Master of Software Engineering Capstone Paper Collection, 2025

Bradley Schmerl, *Editor*.

CMU-S3D-25-123
December 2025

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

This report is a collection of papers authored in the 2025 Summer and Fall sections of the course 17-679 Thesis Writing for Industrial Software Research, taken by students in the distance program of the Masters of Software Engineering professional program.

2025 All Rights Reserved

#'@0$'3)$0-')@0$+.'M?)$,I%*4)@1,*4'3) #'@0$'3)$0-')(%//%@".&),3)6'..:)J'@,"3')'.'$'*&3)
,-')0/&'*)&%41&.C)@0"+.'(?),)@1,*4')&0)0*')$0(".')$,C)*'@'33%&,&')@1,*4'3)%*)0&1'-3)
R-4,*%X,&%0*3),.30)/0"*()(%//%@".&%'3)6%&1)@0*&%*"0"3)('+.0C$'*&:)Z0&)0*.C)(0'3)&1')'*&%-')
3C3&'$)*"()&0)#')('+.0C'()',@1)&%$'?)0/&'*),..)&'3&3)$"3&)#')-"*)6%&1)',@1)('+.0C$'*&)
-'2"%-%*4)%4*%/%)@%&$'),*()-'30"-@'3:))
)
E)3%$%.,-)%33"')%3)3"*)61'*)3@,.%*4),*),++.%@,&%0*:)H*)0-('-)&0)3@,.'),)$0*0.%&1%@),++.%@,&%0*?)
&1'-'),-')&60)0+&%0*3G),((%*4)$0-')-'30"-@'3)&0)&1')@"--'*&)('+.0C$'*&?)%:':)B'-&%@,.)3@,.%*4?)0-)
('+.0C%*4),)*'6)*0(')0/)&1')'*&%-'),++.%@,&%0*?)%:':)10-%X0*&,..C:)7B'*)%/)0,)3$),.)3@,.%*4:)H/),*)%*@-',3'().0,()%3)
+'-3%3&'*&?)B'-&%@,.)3@,.%*4)%3),++-0+-%,&':)[06'B'-?)%*)+-,@&%@')3C3&'$3),-')$0-').%I'.C)&0)1,B')
&0),@@0$$0(,&')(C*,$%@).0,(3:)91%3).',(3)&0)3C3&'$3)*"(%*4)&0)0B')-..0@,&')-'30"-@'3)%*)
0-('-)&0)'*3"-')&1'C)@,*)1,*(.')&%$'3)0/)+',I).0,(:)91%3)%3)B'-C)3%$%.,-)&0)&1')%33"')$0*0.%&13)
/,@')61'*)3@,.%*4)10-%X0*&,..C:)7B'*)%/)0,)3$),.)3@,.%*4)10-%X0*&,..C:)7B'*)%/)0,)3$),.)3"#3'&)0/)&1'),++.%@,&%0*?)%:'3"$%*4)&1')
$,\0-%&C)0/)&1')-'30"-@'3?)&1')',&%*&1')*"()&0)3@,.'?)&1'C)$"3&)3&%*&1')('+.0C'()6%&1)&1')
'*&%-'),++.%@,&%0*:)E4,%*?)$,I%*4)0+&%$%X,&%0*)0/)3@,.%*4)-'30"-@'3)@1,..'*4%*4:
)
H&)310".(),.30)#')*0&'()&1')(%//%@".&%'3)&1,&)$0*0.%&13)$,C)@-',&')0*),*)0-4,*%X,&%0*,.).'B'.),3)
6'..:)90)+,-,+1-,3')5'.B%*)80*6,CG)&1')3&-"@&"-')0/),)3C3&'$)$%--0-3)&1')0-4,*%X,&%0*L3)
@0$$"*%@,&%0*)3&-"@&"-')0/),)3C3&'$)$%--0-3)&1')0-4,*%X,&%0*L3)
%*3%41&)%*&0)61C)$0*0.%&13)+-0B')&-0"#.'30$')/0-)0-4,*%X,&%0*3),&)3@,.':)K%&1)&1')#0"*(,-%'3)
#'&6"*)3'-B%@'3)#'%*4).003'.C)('/%*'(?)&1')#0"*(,-%'3)#'&6"*)&',$3)@,*)#'@0$')3%$%.,-.C)
,$# %4"0"3:)R-4,*%X,&%0*3)$,C)1,B')(%//%@".&%'3)('/%*%*4)&',$)-'3+0*3%#%.%&%'3)0-)&1')06*'-31%+)
0/)30/&6,-')'.'$'*&3)("')&0)&1'),$#%4"%&C)%*1'-'*&)%*)&-,(%&%0*,.)$0*0.%&1%@),++.%@,&%0*3:)

2.3 Microservices Architecture

E3)0++03'()&0)&1')$0*0.%&1%@),-@1%&'@&"-')('3%4*?),)$%@-03'-B%@'),-@1%&'@&"-')'$+.0C3)$".&%+.')
'M'@"&,#.'3)61'-')',@1)3'-B%@')%3)('+.0C'()%*('+'*('*&.C:)E3)('3@-%#'()#C)P06.'-),*()N'6%3)
%*)&1'%-)3'$%*,.),-&%@.')0*)&1')&0+%@G

> *"[T]he microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery.[5]*

91%3)('3@-%+&%0*?)61%.')30$'61,&)#-0,(?)+-0B%('3)&1')#,3'.%*')0/)106)&1%3),-@1%&'@&"-')@,*)#')
%$+.'$'*&'(:)H*)4'*'-,.?)&1%3)&,I'3)&1')/0-$)0/),)3'%&')0/).003'.C)@0"+.'(?)%*('+'*('*&)3'-B%@'3)
61%@1)"&%.%X')'M&'-*,.)@0$$"*%@,&%0*3)B%,)&1')*'&60-I:)91%3)%3)B'-C)3%$%.,-)&0)0/)61,&)@0*3&%&"&'3),)
$%@-03'-B%@')@,*)#')(%//%@".&)&0),-&%@".,&'?),3)&1'-')%3)*0&),)2",*&%/%,#.')$'&-%@)&1,&)60".()%*@."(
0-)'M@."(')&1%3)@.,33%/%@,&%0*:)E)@.,-%/%@,&%0*)#C))D,".0)5'-30*)@,*)1'.+)&0),((-'33)&1%3)
,$#%4"%&C)0/)-),)$%@-03'-B%@'?)S&1')('+.0C$'*&)"*%&)310".()@0*&,%*)0*.C)0*')3'-B%@')0-)\"3&),)/'6)
@01'3%B')3'-B%@)91%3)@-',&'3)&0)+0&'*&%,.)$%@-03'-B%@')#0"*(,-%'3?)0*')&1,&)%*@."('3),)
3%*4.')3'-B%@'),*()0*')&1,&)$,C)+-0B%(',)3$,..)*"$#'-)0/)3'-B%@'3)%/)&1'C),-')%*'M&-%@,#.C)
@01'3%B':)H*)0&1'-)60-(3?),)$%@-03'-B%@')%3)&1')3$,..'3&)"*%&)+033%#.')&0)+'-/0-3),)B%@*)

90)'*/0-@')&1%3)('/%*%&%0)6%..)B%'6)&1'),#%.%&C)&0)/"-&1'-)('@0$+03'),)('+.0C,#.')"*%&),3)
'B%('*@'),4,%*3&)%&3)@.,33%/%@,&%0*),3),)$%@)0-03'-B%@':

,/:/0%1';$28"$<';"8%6"3"7'48%

5%@-03'-B%@')('3%4*3),((-'33)$,*C)0/)&1')&-,('0//3)0/)$0*0.%&13G)%*('+'*('*&.C)3@,.,#.'?)
3&-0*4)3'-B%@'),*()&',$)#0"*(,-%'3 ?)/,".&)%30.,&%0*?),*()3%$+.%/%'()@0*&%*"0"3)('+.0C$'*&),*()
@0*&%*"0"3)%*&'4-,&%0*)]8H^8=_:)91')$03&)0#B%0"3)#'*'/%&3)@0$')/-0$)&1')%*('+'*('*@')0/)',@1)
3'-B%@':)K%&1)',@1)3'-B%@')#'%*4)3'+,-,&')/-0$)&1')0&1'-3?)&1'C)@,*)#')('3%4*'()6%&10"&)
@0*3%('-,&%0*)0&1'-)3'-B%@'3)#'C0*()61'-')&1'C)%*&'-,@&:)91%3)$',*3)/-"(0$)&0)"3')(%//'-'*&)
&'@1*%2"'3),*()&'@1*0.04%'3)&1,&)$,C)*0&)#')+033%#.')%/)&1')3'-B%@'3)6'-')1%41.C)@0"+.'(:)
9',$3),.30)1,B')4-',&'-)/.'M%#%.%&C)&0)('B'.0+),*()'B0.B')3'-B%@'3)%*('+'*('*&.C:)='+. 0C$'*&)
,.30)#'@0$'3) 3%$+.'-),3)0*.C)*'@'33,-C)3'-B%@'3)$"3&)#')&'3&'(),*()('+.0C'(:)
)
E)$,\0-)#'*'/%&?),*()0*')0/)&1')$03&)&0"&'(?)%3)%*)&'-$3)0/)3@,.,#%.%&C:)K%&1),)$%@-03'-B%@')
,-@1%&'@&"-'?)%&)%3)+033%#.')&0)('+.0C)0*.C)&1')3'-B%@'3)&1,&),-')%*)*"()0/),((%&%0*,.)-'30"-@'3:)
91%3)$,I'3)10-%X0*&,.),*()(C*,$%@)3@,.%*4)3%4*%/%@,*&.C)*')3(%%&%0*?)$,%*&,%*%*4)3'-B%@'3)
61'*)/,".&3)0@@"-)#'@0$'3)$0-')+,%*.'33),3)/,".&3)@,*)#')%30.,&'()6%&1%*)&1')3'-B%@')61'-')
&1'C)0@@"--'()
)
P%*,..C?)$%@-03'-B%@'3)$%--0-),)$"@1)$0-')'//%@%'*&)&',$)3&-"@&"-'?),4,%*)-'/'-'*@%*4)80*6,CL3)
N,6:)K%&1)@.',-)-'3+0*3%#%.%&%'3)#'&6"*)&',$3?)',@1)06*%*4)&1'%-)-'3+'@&%B')3'-B%@'?)&',$3)
@,*)4-06),*()('B'.0+)%*('+'*('*&.C)/-0$)',@1)0&1'-:)91%3)%3)+,-&%@".,.C)*'@'33,-C),3)
0-4,*%X,&%0*3)4-06),*()@.',-)#0"*(,-%'3)#'@0$')%*@-',3%*4.C)*'@'33,-C):

,/:/,%1';$28"$<';"%9$*!"2778%

K1%.')&1')#'*'/%&3)3"$)@.',-)@0$+,-'()&0),)$0*0.%&1%@)('3%4*?)$%@-03'-B%@'3)-'2"%-'),).'B'.)0/)
0B'-1',()&1,&)%3)*0&),.6,C3)6,--,*&'(:)K%&1)',@1)3'-B%@')*"'(%*4)&0)#')('+.0C'()
%*('+'*('*&.C?),*),((%&%0*,.).'B'.)0/)%*%&%,.)-'30"-@'3)%3)"'()/0-)&1')3,$'),++.%@,&%0*)61'*)
@0$+,-'()&0),)$0*0.%&1:)!%$%.,-.C?),*),((')0-4,*%X,&%0*,.)@0$+.'M%&C)%3)-'2"%-'()&0)$,%*&,%*)
&1')&',$),*()0-4,*%X,&%0*,.)#0"*(,-%'3)$%@-03'-B%@'3)%*@"-:)P0-)3$,..'-)0-4,*%X,&%0*3?)&1'3')
#0"*(,-%'3)$,C)+-0B')#"-('*30$' :)91%3),.30),++.%'3)&0)+,-&%@".,-4')0-4,*%X,&%0*3)3) 6'..:)
P0-)A#'-?)&1'C)%$+.'$'*&'()=0$,%*)R-%'*&'()5%@-03'-B%@'3)E-@1%&"-')=R5E_)&0)$,%*&,%*)
&1'%-)$%@-03'-B%@'3),-@1%&'@&"-'?)#'3&-,@&)&1')$1,'*C)%*(%*B%(",.)3'-B%@')#0"*(,-%'3)%*&,0-4'-)
(0$,%*)#0"*(,-%'3 !"&#.))
)
5,%*&,%*%*4)(,&,)#0"*(,-%'3),.30)#'@0$'3)$0-')@"$#'-30$':)P0-)&C+%@,.)$%@-03'-B%@')
('3%4*3?)',@1)3'-B%@')06*3)%&3)06*)(,&,?)61%@1)@,*)@-',&')%*@0*3%3&'*@%'3)#'&6"*)(,&,)3&0-'3)
0-)@0$+.%@,&')%3)B%@')&-,*3,@&%0*3:)!%$%.,-)&0)&1'),((')0-4,*%X,&%0*3)/,/%(@".&%'3?)
,@@0"*&%*4)/0-),((%&%0*,.)(,&,)@0$+.'M%&C)-')0B%-@0$+.%@,&%0*3)@'-&,%*),++.%@,&%0*3
91'3')(,&,)#0"*(,-%'3) @,*),.30)@-',&'),((')(@1,..'*4'3)/0-).,-4')3C3&'$3),3)6'..?)61'-')(,&,)
#'@0$'3)1%41.C)(%&3%#"&'(:)91%3)@,*"@'33%&,&')%*&'-,@&)%6%&1)$".&%+.')3'-B%@'3)&0)0#&,%*)
@'-&,%*)%*/0-$,&%0*)N*:4' C3@,.')3C3&'$3)3" @1),3)Z'&/.%M%@1,'()&1%)(%//%@"#&C)%$+.'$'*&%*4)
['M,40*,.),-@1%&'@&"-?)61%@1)3&-,@&)&1')3C3&'$3)3.04%@)/-0$)%&3,.,&,

