'the brake fault state', Req1_Text = "– , , if the engine error equals [True] , the brake fault state equals [True] , .", Req2_Text = "– , , , the engine error equals [True] , ."
$I'10$	In	[inReq([inComp('cond', E1, V1, Rel1, 'no', 1), inComp('act', E2, V2, Rel1, 'no', 1)], ['','', '', '', ''] , Req1_Text), inReq([inComp('cond', E2, V2, Rel1, 'no', 1), inComp('act', E1, V1, Rel1, 'no', 1)], ['','','','',''] , Req2_Text), inReq([inComp('cond', E3, V3, Rel2, 'no', 1), inComp('act', E1, V1, Rel1, 'no', 1)], ['','','','',''] , Req3_Text)]
	Out	E1 = 'the brake pedal', V1 = V2, V2 = V3, V3 = '[True]', Rel1 = Rel2, Rel2 = equals, E2 = 'the sailing inhibitor', Req1_Text = "– , , if the brake pedal equals [True] , the sailing inhibitor equals [True] , .", Req2_Text = "– , , if the sailing inhibitor equals [True] , the brake pedal equals [True] , .", E3 = 'the LSC', Req3_Text = "– , , if the LSC equals [True] , the brake pedal equals [True] , ."

Table.3 lists the single-level indicators each mapped to the relating metric(s).

Table 3: Single-Level Indicators

Indicators	**Metrics**	**Indicators**	**Metrics**
I1: subject coordination	M1	I7 object as:- - adjective: clear, easy, strong, good, bad, efficient, useful, significant, adequate, fast, recent, near, far, close, in front,.etc - subjective language: user friendly, easy to use, cost effective - Loopholes: As far as possible - Superlatives - Comparatives	M4, M6, M15
I2: Predicate Coordination	M1, M8		
I3: object coordination	M1, M8, M15		
I4: ambiguous determinant	M2		
I5: pronoun or demonstrative pronoun	M3, M4, M7		
I6: passive voice	M5		
I8: reveal words previous, next, last..	M7		
I9: must, can, could, may, .etc	M4, M9, M10	I12:- - suffix: possibly, eventually, if case, if possible, if appropriate, if needed - loophole: As far as possible	M13, M15
I10: revealing nouns in the role of subject or object	M11		
I11: Acronym	M12		
I13: negated clause	M14		

Table.4 lists the entire multi-level indicators along side their quality metrics described in details.

Table 4: Multi-level Indicators

Issue	**Metric**	**Indicators**	
Inconsistency	M1: mutually inclusive/ exclusive Preconditions	$I'1$: equivalent preconditions	such case happens when two requirements have equivalent preconditions with conflicting actions.
		$I'2$: fully-contained preconditions	such case happens when two requirements have fully contained relation with conflicting actions
		$I'3$: partially contained preconditions	such case happens when two requirements share some satisfaction cases contained relation with conflicting actions.
	M2: Deadlock	$I'4$: happens when two events are waiting each-other to occur	
	M3: Redundancy	$I'5$: equivalent requirements	the popular know type of redundancy in literature is the fully equivalent requirements –whose preconditions and actions are equivalent.
		$I'6$: Fully contained Requirements	this metric is proposed by us where two requirements have the same action(s) and differences in the preconditions. However, all satisfied cases of the preconditions of one of them is subset of the satisfaction cases of the other.
Incompleteness	M4: Missed Attributes	$I'7$: occur when a value to compare/assign for a system entity is missed	
	M5: Unsatisfiable rule	$I'8$: is the case when action(s) of a specific requirement will never occurrence its preconditions are not satisfied	
	M6: Unchecked - declared value	$I'9$: is the case when an entity is assigned a value that never used within the system requirements	
Correctness	M7: infinite loops	$I'10$: is the case when the system went into infinite loop	