2.4 Hybrid Architectures

E)1C#-%()30/&6,-'),-@1%&'@&"-')%3),)@0$#%*,&%0*)0/)@1,-,@&'-%3&%@3)/-0$)$".&%+.'),-@1%&'@&"-,.)
('3%4*3:)%:':)%&)1,3)'@&3)0/),)$0*0.%&1%@)('3%4*),*(),)$%@-03'-B%@')('3%4*:)E3),)$0*0.%&1)%3),)
3C3&'$),3),)3%*4.')"*%&?)%&)('/%*%&%0*,..C)(0'3)*0&)@0*/0-$)&0),)(%3&%@)3&%@)0*)'*&()('3%4*)
3"@1),3)$%@-03'-B%@'3:)J'@,"3')0/)&1%3?)6')/0@"3'()$0-')0*)&1')@1,-,@&'-%3&%@3)0/)'%&1'-)('3%4*)
-,&1'-)&1,*),)3&-%@&)('/%*%&%0*:)P0-)'M,$+.'?)0*')0/)&1')@1,-,@&'-%3&%@3)0/)$%@-03'-B%@'3)%3)&1')
3&-0*4)&',$),*()3'-B%@')#0"*(,-%'3?)@0"+.'()6%&1)&1')3+,-,&%0*)0/)(0$,%*3),-0"*()#"3%*'33)
@,+,#%.%&%'3:)K1'*),++.%'()&0),)$0*0.%&1?)&1%3)@,*),%()%*)&1')('B'.0+$'*&)0/),)$0(".,-)$0*0.%&1)
('3@-%#'(),#0B' :)
)
H*)&'-$3)0/)3&-"@&"-'?)&1'-')%3),)#-0,()3+'@&-"$)0/)106)&1'3')&60),-@1%&'@&"-'3)@0".()#')
@0$#%*'(:)E)$0*0.%&1?)61%@1)'*@0$+,33'3)&10-%&0)0/),)3C3&'$3?)$,C)('+.0C),)3$,..)
*"$#'-)0/)%*(%B%(",.)$%@-03'-B%@'3)/0-)B,-%0"3)-',30*3?)3"@1),3)3@,.%*4:)P0-),)$%@-03'-B%@'3)
('3%4*'()3C3&'$?)&1')*"()/0-)31,-'()(,&,?)/0-)%*3&,*@'?)$,C)'*@0"-,4')&1')4-0"+%*4)0/)
$".&%+.')3'-B%@'3)%*&0),)3%*4.')"*%&:)H*)4'*'-,.?)61%&)M+'@&'()%*)&1%3)+,+'-)%*)&'-$3)0/),)1C#-%()
,-@1%&'@&"-')%3),)@0$#%*,&%0*)0/)&1'3')3&-"@&"-,.)'.'$'*&3),3)6'..),3)('3@-%+&%B')@1,-,@&'-%3&%@3)
3"@1),3)#0"*(,-%'3),*()+-0@'33'3:))

2.4 Defining Architectures

90)3"$$,-%X')&1'),#0B'?),*()+-0B%(').)3%*4".,-)+0%*&)0/)-'/'-'*@')$0B%*4)/0-6,-(?)#'.06),-')
&1')('/%*%&%0*3)&1,&)6%..)#')"3'()%*)&1%3)+,+'-)/0-)&1'),-@1%&'@&"-,.)('3%4*3)(%3@"33'(G

- W) "#$#%&'(&)!*+)(&',)'-+,.),*),++.%@,&%0*),3),)3%*4.')('+.0C,#.')"*%&)61%@1)
  '*@0$+,33'3),..)%*&'-*,.)3'-B%@'3)0/),)4%B'*)3C3&'$)
- W) "&)+#/,+0&),/!*+)(&',)'-+,.),*),++.%@,&%0*),3),)3"%&')0/)3%*4.')('+.0C,#.')"*%&3?)',@1)
  #'%*4)&1')3$,..'3&)+033%#.')"*%&)6%&10"&)&1'),#%.%&C)/0-)/"-&1'-)('@0$+03%&%0*:
- W) 123+&4!*+)(&',)'-+,.! ,*),++.%@,&%0*),3),)3"%&')0/)(%3&-%#"&'()('+.0C,#.')"*%&3?)61'-')
  30$')"*%&3)'*@0$+,33)$".&%+.')3'-B%@'3)&1,&)@0".()-',30*,#.C)#')/"-&1'-)('@0$+03'(:!

91')(%3&%*@&%0*)0/)106)0-)61C)&1'3')3'-B%@'3)$,C)#')3&-"@&"-'(?)':4:),-0"*()#"3%*'33)
@,+,#%.%&%'3?)%3)%*&'*&%0*,..C),#3'*&)/-0$)&1'3')('/%*%&%0*3:)91%3),#3'*@')%3)("')&0)&1')%*,#%.%&C
6%&1%*)&1%3)-'3',-@1)&0)('&-$%*')&1'3')#0"*(,-%'3?)-,&1'-)&1,*)&1')#)&1,&)&1'3'),-')0+&%0*,.)
@1,-,@&'-%3&%)E0*31'-)+0+".,-),-@1%&'@&"-,.)('3%4*%'/.)$'*&%0*'() %*)&1%3)+,+'-)%3),)
3'-B'-.'33),-@1%&'@&"-K1%.')&1'3'),++.%@,&%0*3)3&%..)-"*)0*)3'-B'-3?)&1'3'-,@&),6,C)&1')
"*('-.C%*4),-@1%&'@&"-)!'-B' -.'33)('3%4*3),-')$03&)0/&'*)3"*),3) P"*@&%0*3) ,)!'-B%@')
]P,,!_)61%@1),-')&-%44'-)0-B'*& G,3'( )3'-B%@')3:

93.:(,;%)%'%*<.

90)0#&,%3)0"-@'3?)6')/0..06'( )F%&@1'*1,$L3)4"%('.%*'3)/0-)+'-/0-$%*4)3C3&'$,&%@).%&'-,&"-')
-'B%'63)%*)30/&6,-')'*4%*"'-%*4!!".#)J'@,"3')6'),-')%*&'-'3&'()%*)&1')30/&6,-') ,-@1%&'@&"-)03)
3C3&'$3)@"--'*&.0('+.0C'(?)6')"&%.%(,)&,%.0-'()B'-3%0*)0/)&1')F%&@1'*1,$)4"%('.%*'3)&1,&)

=/,/0/0%1232('45>%?(*%@

='3+%&')#'%*4),).,-4' C3@,.')6'#),++.%@,&%0*?)!.,@I)%3)0*')0/)&1')/'6)0-4,*%X,&%0*3)3&%..)
%$+.'$'*&%*4),)$0*0.%&1%@)('3%4H*)&1'%-),-&%@'C)(&,%.'() &1'%),++-0,@1)&0)0*')0/)&1')
3%4*%/%@,*&).%$%&,&%0*3)0/)&1')$0*0.%&1G)@0*&%*"0"3)('+.0C$'*&3:)H*)&1'%-),-&%@.'?)!.,@I)('3@-%
"3%*4)Sj'.',3'J0&?V),)3'-B%@')&1'C)@-',&'()&1,&)$0*%&0-3)&1')3C3&'$),*()('&'@&3),*0$,.%'3:)
91%3)3C3&'$)%3)"3'()&0)$0*%&0-)('+.0C$'*&3),*()@,&%0*3)&1')$0-')2"%@I.C)&1,*)&1')-'.',3')
&',$)1,() +-'B%0"3.G)*),#.')&0:)

=/,/0/,%12!A(*$%1232('45>%?52B/7C

H*),*),-&%@.')+"#.%31'()P'#-",-C)bcdk?)!10+%/C)('&,%.3)&1'%-)\0"-*'C)&1-0"41)&1').%$%&,&%0*3)
,-%3%*4)/-0$)&1'%-)$0*0.%&1%@)(.)B1)+,-&%@".,-)%33"'3)&1'),-&%@.')3"I3)&0),((-'33),-')
&103')%*1'-'*&)%*),)$0*0.%&1L3).,@I)0/)@.',-)#0"*(,-%'3:)E3)6%&1)$,*C)$0*0.%&13),3)&1'C)4-06?)
&1%3)-'3".&'()%*)1%41)@0"+.%*4)#'&6"*)@0(')&1,&)1,*(.'()(%//'-'*&)#"3%*'33)+-0@'33*)K1%.')
&1'),"&10-),($%&3),)$%4-,&%0*)&06,-(3)$%@-03'-B%@'3)60".()30.B')&1')%33"'3)!10+%/C)6,3).00I%*4)
&0),((-'33?)&1'C)+-0B%(',)*"$#'-)0/)-',30*3)61C)&1%3)6,3)*0&)&1')@0--'@&)30."&%0*)/0-)&1'%-)
3C3&'$:)P%-3&)#'%*4)&1'),(('()@0$+.'M%&C)%*1'-'*&)%*)$%&)B%@'3:)H/)%$+.'$'*&'(?)!10+%/C)
60".()&1'*)*"'()&0)$,%*&,%*)$".&%+.')(%//'-'*&)&'3&),*()('+.0C$'*&)+%+'.%*'3),3)6'..),3)&1')
,((%&%0*,.)%*/-,3&-"@&"-,.)0B'-1',()0/)',@1)3'-B%@':)91'C)6'-'),.30)@0*@'-*'()/0-)&1')%*@-',3'()
.,&'*@C),*()('@-',3'()-'.%,#%.%&C)1&)60".()#')*"'(')("')&0)3'-B%@'3)*06)@0$$"*%@,&%*4)
,@-033)&1')*'&60-I:)N,3&.C?)&1'C),@I*06.'(4')&1'),(('()(%//%@".&%'3)%*)+'-/0-$%*4).,-4')-'/,@&0-3)
,@-033)$".&%+.')3'-B%@'3:)91"3?)&1')-'3".&)6,3),)$%4-,&%0*)&06,-(3),)$0(".,-)$0*0.%&1)%*3&',(?)
61%@1)%$+()&1')#0"*(,-%'3)*"'(')6%&10"&)&1'),(('()@0$+.'M%&C)&1'C)30"41&)&0),B0%()

=/,/0/:%?"$<';"%D$"'34"!%1232('45>%E'4%A)

!%$%.,-)&0)!10+%/C?);%&1"#L3)3C3&'$)-"*3)0*),).,-4')j"#C)0*)j,%.3)$0*0.%&1:)J'C0*()61,&)1,3)
#"*)(%3@"33'(),#0B'?)61,&)%3)60-&1)*0&%*4)%*)-'4,-(3)&0);%&1"#L3),-@1%&'@&"-')%3)&1'),(0+&%0*)0/)
!'-B%@')R-%'*&'()('3%4*)+-%*@%+.'3)6%&1%*)&1'%-)$0*0.%&1%@)(.)B1)+,-&%@.')
('3@-%#%*4)&1%3)@1,*4')3&,&'3?),)S1C#-%()$0*0.%&1^3'-B%@'3),-@1%&'@&"-')%3)6%&1%*)
(%3@"33'()$0-').,&'-:)91,&)#'%*4)3,%(?);%&1"#L3),++-0,@1)&0),)$0*0.%&1%@),-@1%&'@&"-')%3)"*%2"')
,3)%&),++.%'3,)-B%@'3)0-%'*&'(),++-0,@1)&0),)$0*0.%&1%@),-@1%&'@&"-':)91')%33"'3);%&1"#),%$'()
&0),((-'33)6%&1)&1%3),++-0,@1)1,().'33)&0)(0)6%&1)&1'),++.%@,&%0*)%&3'./),*()$0-')&0)(0)6%&1)&1')
0-4,*%X,&%0*:)E3)&1'),-&%@.')3&,&'3?)&1')+-'B%0"3)0-4,*%X,&%0*,.)3&-"@&"-')0/)"3%*4)@%()06*'-3)
*0&)S2"%&')'M+-'33)106)6')60-I?V)$,I%*4)&1')*"'()/0-)3'-B%@')#0"*(,-%'3)$0-'),++,-'*&:)

=/,/,%1232('45%42%1';$28"$<';%8

e:b:b:d)50*0.%&1)&0)5%@-03'-B%@'3G)A#'-),*()=00-=,31
J0&1)A#'-),*()=00-=,31)6'-'),$0*4)&1')$,*C)0-4,*%X,&%0*3)&0)&-,*3%&%0*)&1'%-)30/&6,-')
,-@1%&'@&"-')/-0$)&1'%-)0-%4%*,.)$0*0.%&1)&0)$%@)A3'-B%@'3)*%*4)6,3)S)&0)/0..06)&1')
.',()0/)0&1'-)1C+'- C4-06&1)@0$+,*%'3?V)%$+.C%*4)&1')&-,*3%&%0*)6,3)%*%&%,&'()("')&0),)('3%-')/0-)
4-06&1)-,&1'-)&1,*),)*'@'33%&C)("')&0)4-06&1:)91%3)%3)0++03'()&0)=00-=,31L3)&-,*3%&%0*)61%@1)
6,3)*'@'33%&,&'()#C)&1'%-)4-06&1:)K1%.')=00-=,31)6,3)'M+'-%'*@%*4)%33"'3)3&'$$%*4)/-0$)
&1'%-)$0*0.%&1%@)(.)B1)+-0@'%&'@&"-'),.-',(C?)&1')8RlH=0k)+,*('$%@)@0$+0"*('()&1'3')%33"'3)6%&1),)
3%4*%/%@,*&)%*@-',3')%*)&1')B0."$')0/)0-('-3)&1'C)-'@'%B'(:)P0-)#0&1)0-4,*%X,&%0*3?)&1'),#%.%&C)&0)

-',@1)&1')('3%-'()B0."$')0/)@"3&0$'-3?),3)6'..),3),B0%()&1')0+'-,&%0*,.)1"-(.'3)/,@'()#C).,-4')
0-4,*%X,&%0*3?,()&1')&-,*3%&%0*)&0)$%@-03'-B%@'3)6,3)3"*),3)*'@'3),-C:
K1%.')#0&1)0-4,*%X,&%0*3),++',-)&0)1,B')#'*'/%&'()/-0$)&1%3)&-,*3%&%0*?)&1'C)1,B'),.30)#"*)
@-%&%@,.)0/)$%@-03'-B%@'3),3)6'..:)H*),)#.04)+03&)+"#.%31'()g)C',-3),/&'-)&1'%-)&-,*3%&%0*?)A#'-)
%*&-0("@'()61,&)&1'C)@,..),)S(0$,%0-%'*&'()$%@-03'-B%@'),-@1%&'@&"-'3:)H*),)&1%3),-&%@.'?)&1')
,"&10-)('3@-%#'3)%33"'3)&1')@0$+,*C)1,3)/,@'()6%&1)#0&1),-@1%&'@&"-'3:)P0-)%*3&,*@'?)+-'B%0"3.C)
S&1')$0*0.%&1%@),-@1%&'@&"-')&%'()&1')/,&')0/)&',$3)&04'&1'-),*()$,(')%&)(%//%@".&)&0)0+'-,&')
%*('+'*('*&.C:V !"&#![06'B'-?)*06)ST6U%&1),)$%@-03'-B%@'),-@1%&'@&"-')0*')&-,('3),)3%*4.')
$0*0.%&1%@)@0(')#,3')/0-)#.,@I)#0M'3)6103')/"*@&%0*,.%&C)@,*)@1,*4'),&),*C)&%$'),*()',3%.C)
@,"3')"*'M+'@&'()#'1,B%0-QV)'M+.,%*%*4G)S'*4%*"-3)T1,B'U)&0)60-I)&1-0"41),-0"*()gc)3'-B%@'3)
,@-033)db)(%//'-'*&)&',$3)%*)0-('-)&0)%*B'3&%4,&')&1')-00&)@,"3')0/)&1')+-0#.$:V !"&#

e:b:b:d)!+'@%,.)80*3%('-,&%0*G)Z'&)/.%M

H*))bcbc),-&%@.')+03&'()&0)&1'%-)Z'&/.%M9'@1J.04?)('&,%.%*4)&1')E*(-0%()&',$L3)'//0-&)&0)$%4-,&')
&1')EDH)#,@I'*()$0*0.%&1)%*&0),)*'6)$%@-03'-B%@':)91%3)#,@I'*()6,3)('3@-%#'(),3)S'33'*&%,..C)
T,U)3'-B'-.'33)$0('.)%*),)$0*0.%&1:V !"&#!K1%.')*0&)/"..C)('3@-%#'(3)%3),33"$'()&1,&)&1%3)
,++.%@,&%0*)$"3&)1,B')#"*)@0*&,%*'()6%&1%*),)3%*4.')3'-B'-.'33)/"*@&%0*:)733'*&%,..C)#'%*4),)
3'-B%@'),3),)3%*4.')('+.0C,#.')"*%&?)('+.0C'(),3),)3'-B'-.'33)/"*@&%0*)@,.:)91'),"&10-3)('&,%.'()
&1')+-0@'33),*()-'3".&3)0/)$%4-,&%*4)&1%3)S$0*0.%&1V)%*&0),)*'6)$%@-03'-B%@':)91%3)%*B0.B'()
$%4-,&%*4)mdnc)-0"&')1,*(.'-3)/0-)&1'%-)-'3+'@&%B')2"'-C)+,&13:)J,3'()0*)&1'),-&%@.'?)&1%3)V O
('3@-%#'()$0*0.%&1)6,3)$%4-,&'()%*&0),)3%*4.')$%@-03'-B%@'),/&'-),)C',-)0/)'//0-&:)

=/,/:%1';$28"$<';"8%42%1232('45>%FG*H23%I$'G"%J'!"2%*3!%K84'2%LE22#("+%K6%L+%*3!%MC74

91')+-'B%0"3)'M,$+.'3)0/),-@1%&'@&"-,.)&-,*3%&%0*3?)/-0$),)$0*0.%&1)&0)$%@-03'-B%@'3?)1,B')#"*)
6'..)(0@"$'*&'()&0)&1')+0%*&)61'-')#00I3)1,B')#"*)+"#.%31'()('&,%.%*4)#'3&)+-,@&%@'3:)K1,&)%3)
.'33)@0$$0*.C)6-%&&'*),#0"&)%3)&1')-'B'-3':)K1,&)#0&1)0/)&1'3')'M+.'3)1,B')%*)@0$$0*)%3)
&1'C)6'-')('3%4*'()&0)&,I'),(B,*&,4')0/)&1')#'*'/%&3)$%@-03'-B%@'3)+-0B%('?),*(',-.C)0*)/0"*()
&1'3')#'*'/%&3).,@I%*4)%*)+-,@&%@':)P0-)E$,X0*)D-%$')I%('0?)&1'%-),"(%0^B%('0)$0*)%*B0.B'()*4)
0-%4%*,)"&%.%X'()EK!)N,$#(,)/"*@&%0*3?),)/0-$)0/)3'-B'-.'33),-@1%&'@&"-'3:)6'..),3)
$%@-03'-B%@'3:)91'C)(%3@0B'-'()&1,&)61'*)3@,.%*4)&1%3),++.%@,&%0*?)#0&&.'*'@I3)'$'-4'(),*()
@03&3)#'4,*)&0)-%3':)J'@,"3')0/)&1%3?)&1')$%@-03'-B%@'3),/'(3&%0*3)6'-')@0*30.%(,&'()
%*&0),)3%*4.')3'-B')$0*0.%&1%@)3'-B%@':)E@@0-(%*4)&0)&1'),-&%@.'?)&1%3)-'("@'()@03&3)#C)kco
)
H3&%0)+-0B%('3),)B'-C)(%//'-'*&)'M,$+.'?)#"&)0*')60-&1)*0&%*4:)91%3),++.%@,&%0*?)61%@1)+-0B%('()S,)
"*%/0-$)6,C)&0)@0**'@&?)3'@"-'?)$,*,4'?),*()$0*%&0-)@.0'(),&%B'),++.%@,&%0*3)('+.0C'()%*)
F"#'-*'&'3V !"&#)6,3)#"%.&)6%&1)&1')%*&'*&%0*)0/)4,*%*4)&1')#'*'/%&3)0/),)$%@-03'-B%@'),-@1%&'@&"-':
[06'B'-?)ST,U3)H3&%0),(0+&%0*)%*@-',3'(),*()&1')&',$)40&)/"(#,@I)/-0$)@"3&0$'-3?)&1'C)
-',.%X'()&1,&)$,*C)0/)&1')#'*'/%&3)&C+%@,..C),330@%,&'()6%&1)$%@-03'-B%@'3?)&1,&)%3
-0..0"&?)%*('+'*('*&)3@,.'?),*()3'@"-%&C)%30.,&%0*?)(%()*0&),"#!H-C3&',()0/),)(%3&-%#"&'()
('+.0C$'*&)0/)H3&%0)3'-B%@'3?),3)#'/%&)&1')('3%4*)0/)&1'),++.%@,&%0*?)@"3&0$'-3)1,(),)3%*4.')&',$)
0-)'B'*),)3%*4.')%*(%B%(",.)$,*,4%*4)%&:)91%3)-'3".&'()%*)*0&)0*.C)@"3&0$'-3)-'@'%B%*4)&1')
%*@-',3'()0+'-,&%0*,.)@0$+.'M%&C)%*1'-%&)$%@-03'-B%@'3)6%&10"&)4,%*%*4)&1')('3%4*L3)#'*'/%&3:

4.3 Unverifiable Sources

E)(%//%@".&C)$'*&%0*'()',-.%'-)%*)&1%3)+,+'-)%3)&1').%$%&'()+"#.%@)I*06.'(4')0/),)3C3&'$L3)30/&6,-')
,-@1%&'@&"-':)5"@1)1,3)#"*)3,%()-'4,-(%*4)(%//'-'*&).,-4' C3@,.'),++.%@,&%0*3?)3"@1),3)&103')
$'*&%0*'(),#0B'?)#"&)&1')3+'@%/%@%&C)&1,&)60".()#')%(',.)/0-)-'3',-@1)3"@1),3)&1%3)%3)3+,-3':)91,&)
#'%*4)3,%(?)&1'-'),-'),)*"$#'-)0/)30"-@'3)&1,&)@.,%$)@0$+,*%'3?)3"@1),3)Z'&/.%M)D,CD,.?),*()
!0"*(8.0"(!"&%.%X')#0&1)$%@-03'-B%@'3),*()$0*0.%&1%@)3'-B%@'3:)A*/0-&"*,&'.C?)&1'3')30"-@'3)
,-')+'-30*,.)#.043)&1,&)(0)*0&)@%&')61'-')&1%3)%*/0-$,&%0*)6,3)0#&,%*'(?)$,I%*4)&1')B'-,@%&C)0/)
&1'%-)@.,%$3)3"3+'@&:)="')&0)&1')@0*3&-,%*&3)0/)&1%3)-'3',-@1?)B'-%/0-$,&%0*)6,3)0#&)
#"*)+033%#.'Q)&1'-'/0-'?)/"-&1'-)-'3',-@1)6%..)#')-'2"%-'(:)
)
K%&1)&1%3)@,B',&?)%&)%3)3&%..)60-&1)%*B'3&%4,&%*4)&1'3')@.,%$3:)910"41)Z'&/.%M)6,3)(%3@"33'(),
,)5'(%"$),-&%@.')&%&.'(?)[06)Z'&/.%M)E@@%('*&,..C)D-0B'()50*0.%&13)!@,.')J'&&'-)91,*)
5%@-03'-B%@'3?)40'3)/"-&1'-)&0)@.,%$)&1')@0$+,*C)"3'3)$".&%+.')$0*0.%&1%(')&1'%-)
$%@-03'-B%@'3:)91%3)%*@."('3)&1'%-)-'@0$$'*(,&%0*)'*4%*'3),*()#%..%*4)3'-B%@'3:)E@@0-(%*4)&0)
&1'),-&%@.'?)&1'3')3'-B%@'3)"&%.%X'),)$0*0.%&1%@),++-0,@1),3),)$',*3)&0)%$+-0B')+'-/0-$,*@':)
91')".&%$,&')@.,%$)0/)&1'),-&%@.'?)*4)$%@-03'-B%@'3),((-'33)0-4,*%X,&%0*)#0&&.'*'@I3)61%.')
$0*0.%&13),((-'33),++.%@,&%0*)#0&&.'*'@I3:)Z0)/"-&1'-)('&,%.3)-'4,-(%*4)&1'3'),..'4')$0*0.%&13)
6'-')+-0B%('(:)
)
E*0&1'-),-&%@.')/-0$)5'(%"$?)+"#.%31'()%*)bcbp?)61%@1)/0@"3'3)0*)#"%.(%*4)1C#-%(),-@1%&'@&"-'3?)
@.,%$3)#0&1)D,CD,C),*()!0"*(8.0"()&0)#')-',.).%/')'M,$+.'3:)J0&1)3C3&'$3),..'4'(.C)
%*&'4-,&'()$%@-03'-B%@'3)%*&0)&1'%-)$0*0.%&1%@),-@1%&'@&"-'3?)&1)&1%3)%*@.'(&0')%
&,-4'&'()3'-B%@'3)&1,&)60".()'M+.%@%&.C)#'*'/%&)/-0$)&1')3@,.,#%.%&C),*()/.'M%#%.%&C)+-0B%('()#C)
$%@-03'-B%@'3)91'-')%3)30$')'B%('*@'?),3)$'*&%0*'()+-'B%0"3.C?)0/)!0"*(8.0"(L3)1C#-%()
,-@1%&'@&"-'?)#"&)*0&)&0)&1')('4-")#'%*4)@.,%$'()%*)&1'),-&%@.':)D,CD,.?)0*)&1')0&1'-)1,*(?)
3"++03'(.C) @103')&1%3),++-0,@1)&0),((-'33)0-4,*%X,&%0*,.).%$%&,&%0*3?)+0&'*&%,..C),3),)$',*3)&0)
%*@-'$'*&,..C)$%4-,&')&06,-(3)$%@-03'-B%@'3),.&04'&)1'-:)
)
H&)310".()#')*0&'()&1,&)&1'3')@.,%$3)6'-')%*B'3&%4,&'()6%&10"&)-'3".&:)7B%('*@')-'4,-(%*4)
!0"*(8.0"()1,3),.-',(C)#"*)$'*&%0*'(?)#"&)%*B'3&%4,&%*4)D,CD,.)6,3).'33)/-"%&/".:)J0&1)
D,CD,.),*()!0"*(8.0"()103&)'*4%*"-%*4)#.043?)&10"41)D,CD,.L3)%3)103&'()&1-0"41)5'(%"$:)
Z0)%*/0-$,&%0*)6,3)/0"*()61'*)%*B'3&%4,&%*4)D,CD,.)61%.')!0"*(8.0"()1,3),-&%@.'3)('&,%.%*4)
&1'%-)&-,*3%&%0*)/-0$),)$0*0.%&1)&0)$%@-03'-B%@'3:)E&&'$+&3)&0)"&%.%X')81,&;D9)-'3".&'()%*)@.,%$3
0/)#0&1)@0$+,*%'3)"3%*4),)$%@-03'-B%@'),-@1%&'@&"-'3?)&1'-)%*B'3)30"-@'3)-'&"-*%*4)ece)
'--0-3:)

4.4 Explicit or Implied Hybrid Architectures

960)0/)&1')@,3')3&"(%'3)('&,%.'(),#0B'?)Z'&/.%M),*()E$,X0*)D-%$')I%('0?)+-0B%(')'B%('*@')0/),)
1C#-%()30/&6,-'),-@1%&'@&"-')&1,&)6%..)#')(%3@"33'()$0-')%*)&1')*'M&)3'@&%0*:)!%$%.,-.C?)&1')
$%4-,&%0*)#C)!0"*(8.0"(?)$'*&%0*'()+-'B%0"3.C?)%*(%@,&'3),&).',3&)+0&,-C)1C#-%()
,++-0,@1:)N,3&.C?)%*),*)%*&'-B%'6)%*)bcbc)#C)91')j'4%3&'-?)&1')ID)0/)!0/&6,-')7*4%*"'-%*4),&)
;%&1"#?)!1,)5,?)(%3@"33'3)&1')@0$+,*CL3)%*&'*&%0*)&0)$%4-,&')&06,-(3)$%@-0)91%3)
%*&'-B%'6)%3)"*%2")@0$+,-'()&0)0&1'-)$%4-,&%0*3)$'*&%0*'(?),3)5,)'M+.%@%&.C)3&,&'3)&1')%*&'*&%0*)

/0-),)1C#-%()'*B%-0*$'*&:)[06'B'-?)&1'),-&%@.'3)$'*&%0*'(),#0B')('&,%.%*4);%&["#L3)$0*0.%&1%@)
,-@1%&'@&"-')6'-')6-%&&'*),/&'-)&1%3)%*&'-B%'6?),*()(0)*0&)$'*&%0*)$%@-03'-B%@'3:)P"-&1'-)
%*B'3&%4,&%0*)%*&0);%&["#L3),++,-'*&)1C#-%(),-@1%&'@&"-')%3)-'@)0$$'*('(:

?3.@6-"5--6%$

K1%.')&1%3)-'3',-@1)6,3)@0*("@&'(),&),).'B'.)0/)0+,@%&C)&1,&)$,I'3)'M+.%@%&)@0*@."3%0*3)(%//%@".&?
&1'-'),-'),)*"$#'-)0/)/%*(%*43)&1,&),-')3&%..)2"%&')@.',-)0)#'4%*?),..)0/)&1')30"-@'3)&1,&)
$'*&%0*'()&1')%*%&%,.)('3%4*)0/)&1C3&'$)('3@-%#'()%&),3),)$0*0.%&1:)91%3)%3)&0)#')'M+'@&'(),3)
$0*0.%&13),-')$"@1)2"%@I'-),*()3%$+.'-)&0)%*%&%,..C)('B'.0+)&1,*)$%@-03'-B%@'3:)E@@0-(%*4)&0)
5,-&%*)P06.'-?)%*)1%3),-&%@.')50*0.%&1)P%-3&?)ST,U.$03&),..)&1')3"@@'33/".)$%@-03'-B%@')3&0-%'3)
3&,-&'()6%&1),)$0*0.%&1)&1,&)40&)&00)#%4),*()6,3)#-0I'*)=!$%#!9)1'3')3,$')3C3&'$3)-',@1'(),)
+0%*&)61'-')&1%3)%*%&%,.)$0*0.%&1)6,3)*0).0*4'-),++-0+-%,&')%*)%&3)@"--'*&)3&,&':)N,3&.C?)$,*C)0/)
&1')0-4,*%X,&%0*3)&1,&)('&,%.'()&1%3)*"()&0)'B0.B')/-0$)&1'%-)%*%&%,.)$0*0.%&1)3%$%.,-.C)@,+&"-'3)
.%$%&,&%0*3)0/)&1')$%@-'0)(3-B%@)

5.1 Architectural Discussion

. /0/0%1232('45%6

E)@0$$0*)+,&&'-*)1'-'?),.-',(C)$'*&%0*'(?)%3),++.%@,&%0*3)#'%*4)#"%.&)/%-3&),3),)$0*0.%&1:)K1'*)
0-4,*%X,&%0*3),-')3$,..?)6%&1),).%$%&'()*"$#'-)0/)&',$3)6103')$'$#'-3)$,C)@0*&-%#"&')&0)
$".&%+.')3'-B%@'3?)&1').,@I)0/)@.',-)#0"*(,-%'3)@,*)#'),*),(B,*&,4':)[06 'B'-?)0*')@.',-)-'3".&)
0/)&1%3)-'3',-@1)%3)&1')'B'*&",.)*"()/0-)&1'3')#0"*(,-%'3:)H&),++',-3)&1%3)*"()%3)('-%B'(,3)
$"@1?)%/)*0)&0-' ?)/-0$)&1')@0$+.%@,&%0*3)0/),)4-06%*4)0-4,*%X,&%0*3)%&1)*"%3)/-)0$-06%*4)
,++.%@,&%0*?)&10"41)&1')&60),-')%*&-%*3%@,..C).%*I'(:)E3)3&,&'()#C)A#'-?)S0-4,*%X,&%0*3),(0+&)
$%@-03'-B%@'3)/0-),*)0+'-,&%0*,.)#'*'/%&),&)&1')'M+'*3')0/)+'-/0-$,*@':"&#!)!%$%.,-)3'*&%$'*&3)
6'-')(%3@"33'()#C)0&1'-)0-4,*%X,&%0*3?)3"@1),3)!10+%/C?)&10"41).'33)(%)-'@&.C:
)
50*0.%&13)6'-'),.30)0#3'-B'()&0)#'),(,+&,#.')/0-)(%//'-'*&)3C3&'$3:)P0-)'M,$+.'?)!.,@I)
('$0*3&-,&'()&1,&)8H^8=) .%$%&,&%0*3)0/)$0*0.%&1)@,*)#')0B'-@0$':)50("..,-)$0*0.%&13),.30)
,..06)3C3&'$3)&0)$,%*&,%*)&1')#'*'/%&3)0/),)$0*0.%&1)61%.')%$+03%*4)*"('()#0"*(,-%'3:);%&1"#)
3'+,-,&'.C)('$0*3&-,&'()106)#0"*(,-%'3)@,*)#')@-',&'()%*),)$0*0.%&1?)0"&3%(')0/)&1')
,-@1%&'@&"-,.)('3%4*)%&3'./:)E..)&1'3')'M,$+.'3)%*(%@,&')/.'M%#%.%&C,)@1%&3)&0)/),)
3C3&'$)$,C)'B0.B')61%.')$,%*&,%*%*4)$"@1)0/)&1')0-%4%*,.)('3)%4*:

./ 0/,%1';$28"$<';"8 %

91'),(0+&%0*)0/)$%@-03'-B%@'3)+-0B%('3),).'33)@0*@-'&'?)#"&)'2",..C)*0&,#.'?)@0*&-,3&)&0)&1')
%*%&%,.)$0*0.%&1%@)('3%4*3:)91'-'),++',-3)&0)#'),)4'*'-,.)+"31)&06,-(3)&1%3)('3%4*)+,&&'-*)&0)
,((-'33)&1')%33"'3)@,"3'()#C)&1')$0*0.%&1:)[06'B'-?)&1%3)@1,*4')%3)*0)"*%B'-3,.:)!10+%/C)
'M+.%@%&.C)@0*3%('-3)&1%3)$%4-,&%0*),*()-'\'@&3)%&:);%&1"#),.30)3"$3)&0)+-'/'-),*?),&).',3&)+,-&%,.?)
$0*0.%&1%@)('3%4*:)E*()'B'*)6%&1)&103')&1,&)$%4-,&'(?)$".&%+.')0-4,*%X,&%0*3)(%3@"33'()%33"'3)
-'4,-(%*4)&1')"3')0/)$%@-03'-B%@'3:)A#?)=00-=,31?),*()H3&%0)-'/'--'()&0)&1'),((%&%0*,.)
@0$+.'M%&C)%*1'-'*&)%*),)$%@-03'-B%@')('3%4*:)H3&%0),*()E$,X0*)D-%$')I%('0)+-0B%(',*)

'M,$+.')61'-'),*),++.%@,&%0*)%3)"*,#.')&0)4,%*)&1')#'*'/%&3)0/)&1%3),((%&%0*,.)@0$+.'M%&C:)
J'C0*()&1'3')%33"'3?)&1')(%3@"33%0*3),-0"*()$%@-03'-B%@'3?)+,-&%@".,-.C)61C)$%@-03'-B%@'3),-')
"&%.%X'(?)6,3),.30)*0&,#.':)E3)('3@-%#'()#C)A#'-?),)+-%$,-C)#'*'/%&3)0/)0C%*4)&1%3),-@1%&'@&"-')
%3)0+'-,&%0*,.)3@,.%*4?)%:':)&1'),#%.%&C)&0)$,*,4')1"*(-'(3)&0)&10"3,*(3)0/)('B'.0+'-3)60-I%*4)
0*),)3%*4.')3C3&'$:!" &#))
)
A#'-),.30)@,"&%0*'(),4,%*3&)S+-'$,&"-'),(0+&%0*)0/),)$%@-03'-B%@'),-@1%&'@&"-')&1'-')J-C0*()&1')
,(('()%*/-,3&-"@&"-')@03&3),*()3C3&'$)@0$+.'M%&C?)0-4,*%X,&%0*,.)@0$+.'M%&C)%*@-',3'(),3)6'..:)
H*)A#'-L3)'M+'-%'*@'?)('B'.0+%*4)'B'*)3%$+.')/',&"-'3)@0".()*'@'33%&,&')@0..,#0-,&%0*)#'&6"*)
$".&%+.')&',$3?)6%&1)&1'),(('()&%$'),*()'M+'*3')*'@'33,-C)/0-)3"@1)@0..,#0-,&%0*:)91'C)
6,-* '( )ST/U0-)0-4,*%X,&%0*3)&1,&)1,B'),.-',(C),(0+&'(),)$%@-03'-B%@'),-@1%&'@&"-')&1'-')%3)*0)
&"-*%*4)#,@I:*)K1%.')&1'%-)@-%&%@%3$3),*()'M+'-%'*@'3),-')#'*'/%@%,.)/0-)&1%3),*,.C3%3?)%&)%3),.30
%$+0-&,*&)&0)I*06.'(4')&1,&)&1'C),-')*0&),(B0@,&%*4),4,%*3&)&1%3),-@1%&'@&"-':)R*)&1')
@0*&-,-C?)&1'),-&%@.')61'-')&1%3)%3)(%3@"33'()+-0+03'3)61,&)A#'-)-'/'-3)&0),3),)=0$,%*'*&'()
5%@-03'-B%@'3)E-@1%&'@&"-'?)0-)=R5E:)
)
91'3')(%3@"33%0*3),-0"*()&1').%$%&,&%0*3)0/)$%@-03'-B%@'3),-')*0&,#.'?)*0&),3),)$',*3)&0)
,(B0@,&'),4,%*3&)&1'%-)"3'?)#"&)&0)@0*&-,3&)&1'%-)"3'3)%*)+-,@&%@'),4,%*3&)&1')(%3@"33%0*3),-0"*()
&1'$)4'*'-,..C:)E3),*)'M,$+.'?),)@0$$0*)+1-,3')"3'()61'*)(%3@"33%*4)& 1')"3')0/)
$%@-03'-B%@'3)%3)S$0('-*%)%1%3)%3)@0*&-,3&'()6%&1)&1')@0$$0*)('3@-%+&%0*)0/)$0*0.%&1%@)
3'-B%@'3),3)S.'4,@C:)91)(%3@"33%0*3),#0B')3106)*'%&1'-),-@1%&'@&"-')%3)2"%&')30)#%*,-C:

./0/:%OC)$'!%F$;5'4";4A$%8
J,3'()0*)&1')-'3".&3)(%3@0B'-'(?)&1'-'),-'),)*"'$#'-)0/),-@1%&'@&"-'3)&1,&)@0".()#')@0*3%('-'()
S1C#-%(:V)P%-3&?)&1')'M,$+.')0/)Z'&/.%ML3)EDH)#,@I*(:)E3)('3@-%#'()#C)&1')@0$+,*C?)&1%3)%3),*)
'M,$+.')0/),)$0*0.%&1)#'%*4)&-,*3%&%0*'()&0),)$%@-03'-B%@':)[06?)&1')$0*0.%&1)%*)2"'3&%0*)
6,3),)S3'-B'-.'33)$0*0.%&1?V)61%@1)%3)@0&3,-%.C&1')3,$'),3),)&-,(%&%0*,.)$0*0.%&1:)H&)%3)
"*@.',-)&1')('4-")&0)61%@1)&1%3)3'-B%@')@0*/0-$3)&0)61,&)%3)$',*&)%*)&1%3)+,+'-)#C)&1')&'-$)
$0*0.%&1:)91'),-&%@.'),.30)('&,%.3)&1')&-,*3%&%0*)0/)&1%3)S$0*0.%&1V)%*&0),)3%*4.')$%@-03'-B%@':)H
6'-'),)$0*0.%&1?)&1%3)60".()&1'*)3"44'3&),)$0*0.%&1%@)3'-B%@')61%@1)60".()%$+.C),*)0B'-,..)
1C#-%(),-@1%&'@&"-':)91')'B%('*@')/0-)0-),4,%*3&)&1%3)%*&'-+-'&,&%0*)(0'3)*0&)'M%3&)%*)&1'),-&%@.'?)
$,I%*4)%&3)"3'),3)+-00/)0/),)1C#-%(),-@1%&'@&"-')3"3+)'@&:)
)
91')'M,$+.')+-0B%('()#C)E$,X0*)D-%$')I%('0)%3)$0-')@.',-:)['-'?),)3'-B%@')&1,&)1,()
+-'B%0"3.C)#"*)('@0$+03'()%*&0)(%//'-'*&)$%@-03'-B%@'3?)#'%*4)&1'*)&-,*3%&%0*'()&0),)3%*4.')
$0*0.%&1%@)3'-B%@':)910"41?)&1'3')$%@-03'-B%@'3)6'-')3'-B'-.'33)/"*@&%)61%@1)32?),(3)&0),)
3%$%.,-)%33"')+03'()#C)&1')Z'&/.%M)'M,$+.':)K1'-')&1%3)(0'3)('3'-B')$0-')@0*3%('-,&%0*?)
106'B'-?)%3)&1')/,@&)&1,&)&1%3)'*()3'-B%@')6,3),#.')&0)#')('@0$+03'()/"-&1'-:)J,3'()0*)&1')
(/%*%&%0*)0/)1C#-%()('&,%.'(')',-.%'-)%*)&1%3)+,+'-?)&1%3)@)'B%('*@'),4,%*3&)&1')'*()
3'-B%@')#'%*4)@0*3%('-'(),)&-'")$%@-03'-B%@':
)
;%&1"#)+-0B%('3)&1')$03&)'M+.%@%&)'M,$+.')0/),*)0-4,*%X,&%0*)@0*3@%0"3.C)'$+.0C%*4),)$0*0.%&1)
6%&1)$%@-03'-B%@'3:)!%$%.,-)&0)0&1'-)0-4,*%X,&%0*3?)&1%3),++',-3)&0)1,B')#"*),&).',3&)%*%&%,&'(),3)
,)$%4-,&%0*)&06,-(3)$%@-03'-B%@'3Q)#'&?)"*.%I')0&1'-3?)&1%3)%&)40,.)%3)*0&)3%$+.C),)$0*0.%&1)

-'+.,@'$'*&:)91%3)%3)"3'/".)@0*&'M&?)+,-&%@".,-.C)3%*@').,&'-),-&%@.'3)+"#.%31'()#C);%&1"#)0*.C)
$'*&%0*)&1'%-)$0*0.%&1%@)('3%4*:)91%3)%*(%@,&'3)@%-@"$3&,*@'3)61'-')%&)%3)+-'/'--'()&0)$,%*&,%*),
$0*0.%&1)61%.')'$+.0C%*4)3+'@%/%@)3'-B%@'3)'M&)-*,.)&0)%&:

./0/=%9$*38'4'23*(%F$;5'4";4A$%'8

!0$'&1%*4)&1,&)6,3)*0&)@,+&"-'()%*)&1%3)-'3',-@1?)&10"41)%$+.%'(?)%3)106)&-,*3%&%0*,.)
,-@1%&'@&"-'3)*'@'33%&,&'),)@"3&0$)('3%4*:)91%3)-'3".&3)%*)B,-C%*4)1C#-%(),-@1%&'@&"-'3)&1,&)
'B0.B'),3)&1')&-,*3%&%0*)+-04-'33'3:)91')3&,4'3)0/)&1%3)'B0."&%0*?),3)6')(,&).',(3)&0)&1')
(%//'-'*&)$%@-03'-B%@')^)$0*0.%&1)^)1C#-%()B,-%,&%0*3)%3)0/)+,-&%@".,-)%*&'-'3&)/0-)/"&"-')-'3',-@1:)
K')#'.%'B')&1'3')&-,*3%&%0*,.)('3%4*3)@,*)+-0B%(')/"-&1'-)%*3%41&)%*&0)#0&1)+0&'*&%,.)1C#-%()
,-@1%&'@&"-,.)+,&&'-*3?)#"&),.30)'."@%(&)-@1%&'@&"-,.)('3%4*3),((-'33)+-0#.'$3),&)(%//'-'*&)
3&,&'3)

5.2 Interpretation of Results

91'3')-'3".&3)3106),*)'B0."&%0*)0/),-@1%&'@&"-'3),*(),-@1%&'@&"-,.)*"'(3),3),)3C3&'$)'B0.B'3:)
E3),.-',(C)3&,&'(?)3C3&'$3)&C+%@,..C)#'4%*)6%&1),)$0*0.%&1%@),++-0,@1:)91%3),..063)&1'$)&0)4-06)
,*()3@,.')$0-')2"%@I.C),*()6%&10"&)*"'(%*4)&0)*,B%4,&'),-0"*()#0"* (,-%'3)%*)&1')+-0@'33:)
91'*?)&1'3'),++.%@,&%0*3)-'2"%-'),)-',-@1%&'@&"-'),6,C)/-0$),)&-,(%&%0*,.)$0*0.%&1:)E3)&1')@,"3')
0/)&1%3)*"'()%3).,-4'.C)0-4,*%X,&%0*,.?)&1'3')%33"'3)@,*)#')+-"$+&'(:)P0..06%*4)&1'),++-0,@1'3)
('&,%.'()#C)!10+%/C?)A#'-?),*();%&1"#?),3'-B%@')^)(0$,%*)0-%'*&'(),++-0,@1)@,*)#')"3'(:)91,&)
60".()$',*)%('*&%/C%*4)&1')#0"*('()@0*&'M&3)0/),*),++.%@,&%0*),*()('3%4*%*4),-0"*()&1'$:)91%3)
310".()*0&)0*.C)#')(0*')',-.C)#"&),.30)-'B%3%&'()+'-%0(%@,..C)&0)@,+&"-')0-4,*%X,&%0*,.),*()
3C3&'$)@1,*4'3:)JC)(0%*4)30?)&1%3)60".()+-0B%(')%$+.%@%&)#0"*(,-%'3),*()%*/0-$)106)&1')
,++.%@,&%0*)%3)3&-"@&"-'(),3)%&)(4-B%3'3:)

./,/ 0%)8"$<*4'238%*3!%?A##"84'238

91')+-%$,-C)0#3'-B,&%0*)0/)&1%3)-'3',-@1)%3)106)&1')3&-"@&"-')0/),)30/&6,-'),-@1%&'@&"-'),//'@&3)
&1')3C3&'$)%&3'./:)91%3)@,*)#')3"')%*)&1').%$%&,&%0*3)&1,&)0-4,*%X,&%0*3)'M+'-%'*@')/0-)',@1)
,-@1%&'@&"-'),*()61'*)^)106)&1'3').%$%&,&%0*3)+-'3'*&)&1'$3'.B')$,..'-)0-4,*%X,&%0*3?)
+,-&%@".,-.C)0*'3)&1,&),-')-'.,&%B'.C)*'6?)*"'()&0)',3%.C)&-,B'-3')#'&6"*)#0"*(,-%'3)3%*@')&1')
'$+.0C"3)&1'$3'.B'3)0/&*)60-I)#'&6"*)#0"*(,-%'3 :)E3)0-4,*%X,&%0*3)4-06?)&1')*"()/0-)
3&-"@&"-')#'@0$'3)$0-')*'@'33,-C:)91')&-,*3%&%0*3)0#3'-B'()%*)&1%3)-'3',-@1?),6,C)/-0$)&1')
%*%&%,.)$0*0.%&1?)3"I),)$',*3)&0)+-0B%(')&1%3)3&-"@&"-':)K1%.')#0"*(,-%'3)@,*)#'),((')&0),)
&-,(%&%0*,.)$0*0.%&1?)6')#'.%'B')&1')#'@,"3')&1'C),-')*0&)%*1'-'*&)&0)&1')('3%4*)&1'3'),-')
".&%$,&'.C)6',I)#0"*(,-%'3:)91%3)%3)61C)&1'),-@1%&'@&"-'3)&1,&)0-4,*%X,&%0*3)&-,*3%&%0*'()&0),..)
1,B')@.',-)#0"*(,-%'3)#"%.&)%*&0)&1'%-)('3%4*:)J0&1)$%@-03'-B%@'3),*()$0(".,-)$0*0.%&13)
'@'33%&,&')#0"(,-%'3?)61%@1),..06)&1')0-4,*%X,&%0*,.)3&-"@&"-')&0)$0-')@.03'.C)$%--0-)&1')
3C3&'$L3)('3%4*:)[06'B'-?)6')#'.%'B')&1')%33"'3)3"')%*)+,-&%@".,-.C).,-4')0-4,*%X,&%0*3?)
*,$'.C)Z'&/.%M),*()A#'-?),-')("')&0)&1'3')3&-%@&)#0"*(,-%'3)%*1'-'*&)%*)$%@-03'-B%@'3))
/0"*()(%//%@".&%'3)*,B%4,&%*4)(0$,%*).'B'.)@1,*4'3)&1,&),//'@&'()$".&%+.')$%@-03'-B%@'3?)61'-',3)
Z'&/.%M)/0"*()(%//%@".&%'3)*,B%4,&%*4)&1')(,&,)#0"*(,-%'3)0/)&1'%-)$%@-03'-B%@'3:)J0&1)0/)&1'3')
%33"'3),-03')/-0$)&1')('3%4*)0/),)$%@-03'-B%@'),-@1%&'@&"-'*4).003'.C)@0"+.'()6%&1)3&-%@&)
3'-B%@'),*()(,&,)#0"*(,-%'3:)K1,&)&1%3)3"44'3&3)%&)0/0.(G)]%_)&1'-')%3),).%$%&)&0)61'-'),)

&-,(%&%0*,.)$%@-03'-B%@'3)('3%4*)+-0B%('3)#'*'/%&3?),*()]%%_)&1'-')%3),*),-@1%&'@&"-,.)('3%4*)4,+)
61'-')&1'3').%$%&,&%0*3)/0-)+,-&%@".,-.C).,-4')3C3&'$3),-')%*1'-'*&.C),((-'33'(),3),)+,-&)0/)&1')
('3%4*:)
)
!%*@')&1')&-,(%&%0*,.)$0*0.%&1)(0'3)*0&)+-0B%(')&1')3&-"@&"-')&1,&)%3)-'2"%-'()&0)/@%,%*&,&'),)
.'B'.)0/)3@,.'?)'B0."&%0*)#'@0$'3)%*'B%&,#)K1%.'),)&-,*3%&%0*)$%@-03'-B%@'3)%3)@"--'*&.C)&1')
$03&)+0+".,-)'B0."&%0*?)%&)%3)*0&)*'@'33,-%.C)&1')$03&%,&')E)$0(".,-)$0*0.%&1?)$C#-%()
$0*0.%&1^3'-B%@'3),-@1%&'@&"-'V),3)%$+.'$'*&'()#C);%&1"#?)0-),*)0B'-,..)1C#-%(),-@1%&'@&"-'),..)
,((-'33)&1').%$%&,&%0*3)0/)&1')&-,(%&%0*,.)$0*0.%&1)%*)&1'%-)06*)6,C3:)K1%@1),-@1%&'@&"-')&0)"3')
,*()61'*)('+'*(3)1',B %.C)0*)#0&1)&1')3C3&'$),*()0-4,*%X,&%0*:)91'3')-'3".&3)3106)"3)&1,&),)
.'B'.)0/)0-4,*%X,&%0*,.),*()0+'-,&%0*,.)3@,.')%3)*'@'33,-C)#'/0-'),)$%@-03'-B%@'),-@1%&'@&"-')%3)
,++-0+-%,&')61%@1)$,C)-'3".&)%*),)3C3&'$)-',@1%*4)&1').%$%&,&%0*3)0/)&1'%-)%*%&%,.)$0*0.%&1)#"&)*
#'),&),)3@,.')C'&)61'-')$%@-03'-B%@'3),-')&1')$03&),++-0+-%,&')0+&%)0*:
)
K')/%*()&1')30."&%0*)+-'3'*&'()#C)A#'-)&0)#')0/)+,-&%@".,-)%*&'-'3&:)J'@,"3')0/)%&3),#%.%&C)&0)
,((-'33)&1') @1,..'*4'3)0/)$%@-03'-B%@'3),&),).,-4')3@,.'?)6')#'.%'B')*),#3&-,@&%0*)0/)&1)-
=R5E),++-0,@1)@0".()#')"&%.%X'(),&)B,-%0"3)3&,4'3)0/),*),++.%@,&%0*:)H/),)3C3&'$)%3)#-0I'*)%*&0)
,++-0+-%,&')(0$,%*3?)61%@1),-'),4*03&%@)&0)',@1)0&1'3')(0$,%*3)@,*)#')('3%4*'(),3),)
@0..'@&%0*)0/)$%@-03'-B%@'3?),)3%*4.')$0*0.%&1)+-0B%(%*4)$".&%+.')&-B%@'3,&%0*),)0/)&1')
&60:)E)#'*'/%&)0/)&1%3),++-0,@1)%3)&1,&)%/)3$,..'-)0-4,*%X,&%0*3)&1,&),-')*0&)%*),)+.,@')61'-')
$%@-03'-B%@'3),-'),++-0+-%,&')"&%.%X'(),)* 0-%'*&'(),++-0,@1?)&1'C)6%..)#')$0-')+-'+,-'()
/0-),)&-,*3%&%0*)E)%.%$%&,&%0*)&0)&1%3),++-0,@1)%3)&1,&)%&)%3)*0&)&%'()&0)&1'),-@1%&'@&"-,.)('3%4*
61%@1)@,*).%$%&)%&3)%$+,@&:)K')-'@'3$$@(1)%*&0)&1%3),++-0,@1),3)6'..),3)&1').0&4$)
'//'@&3)0/)=R5E),&)A#'-:)

5.2 Research Limitation

91'-'),-')&60)+0&'*&%,.)%33"'3)6%&1)&1'),-4"$'*&),#0B':)91')/%-3&)%3)&1').,@I)0/)'M+.%@%&)'B%('*@':)
K1%.'),)3%*4.'),++.%@,&%0*)6,3)/0"'*()&0)"3')&60)$0*0.%&1%@)3'-B%@'3?)&1,&)(0'3)*0&)+-0B%(')
'*0"41)%*/0-$,&%0*)&0)#"%.(),*),-@1%&'@&"-,.)('3%4*:)E*()61%@1)%3)&1'.C)&1%3)%3)$0-')@0$$0*?)
0-4,*%X,&%0*3)1,B')*0&)#'*)/0-&1-%41&)6%&1)('&,%.3)0/)&1'%-),-@1%&'@&"-,.)('3%4*3:)A.&%$,&'.C?).%&&
@,*)#')3,%()@0*@."3%B'.C),#0"&),-@1%&'@&"-'3)%*)+-,@&%@')6%&10"&)(%-'@&)%*3%41&)%*&0)&1'3')
0-4,*%X,&%0*3),*()&1'%-)3C3&'$
)
91')3'@0*()%33"')6%&1)&1%3),*,.C3%3)%3)&1').%$%&'()B%'6)0*)30/&6,-'),-@1%&'@&"-'3:)H&)@0".()#')
,-4"'()&1,&)61,&)%3)&-C%*4)&0)#')'."@%(,&'()1'-')%3),)!'-B%@')R-%'*&'()E-@1%&'@&"-'?)0-)!RE:)
K1%.'),-4",#.C)&-"'?)&1%3)+03%&%0*)0*.C)/0@"3'3)0*)&1')1%41&)6%&1)B%&1,*)&1')
,-@1%&'@&"-,.)('3%4*)%&3'./:)P0-)%*3&,*@'?)$%@-03'-B%@'3)&1'$3'.B'3),-')+,-&)0/)!RE)-,&1'-)&1,*),)
3'+,-,&'),-@1%&'@&"-,.)@,&'40-C:)!%$%.,-.C?)&1'-'),-')$,*C)610)$,C),-4"'),*C)('3%4*)&1,&)
"&%.%X'3)$".&%+.')('+.0C,#.')"*%&3)%3),)$%@@03'-B%@'3)&1'-':)K1%.')$0-')('&,%.3),3)&0)61C)
&1%3)%3)%*@0--'@&)1,3),.-',(C)#'*)(%3@"33'(?)%&)%3)60-&1)*0&%*4)106)&1').,@I)0/),*),4-"()"+0*)
('/%*%&%0*)/0-),)$%@-03'-B%@')$,I'3)*"',*@'()(%3@"33%0*3)3"@1),3)&1%3)0*')$0-')(%//%@".&:

5.3 Qualifiers

91%3)-'3',-@1)/0@"3'3)0*).,-4' C3@,.'),++.%@,&%0*3?)61%@1)%*@."(')#0&1)1%41)&-,//%@),3)6'..),3),)
.,-4')0-4,*%X,&%0*,.)3&-"@&"-':)!C3&'$3)&1,&)(0)*0&)-'@'%B'),)1%41)B0."$')0/)&-,//%@),-').%I'.C)*0&)
40%*4)&0)#'*'/%&)/-0$),)$%@-03'-B%@'),-@1%&'@&"-':)!%$%@-03'-B%@'3)#'*'/%&)0-4,*%X,&%0*3)
61'-')3'-B%@'),*()&',$)#0"*(,-%'3)*"()&0)#')-'%*/0-@'(:)K1%.')%&)%3)%$+0-&,*&)&0)%('*&%/C)61'-')
$%@-03'-B%@'3)$,C)*0&)#')&1')-%41&)/%&?)'B'*)/0-)3$,..'-),++.%@,&%0*3?)&1%3)+,+'-)%*&'*(3)&0)"3')
&103')'M,$+.'3)&0)/"-&1' -)%('*&%/C).%$%&,&%0*3)0/)&1%)3)('3%4*:)
)
[06'B'-?)('3+%&')&1%3)-'3',-@1)&,I%*4),)4'*'-,.%X'(),++-0,@1?)&1'-'),-')&1-',&3)&0)&1')B,.%(%&C)0/)
%&3)@0*@."3%0*3)&1,&)$"3&)#'),((-'33'(:)P%-3&)%3)&1'),@I*06.'(4'$'*&)&1,&),..),-@1%&'@&"-,.)
('3%4*3)1,B')#'*'/%&3),*()(-,6#,@I3:)!%$+.C)#'@,"3'),*)0-4,*%X,&% 0*?)0-)$".&%+.')
0-4,*%X,&%0*3?),(0+&)0-)&-,*3%&%0*)&0),)@'-&,%*)('3%4*?)&1,&)#0&1)(0'3)*0&)'M+.%@%&.C)%*(%@,&')&
6,3)&1')-%41&)('@%3%0*?)*0-)(0'3)%&)@0*/%-$)&1,&)&1')-'3".&%*4)('3%4*)%3),++-0+-%,&')/0-),*C)0&1'-)
3C3&'$:)E..)3C3&'$3),-')"*%2"'),*()%&)%3)*0&)&1')%*&'*&)0/)&1%3)&1'3%3)&0),(B0@,&')/0-)3,)$0)*'
/%&3),..V),++-0,@1:)P0-)%*3&,*@'?),)-'.,&%B'.C)3%$+.')3C3&'$)/-0$),)3$,..)0-4,*%X,&%0*)$,C)*0&)
-'@'%B')&1')0+'-,&%0*,.)0-)0-4,*%X,&%0*,.)#'*'/%&3)0/)$%@-03'-B%@'3:)E((%&%0*,..C?)A#'-)+-0B%('3)
,*)'M,$+.')61'-')&1')3%X')0/),*)0-4,*%X,&%0*)*'@'33%&,&'3)3+'@%,.)@0*3%('-,&%0*3)$,*C)
3C3&'$3)6%..)*'B'-)*"()&0),@@0"*&)/0-:)N,3&.C?)&1').%$%&'()-'3',&)3) +-0B%(%*4)*C)('/%*%&%B')
@0*@."3%0*3)(%//%)@".&:)

5.4 Impact and Next Steps

91')".&%$,&')40,.)0/)&1%3)-'3',-@1)%3)&0)'*@0"-,4'),((%&%0*,.)@0*3%('-,&%0*)0*),-@1%&'@&"-,.)
('3%4*3:)H&)%3)&1')0+%*%0*)0/)&1'),"&10-3)&1,&)&1')'B%('*@')+-'3'*&'()6,--,*&3),)$0-')@-%&%@,.).00I)
,&)&1')&-,(%&%0*,.)$%@-03'-B%@')('3%4*?)61'*)&0),++.C)%&%)&1,&)&1,C)#'),.&'-'(:)A&%.%X%*4)&1')
3"44'3&%0*3),#0B')6%..),.30)+-0B%(')0-4,*%X,&%0*3)6%&1),)3&,-&%*4)+0%*&)&1,&)%3)$0-')+-'+,-'()/0-)
"&%.%X%*4)&-,(%&%0*,.)$%@-03'-B%@'3)61'*)&1'),++.%@,&%0*),*()0-4,*%X,&%0*)*'@'33%&,&')&1,&)('3%4
91%3)-'3',-@1)('$0*3&-,&'3)&1')@.',-)4,+)%*)&1'),B,%.,#.')30/&6,-'),-@1%&'@&"-,.)('3%4*3:)91%3)%3)
@,"3'()#0&1)#C)&1'),$#%4"%&C)0/)61,&)@0*3&%&"&'3),)$%@-03'-B%@'),3)6'..),3)&1').,@I)0/)-'3',-@1)
0*),)$%M'(),++-0,@1)!RE)&1,&)#'*'/%&3)/-0$)#0&1)$0*0.%&13),*$%@-03'-B%@'3:)K%&1),)$0-')
@-%&%@,.),++-0,@1)&06,-(3),-@1%&'@&"-,.)('3%4*3?)0-4,*%X,&%0*3)$,C)#')#'&&'-)3"%&'()&0),((-'33)
#0&1)0-4,*%X,&%0*,.),*()0+'-,&%0*,.).%$%&,&%0*3:
K%&1)&1')-'3".&3)&1%3)-'3',-@1)6,3),#.')&0)4.',*)/-0$)1%41).'B'.)-'3',-@1?)%&)%3)@.',-)&1,&),)$0-')
('&,%.'()3&"(C)%3)*'@'33,-C:)Z0&)0*.C)$,C)/"&"-')-'3',-@1)%('*&%/C)$0-'),@@"-,&'.C)106)3C3&'$)
,-@1%&'@&"-'3)(%//'-)%*)+-,@&%@')/-0$)&1'%-)(3'0-,%+&%0*3)@0*&)%&,.30)+-0B%(')$0-')
%*/0-$,&%0*)0*)&1')-',30*%*4)#'1%*(),-@1%&'@&"-,.)('@%3%0*3:)E)2"'3&%0*)&1%3)-'3',-@1)#-%*43)&0)
.%41&)%3G)61,&)('4-'),-')$%@-03'-B%@'3),)30."&%0*)&0),++.%@,&%0*)3@,.%*4)B'-3"3)0-4,*%X,&%0*,.)
3@,.%*4:)H/)%&)%3)+-%$,-%.C)&1')#&)3"-44'3&'(!"&#?),-')&1'-')30."&%0*3)&1,&),((-'33).,-4')
0+'-,&%0*,.)3@,.')%33"'3)6%&10"&)%*@"--%*4)&1'),(('()@0$+.'M%&C)%*1'-'*&)%*)$%@-03'-B%@'3`)
H('*&%/C%*4)#0&1)61,&)%33"'3)0-4,*%X,&%0*3)/,@'),3)6'..),3)106)&1'C),++-0,@1)30."&%0*3)@,*)
+-0B%(')4"%(,*@')0*)&1')/"&"-')'B 0."&%0*)0/)30/&6,-'),-@1%&'@&"-':

6. Conclusion

91%3)+,+'-)%*&'*('()&0)%('*&%/C)&1')4,+3)%*)30/&6,-'),-@1%&'@&"-,.)('3%4*3),*()106)&103')4,+3)
,-'),((-'33'()%*)+-,@&%@':)91%3)/0@"3'()0*)&1')&60)$,%*),-@1%&'@&"-,.)('3%4*3?)$%@-03'-B%@'3)
,*(),)$0*0.%&1:)A*('-3&,*(%*4)&1'3')&60)('3%4*3),@@0"*&)/0-)0.,-)0++03%&,)++-0,@1'3) Y&0)
0B'-3%$+.%/C?)('+.0C%*4),)3"%&')0/)3$,..)"*%&3)0-),)3%*4.').,-4)%*&'*('()&0)3106)
3C3&'$3)%*)+-,@&%@')%$+.'$'*&),)@0$#%*,&%0*)0/)&1'3'),++-0,@1'3:)91%3)6,3)(0*')&1-0"41),)
&,%.0-'()3C3&'$,&%@).%&'-,&"-')-'B%&'@"3'()0*)&'@1*%@,.)6-%&%*43)#C)&1')0-4,*%X,&%0*3)
&1'$3'.B'3:) K')(%3@0B'-'()B,-%0"3),-@1%&'@&"-,.)('3%+&%0*3)6'..),3) +,&&'-*3)(-%B%*4)
,-@1%&'@&"-,.)@1,*4'3:)K')/0"*()0-4,*%X,&%0*4)06 &1)&0)#')&1')(-%B%*4)/0-@')%&1)#0
,-@1%&'@&"-,%*)+0%*&3),*(@'33,-C) &-,*3%&%0*3:)K')3106'()106)0-4,*%X,&%0*3),(,+&'()
$0*0.%&13)-)&-,*3%&%0*'()&0)$%@-03'-B%@'3)&0)B'-@0$')&1'3')+,%*)+0%*&3)3,3&.C?)6')(%3@-'()
,).%$%&)&0)&1')#'*'/%&3)0/)$%@-03'-B%@'3)&0)B'-&,%*)0-4,*%X,&%0*3@,.')%3)-',@1'(:)K')
3"44'3 &'(),)4'*'-,.%X'(),-@1%&'@&"-,.) .%/'@C@)0,*()-'@0$$'* ('()+-"$+&%*4)#0"*(,-C)%33"'3)
',-.C)6%&1),)(0$,%*)^)3'-B%@')0-%'*&'(),++-0,@1)

>(A(0($"(- .

TdU) Kirsten Westeinde. (2019, February 21). *Deconstructing the Monolith: Designing Software that Maximizes Developer Productivity*. Shopify. https://www.shopify.engineering/deconstructing-monolith-designing-software-maximizes-developer-productivity
TbU)5'-30*?)D:)]bcdg?)Z0B'$#'-)q_:)='/%*%*4)5%@-03'-B%@'3:)!7H)J.04:)
1&&+3G^^666:3'%:@$":'("^#.04^('/%*%*4^$%@-03'-B%@'3^
TpU)5'-30*?)D:)]bcdg?)Z0B'$#'-)q_:)5%@-03'-B%@'3)J'C0*()&1')[C+'G)K1,&)r0");,%*),*()K1,&)
r0")N03':)!7H)J.04:) 1&&+3G^^666:3'%:@$":'("^#.04^$%@-03'-B%@'3^(C&1'C1C+'G1,&C0"O
4,%*(O*(G1,&C0"O03')
TpU)-.,*?)=:?)s)!1,6?)5:)]dkke_:)E*)H*&-0("@&%0*)&0)!0/&6,-')E-@1%&'@&"-':)
1&&+3G^^666:@3:@$":'("^,/3^@3^+-0\'@&^,#.'^/&+^%*&-0t30/&,-@1^%*&-0t30/&,-@1:+(/
TgU)N'6%3?)u:?)s)P06.'-?)5:)]bcde?)5,-@1)bg_:)5%@-03'-B%@'3:)5,-&%*/06.'-:@0$:)
1&&+3G^^$,-&%*/06.'-:@0$^,-&%@.'3^$%@-03'-B)%@'3:1&$.
TqU)80*6,C?)5'.B%*)]dkqi_:)v[06)(0)@0$$%&&"3)%*B'*&v)]D=P_:)=,&,$,&%0*Ybd)
TnU)Z'6$,*?)!,$:)50*0.%&1)&0)$%@-03'-B%@'3G)'B0."&%0*,-C)+,&&'-*3)&0)&-,*3/0-$)C0"-)$0*0.%&1:)
Rwj'%..C)5'(%,?)bcdk:
T U)B-&,* ?)=:?)s)5,I,40* ?) :)]bcbc?)5,-@1)dc_)j',(C)/0-)81,*4'3)6%&1)['M,40*,.)
E-@1%&'@&"-'Z'&/.%M9'@1J.04Q)5'(%"$:)'&/.%M&'@1#.04:@0$^-'(C-@1,*4'3 6%&1'M,40*,. O
,-@1%&'@&"-'pdg'@kqnnek)
TkU)K1,&)%3)!'-B'-.'33)E-@1%&'@&"-':)]*:(:_:);004.')8.0"(:)
1&&+3G^^@.0"(:4004.':@0$^(%3@0B'-^3'-B'-.'33 O-@1%&'@&"-'
TdcU)F%&@1'*1,$?)J,-#,-,:)]bccn_:)F%&@1'*1,$?)J:G);"%('.%*'3)/0-)+'-/0-$%*4)!C3&'$,&%@)
N%&'-,&"-')j'B%'63)%*)30/&6,-')'*4%*"-%*4:)7J!7)9'@1*%@,.)j'+0-&)7J!7 bccn dd:))
TddU)(')F0-&?)K:?)s)!@16,*X?)u:)]bcbg?)E"4"3&)bi_:)50B%*4)&1')+"#.%@)!&,@I)RB'-/.06)3%&'3)&0)&1')
@.0"(G)D,-&)b)!&,@I)RB'-/.06:)!&,@I0B'-/.06:#.04:)
1&&+3G^^3&,@I0B'-/.06:#.04^bcbg^ci^bi^$0B%*4"#.%@3&,@I0B'-/.06 3%&'3 &0 &1' @.0"(+,-& d)
TdbU)8-'%M'..?)u:)]bcbd?)u".C)bk_:)!'-B%@')E-@1%&'@&"-'),&)!0"*(.0"(-&)dG)J,@I'*(3)/0-)
P-0*&'*(3:)='B'.0+'-3:30"'*(@.0"(:@0$:)1&&+3G^^('B'.0+'-3:30"'*(@.0"(:@0$^#.04^3'-B%@ O
,-@1%&'@&"-'
TbU)5@H.-0C?)!',*:)S91')!@,-C)91%*4),#0"&)E"&0$,&%*4)='+.0C3:V)!.,@I:'*4%*'-%*4?)bcbe?)
3.,@I:'*4%*'-%*4^&1'3@,-C&1%*4#0"&O"&0$,&%*4O'+.0C3^:)E@@'33'()ck)Z0B:)bcbg)
TcU)J'%X'-?)5:)]bcbp?)='@'$#'-)dk_:)[06)6')0-4,*%X'),*()4'&)&1%*43)(0*')6%&1)
!'-B%@'R6*'-3:)91');%&["#)J.04:)4%&1"#:#.04^'*4%*'-%*4^,-@1%&'@&"-'+&%$%X,&%0*O1O6
0-4,*%X'O*(4'&&1%*43(0*' 6%&13'-B%@'06*'-3^)
TgU)[,((,()?)7%*,3:)S!'-B%@'R-%'*&'()E-@1%&'@&"-'G)!@,.%*4)&1')A#'-)7*4%*'-%*4)80('#,3'),3)
K');-06:V)A#'-:@0$?)i)!'+&:)bcdg?)"#'-:@0$^'* O;^#.04^3'-B%@0-%'*&'(O-@1%&'@&"-'^:)
E@@'33'()k)Z0B:)bcbg)
TdqU);."@I)?)E(,$:)SH*&-0("@%*4)=0$,%*R-%'*&'()5%@-03'-B%@')E-@1%&'@&"-':V)A#'-:@0$?)bp)u".C)
bcbc?)"#'-:@0$^'* O;^#.04^$%@-03'-B%@'@1%&'@&"-'^:)E@@'33'()k)Z0B:)bcbg:
TnU)=1-"B,?)j:?)s)J,..0&?)7:)]bcbc?)!'+&'$#'-)i_:)!',$.'33.C)!6,++%*4)&1')EDH)#,@I'*()0/)&1')
Z'&/.%M)E*(-0%(),++:)Z'&/.%M9'@1J.04Q)5'(%"$:)*'&/.%M&'@1#.04:@0$^3',$.'33.C%*4&1'O
,+%*#,@I'*(0/&1'*'&/.%M*(-0%(O++ p(epdndggdin)

Td i U)F0.*C?)5:)]bcbp_:)!@,.%*4)"+)&1')D-%$')l%('0),"(%0^B%('0)$0*%&0-%*4)3'-B%@'),*()-'("@%*4)
@03&3)#C)kco:)
1&&+3G^^6'#:,-@1%B':0-4^6'#^bcbecndkdgbdck^1&&+3G^^666:+-%$'B%('0&'@1:@0$^B%('0
3&-',$%*4^3@,.%*4 O&1'O+-%$'0B%('0,"(%0 0B%('0$0*%&0-%*4-B%@,*(O'("@%*4 0@03&3 0kc)
R-%4%*,..C)+03&'(G)1&&+3G^^666:+-%$'B%('0&'@1:@0$^B%('0 0$^B%('0 4^3@,.%*4 O&1'O
+-%$'0B%('0,"(%0 0B%('0$0*%&0-%*4-B%@,*(O'("@%*4 0@03&3 0kc)
Td kU)Z:)8:)5'*(0*y,?)8:)J0M?)8:)5,*0.,@1'),*()N:)jC,*?)v91')50*0.%&1)!&-%I'3)J,@IG)K1C)H3&%0)
5%4-,&'()P-0$)5%@-03'-B%@'3)&0),)50*0.%&1%@)E-@1%&'@&"-'?v)%*)H777)!0/&6,-'?)B0.:)pi?)*0:)g?)++:)
dnObb?)!'+&:OR@&:)bcbd?)(0%G)dc:ddck^5!:bcbd:pcicpp)g:
Tbc U)J,&-,?)r:)]bcbg?)!'+&'$#'-)bb_:)[06)Z'&/.%M)E@@%('*&,..C)D-0B'()50*0.%&13)!@,.')J'&&'-)
91,*)5%@-03'-B%@'3:)5'(%"$:)1&&+3G^^$'(%"$:@0$^zC,31#,&-,ddddd^10&/.%M)@@%('*&,.0C
+-0B'(O$0*0.%&13@,.'O#'&&'-O&1,*O$%@-03'-B%@'3 0b3qq/c,c##g)
Tbd U)D,*('C?)E:)]bcbp?)u"*')b_:)J"%.(%*4),)[C#-%()E-@1%&'@&"-'G)[,-*'33%*4)&1')D06'-)0/)
5%@-03'-B%@'3),*()50*0.%&13:)5'(%"$:)1&&+3G^^$'(%"$:@0$^z+,*('C,-+%&ii^#"%Q)%*4#-%Q
,-@1%&'@&1-,-*'33%*4 O&1'O+06'- O0/O$%@-03'-B%@'3 O*(O$0*0.%&13c/nk#pdc(ng)
Tbb U)E*('-30*?)9:)]bcbc?)='@'$#'-_:);%&["#L3)\0"-*'C)&06,-(3)$%@-03'-B%@'3),*()$0-':)
91'j'4%3&'-:@0$Q)91')j'4%3&'-:)
1&&+3G^^666:&1'-'4%3&'-:@0$^bcbc^db^cd^4%&1"#3t\0"-*'Ct&06,-(3t$%@-03'-B%@'3^
Tbp U)P06.'-?)5:)]bcdg?)u"*')p_:)50*0.%&1P%-3&:)5,-&%*/06.'-:@0$:)
1&&+3G^^$,-&%*/06.'-:@0$^#.%I%^50*0.%&1P%-3&:1&$.

)

A Decision Framework for Prioritizing Technical Debt in Agile Development

Submitted in partial fulfillment of the requirements for the

degree of

Master of Software Engineering

Alaa Mahmoud Hussein

Master of Science, Software Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

December degree will be awarded, 2025

© Alaa Mahmoud Hussein
2025 All Rights Reserved

Keywords: technical debt prioritization, decision frameworks in agile, defect triage frameworks

Acknowledgments

I am grateful to my thesis advisor, Professor Bradley Schmerl, for his guidance, feedback, and patience throughout this work. As both the instructor for 17-679-A and my thesis advisor, he helped provide insight into each section of the paper, from the earliest proposal to the final draft.

I would also like to thank my colleagues in 17-679-A for a semester of thoughtful discussions and generous peer reviews. Their comments on successive drafts and ideas significantly improved the clarity and quality of this thesis.

This work would not have been possible without the support of my company, which allowed me to conduct a pilot study and collect data in a real project setting. I am especially grateful to the teammates who participated in the study and shared their time, experiences, and backlog artifacts, making it possible to ground this research in actual team practices.

I am grateful to the faculty and staff of the Master of Software Engineering program at Carnegie Mellon University for creating an environment that supports both rigorous study and applied research. Their administrative help and flexibility made it feasible to balance coursework, professional responsibilities, and this thesis.

Finally, I owe my deepest thanks to my family for their constant encouragement, patience, and understanding. In particular, I am profoundly grateful to my wife for carrying so much of the load at home and caring for our children while this thesis consumed much of my free time. Her support made this thesis possible.

Abstract

Technical debt describes design and implementation choices that accelerate delivery in the short term but increase the future cost and risk of change. Prior work has clarified the concept of technical debt and documented its impact on maintainability, risk, and delivery outcomes. However, most existing approaches are not designed to run inside time-boxed agile ceremonies, and they rarely produce a small, repeatable decision aid that teams can use during routine backlog refinement to make technical debt decisions visible and defensible alongside feature work. This thesis addresses that ceremony-time gap by designing and evaluating a lightweight rubric for prioritizing technical debt in agile backlog refinement.

In the rubric round, three technical debt items moved into the top five positions, replacing feature work that had dominated the control round. Inter-rater agreement on item rankings increased from 0.44 to 0.76. Survey responses indicated that participants found the criteria clear, the resulting ranking defensible, and the decision trace useful, and they expressed willingness to adopt the method in future refinement.

The study provides insight into how a ceremony-fit rubric can help agile teams make technical debt more visible, improve agreement about priorities, and leave a brief, auditable decision trail that better supports communication with stakeholders.

I. Introduction

Technical debt (TD) describes design or implementation choices that speed delivery now but create a technical context in which future changes become more costly or risky. TD spans code, tests, documentation, and architecture, and is commonly framed in terms of principal (remediation cost) and interest (added cost/risk while deferred) [1], [2]. In industry, planning and management pressures, tight schedules, shifting priorities, and competitive dynamics frequently drive TD creation, and large practitioner studies report uneven, ad-hoc management despite high perceived impact on outcomes [3].

Despite extensive work on TD at multiple levels, prioritization remains fragmented. A systematic review of 44 studies reports limited consensus and thin empirical validation of methods/tools, with few approaches described in a way that teams could apply directly inside time-boxed agile ceremonies [4]. Industrial evidence shows that bringing a business perspective to TD decisions can improve alignment, but approaches often assume tools, setup, or time budgets teams do not have during refinement [5]. The recurring gap is the absence of a repeatable, ceremony-fit decision aid that yields a brief, auditable decision trace teams can defend to stakeholders.

This thesis addresses that ceremony-time problem. We propose a lightweight, ceremony-fit rubric for TD prioritization designed to run in ~15-30 minutes during backlog refinement:

- **Ceremony-fit:** Runnable inside standard Scrum/Kanban rituals without special tools or long training, producing a one-page decision trace (scores, rank, one-line rationale).
- **Rubric:** A 1-5 scale for five criteria where each point has a behaviorally specific descriptor to calibrate rater interpretation and improve repeatability (e.g., *Risk=5: likely severe incident if deferred*).

The rubric operationalizes five decision dimensions widely surfaced in research and practice: Risk/Safety impact**,** Architectural/Maintainability impact**,** Cost of delay / interest growth**,** Effort band (S/M/L aligned to the team's scale), and Customer/Operational impact. Consolidating factors reported in the SLR and practitioner studies into a single ceremony-ready instrument [4], [3], [1], [2]. A simple composite (equal weights by default with interest as a tie-break) produces a rank-ordered TD list with brief rationales suitable for stakeholder review.

TD is often cross-cutting and partially invisible, with interest growth over time and story-by-story scoring underweights systemic risk and long-run cost. Prior work explicitly cautions against treating TD like ordinary feature tickets and urges explicit visibility and valuation mechanisms [1], [2].

The scope of this thesis will focus on team-level agile (Scrum/Kanban) and backlog-sized TD items (code/tests/docs/sliceable architecture) that can be scheduled within or across 1 sprint and not multi-quarter platform initiatives.

We empirically evaluate the rubric against status-quo practice in a single-sprint pilot with real teams to answer the following research questions:

- **RQ1: Do participants report higher clarity, defensibility, and alignment of outcomes when using the rubric?**

  We will answer this question by conducting a short post-session Likert survey (clarity, defensibility, alignment, adoption likelihood) and summarize means, internal consistency, and include one-line rationales from the decision trace.

- **RQ2: Can teams apply the rubric within ceremony time constraints without unacceptable overhead?**

  We will answer this question by recording the time per item/round and include a survey item on time acceptability.

Our contributions are:

1. A minimal**,** ceremony-fit TD prioritization rubric and a one-page decision trace.

2. An industrial-context pilot comparing agreement, perceived decision quality, and time burden with vs. without the rubric

II. Background

2.1 Technical Debt: Definition and Properties

Technical debt (TD) in this thesis refers to shortfalls in internal quality across code, tests, documentation, and architecture that increase the future cost or risk of change. The metaphor was first introduced by Cunningham in the early 1990s to describe how taking design shortcuts is like taking on financial debt, with ongoing *interest* in the form of extra effort until the code is refactored [6]. The literature commonly characterizes TD using the economic notions of *principal*, which is the effort required to remediate, and interest, the incremental cost or risk that accrues while remediation is deferred; visibility and accretion are treated as important properties that influence how TD is noticed and grows over time [1], [2]. This definition intentionally spans operational and architectural concerns because industrial studies report that teams experience TD both as near-term friction (e.g., incidents, support tickets) and as longer-horizon maintainability constraints [3].

2.2 Agile Ceremonies

In most agile teams, decisions about what work to do next are concentrated in a small set of recurring ceremonies. Backlog refinement and sprint planning are typically time-boxed meetings where engineers, tech leads, and product owners clarify requirements, estimate effort, and negotiate the scope and ordering of work for upcoming sprints. The primary focus in these sessions is to align delivery commitments with product goals and capacity.

Within this setting, technical debt is often introduced as just another backlog item that competes with features and bugs. This framing can be misleading because many TD items are cross-cutting or partly invisible, and because interest growth means the cost of deferral may rise in ways that ordinary feature triage underestimates [1], [2]. Practitioner evidence reinforces this distinction: teams frequently report architectural and operational pain from sustained deferral and inconsistent prioritization, even when everyone acknowledges that debt exists [3]. These ceremonies are therefore the practical decision surface for TD, but they rarely provide explicit structure for reasoning about risk, maintainability, and cost of delay alongside feature work.

2.3 Decision Framework and Rubric Terminology

This thesis uses the term decision framework in its standard sense: a structured, multi-criteria procedure that helps a group evaluate options and produce an auditable outcome (e.g., a rank with brief rationale). Specifically, we employ a defined rubric, meaning each criterion is scored on a small ordinal scale where every scale point is described behaviorally to calibrate rater interpretation and support repeatability during live ceremonies. The rubric's output is a concise decision trace: criterion scores, an overall rank, and a one-line rationale for each selected item, designed to be readable by both engineering and product stakeholders.

III. Related Work

3.1 How technical debt is treated today and why that is inadequate

Technical debt has moved from a metaphor to an operational concept. Core papers frame debt in terms of principal and interest and highlight properties like visibility and accretion that influence how debt is noticed and how it grows over time. The guidance is to make debt visible so it can be considered in system level decisions and to reason about how deferral adds cost and risk over time [2] [1] [12]. In practice, large practitioner studies report that architectural decisions are a primary source of debt and that existing tools rarely help with day to day prioritization. Teams feel the pain but management awareness and usable support are uneven. Planning pressure and shifting priorities are frequent drivers. Effects include delivery delays, lower maintainability, and operational friction [3]. This body of work explains what TD is and how it shows up, but it does

not give teams a small repeatable way to prioritize during ceremony time. Teams often revert to local heuristics or ad hoc debates that do not leave a clear trace.

3.2 What has been proposed to aid TD management

Prior work falls into two main streams for prioritization. First, factor catalogs and systematic reviews collect strategies, processes, factors, and tools. The JSS review of 44 studies concludes that the area is preliminary and lacks validated tools. It reports little consensus on which factors matter most or how to operationalize them inside regular development work [4]. Second, business driven approaches link debt items to IT assets and value sources and report better alignment when decisions are framed in business terms. In one multi month deployment, the Tracy and TDM approach showed a downward trend in high priority debt and surfaced business factors that shaped decisions [5]. These cases show that structure can move teams beyond unstructured debate, but they often assume extra tooling and facilitation that are not easy to fit into a 15 to 30 minute refinement window [5].

3.3 Remaining gaps and positioning of this thesis

Taken together, the field explains what to consider and shows that business framing can help. In day-to-day practice, agile teams already use simple decision rules or matrices for user stories to decide what fits into a sprint during backlog refinement. What remains inadequate is an equally practical decision rule for technical debt that fits inside the same ceremony and consistently produces a rank and a one-page decision trace that engineers and product partners can defend after the meeting. Reviews are explicit that validated tools and consensus are scarce [4]. Many industrial frameworks assume canvases and support systems that are hard to run inside routine ceremonies [5]. Reporting on inter-rater agreement and on the transparency of the decision trail is uncommon. This thesis targets that ceremony-time gap. The next section describes the rubric design and the pilot used to evaluate clarity, defensibility, agreement, and time burden.

3.4 Derivation of the five criteria used later

The five criteria consolidate dimensions that recur across reviews and practitioner reports rather than coming from a single source. Risk and safety impact reflects emphasis on architectural risk and incident likelihood in surveys and architecture-focused discussions [3] [2]. Architectural and maintainability impact sits at the center of TD definitions and appears throughout practitioner evidence [2] [1] [3]. Cost of delay and interest growth follows from the economic framing and the factor maps reported in the review [2] [4]. Effort band represents principal and is present in nearly every prioritization account [4] [1]. Customer and operational impact is supported by studies that link debt to delays, bugs, and operational friction [4] [3].

No individual work proposes exactly this five-factor set, but across these studies there is consistent overlap in the families of concerns they highlight: risk, maintainability, time/value impact, effort,

and user or operational consequences. The rubric later in this thesis makes these recurring families explicit by mapping them onto short behavioral scales that can be applied consistently during a live meeting.

IV. Rubric Design

4.1 Design of the Rubric Artifact

The rubric was designed as a small, ceremony-fit decision aid that could be applied in 15–30 minutes during backlog refinement without additional tooling or lengthy training. The choice of criteria reflects dimensions repeatedly surfaced in prior research and practitioner reports: Risk/Safety impact, Architectural/Maintainability impact, Cost of delay/interest growth, Effort (principal), and Customer/Operational impact [4], [3], [1], [2]. As discussed in Related Work, these dimensions do not come from a single source; rather, they consolidate recurring "families" of concerns across systematic reviews, industrial case studies, and survey results.

The design process followed three filters. First, candidate dimensions had to appear with some consistency across the literature (e.g., economic framing of principal and interest, risk and maintainability concerns, customer-facing impact). Second, they had to match how practitioners in the target organization already talked about technical debt in interviews and informal discussions (e.g., "risk to patients," "how hard this is to change," "how much it slows us down"). Third, each dimension had to be rateable quickly in a live meeting using information the team already had at hand, without requiring specialized measurement tools.

Given these constraints, the rubric uses five criteria, each scored on a 1–5 ordinal scale with behaviorally specific descriptors at key points. Scores are combined into a simple composite using equal weights by default, with Cost of Delay/Interest used as a tie-breaker to keep time effects visible. The intention is not to claim a theoretically optimal weighting scheme, but to provide a transparent, repeatable decision rule that teams can apply within existing refinement timeboxes and explain to stakeholders after the meeting.

4.2 Rubric Descriptors

Each backlog item is rated independently by participants on the five criteria below using a 1–5 scale. The descriptors at levels 1, 3, and 5 anchor the scale; values 2 and 4 are used when an item clearly falls between those anchors.

4.2.1 Risk / Safety impact

- 1 – No user-visible risk; issues are purely internal and unlikely to affect users or safety in the current release.
- 3 – Possible moderate incidents or user-facing errors if the item is deferred (e.g., incorrect data in non-critical views, recoverable workflow issues).
- 5 – Likely severe incidents or safety-relevant failures if the item continues to be deferred.

4.2.2 Architectural / Maintainability impact

- 1 – Local or isolated impact; changes are confined to a small, well-bounded area of the codebase.
- 3 – Cross-module complexity; the item touches multiple components and makes evolution noticeably harder.
- 5 – Systemic hotspot that blocks or significantly complicates broader change, refactoring, or evolution.

4.2.3 Cost of Delay / Interest growth

- 1 – Effect of deferral is essentially flat; delaying the work is not expected to meaningfully increase future effort or risk.
- 3 – Approximately linear growth in cost or risk as the item is deferred (e.g., incremental rework or friction over time).
- 5 – Accelerating rework or risk if deferred; cost grows faster than linearly as more features and dependencies accumulate around the debt.

4.2.4 Effort (principal)

- 1 – Small (S): roughly 0–0.5 day of work for one engineer.
- 3 – Medium (M): roughly 2–3 days of work.
- 5 – Large (L): roughly 3–5 weeks of work, spanning multiple sprints or significant coordination.

4.2.5 Customer / Operational impact

- 1 – Internal only; effects are limited to engineering workflows or internal tooling, with no direct user or operational impact.
- 3 – Occasional user friction or operational inconvenience (e.g., extra manual steps, sporadic confusion).
- 5 – Frequent user-facing problems or operational tickets at risk (e.g., recurring support cases, regular workarounds required).

These behaviorally anchored descriptors are intended to reduce interpretation drift across participants and sessions. They give teams a shared vocabulary for scoring items quickly while still leaving room for context-specific judgment in the middle of the scale.

V. Methods

5.1 Research Design & Justification

We evaluate a lightweight rubric for technical debt prioritization using a within-team comparison during normal backlog refinement. Each team completes two short rounds on the same set of backlog items, one with the team's typical ad hoc approach and one with the rubric. This design isolates the contribution of the rubric inside real Scrum or Kanban constraints where time is tight and multiple stakeholders are present. The warrant is that if the rubric reduces ambiguity in how participants interpret risk, maintainability, and cost of delay, then inter rater agreement should increase and the decision trace should be clearer without exceeding ceremony time. The backing is the design science view that artifacts should be evaluated for utility and quality in their environment and the empirical software engineering guidance to use clear instruments, repeatable procedures, and transparent reporting [7] [8] [9] [10] [11]. The primary outcome is inter rater agreement on item ranks, and secondary outcomes are perceived clarity, defensibility, alignment, and time burden.

5.2 Setting and Participants / Systems

This study was conducted as a single-team pilot with a product team at a large biotechnology company. The team owns an internal planning tool for clinical trials, implemented as a greenfield web application built with Angular, TypeScript, Next.js, Python, and Postgres and deployed on AWS. The team follows Scrum with two-week sprints, and backlog refinement normally occupies a one-hour timebox.

Five practitioners participated in the session: a lead engineer, three senior engineers, and a project manager. All had at least three years of experience and had been involved in backlog refinement for this product. This composition reflects the people who actually make prioritization decisions for the system. The implications of relying on a single-team, single-organization pilot are discussed later in the threats to validity.

5.3 Study Scope and Boundaries

Because refinement and planning are tightly time-boxed, the framework artifact must satisfy explicit design constraints. In this thesis, design constraints refer to the properties of the rubric and its use, not of agile methods broadly: the procedure should run in roughly 15–30 minutes without special tooling or lengthy training; it should impose low cognitive overhead; its scales should support consistent interpretation among participants; and it should emit a one-page decision trace that is suitable for later review. Signals from CI/CD, static analysis, incidents, or support systems may inform ratings, but the rubric supplies the decision rule that translates heterogeneous evidence into a ceremony-ready outcome.

Finally, the scope of the study is limited to team-level agile (Scrum/Kanban) and backlog-sized TD items including code, tests, documentation, and sliceable architectural work that can be scheduled within or across a small number of sprints. Multi-quarter platform initiatives and organization-wide transformations are out of scope. This scoping keeps the artifact focused on the realities of everyday planning while enabling an empirical evaluation consistent with established guidance in design-science and empirical software engineering (discussed in Methods) [7], [8], [9], [10], [11].

5.4 Work artifacts and item sets

Each session uses ten real backlog items drawn from the team's tracker. Five items are technical debt and five items are non-technical work such as features or bugs. Including both types prevents participants from placing all attention on a single class of work and allows a check that the rubric does not simply favor a category. Items are selected with the product owner and tech lead before the session and are presented as one-liners with one to two sentences of neutral context. No estimates are provided in advance to avoid anchoring. Technical debt items are in scope at the team level and are schedulable within one or two sprints, which matches the scope defined in the thesis introduction.

5.5 Data Sources

In the rubric round, each participant independently rates each item on five criteria using 1 to 5 scales. The criteria are Risk and safety impact, Architectural and maintainability impact, Cost of delay or interest growth, Effort band which represents principal, and Customer or operational impact. These families are derived from the literature and practitioner evidence [4] [3] [2] [1].

A composite score uses equal weights by default and ties break in favor of higher cost of delay to keep time effects explicit. Group ranks are then produced from the composites. In the control round the team performs its typical ad hoc prioritization and records a group rank. We then record time per item and total time per round to check ceremony fit. For the top N items in each round we capture a one-line rationale in the team's own words.

After both rounds, participants complete a short survey with ten Likert items and one open-ended question. Constructs are clarity of criteria, defensibility of the resulting rank, perceived alignment, perceived fairness of participation, time acceptability, usefulness of the output artifact, and adoption likelihood. Items are phrased in plain language, reviewed for face validity with two independent engineers, and grouped to allow a simple reliability check. Survey development follows empirical reporting guidance that recommends clear constructs, simple wording, and internal consistency checks [9] [10].

When available we collect the last two refinement decisions that involved technical debt from team records. We record the chosen items and, if present, short rationales or meeting notes. This provides a descriptive baseline for trace quality and choice defense.

5.6 Procedures

Preparation in under 30 minutes. A coordinator meets with the tech lead and product owner to select ten items and to confirm scope. We load the items into a simple spreadsheet that computes composites and ranks and that contains the 1 to 5 descriptors for the rubric. We print or share rating sheets and the post-session survey.

Session flow in 45 to 60 minutes total.

1. Briefing is conducted in three minutes where we explain the purpose, the two rounds, and the data we will collect.
2. Round A is completed in 15 to 20 minutes where each item is briefly presented and the team performs its typical ad hoc prioritization to produce a group rank. We record time per item and total time.
3. Two minute calibration where the 1 to 5 descriptors for each criterion together are read. Take one practice rating on a warm-up item not in the set to align interpretations.
4. Round B in 20 to 25 minutes where each participant independently rates each item on the five criteria. Ratings are entered and composite scores and the group rank are produced. For the top N items we record a one-line rationale. We record time per item and total time.
5. The survey is conducted in 5 to 8 minutes where we administer the ten item Likert survey and the open-ended question.

6. Share the ranks and rationales from both rounds with the team so they can reuse any helpful artifacts.-8 min): Administer the 10-item Likert + open-ended questionnaire.

5.7 Post-Session Survey

Constructs reflect what the literature and practitioners highlight as important outcomes of a usable decision aid. Clarity and repeatability, defensibility to stakeholders, perceived alignment, fairness of participation, time acceptability, usefulness of the produced artifact, and adoption likelihood [4] [9] [10] [3]. Items were written in plain language and reviewed by two engineers for clarity.

The survey uses a five point scale from strongly disagree to strongly agree unless noted.

1. The criteria were clear to apply.
2. I can justify the resulting rank to stakeholders.
3. How long did the process take overall? (not likert)
4. The output artifact (scores/rationale) is helpful.
5. The method improved team alignment.
6. Everyone's ranking was considered equally, without favoring any particular participant.
7. The method made tech debt risk and prioritization more explicit.
8. The method produced better decisions than our prior approach.
9. I would adopt this method in future refinement.
10. Open-ended: What was most/least useful, and why?

5.8 Historical baseline procedure

For context, we also examine a small historical baseline of recent refinement decisions that involved technical debt. Specifically, we identify two recent refinement sessions in which at least one TD item was considered, and record (i) which items were ultimately selected, (ii) any documented rationale, if present, and (iii) any incidents or support tickets related to those items in the following sprint. These observations are used only as descriptive background to characterize trace quality and are not included in the main quantitative analysis.

VI. Results and Analysis

6.1 Session and participant overview

The pilot was run with one product team at a large biotech company that owns an internal planning tool for clinical trials. The system is a greenfield web application built with Angular, TypeScript, Next.js, Python, and Postgres deployed on AWS. The team follows Scrum with two-week sprints, and backlog refinement normally occupies a one-hour timebox.

Five practitioners participated in the study: a lead engineer, three senior engineers, and a project manager. All had at least three years of experience and had been involved in refinement for this product. The session followed the planned structure. The control round using the team's usual ad hoc method took about 20 minutes. The rubric round took about 24 minutes. The remaining time covered briefing, a short calibration step, and the post-session survey.

The item set for the session consisted of ten real backlog tickets taken from an upcoming refinement. Five tickets were technical debt items that affected rules logic, test coverage, and architecture, for example, refactoring the patient-matching rules engine to remove duplicated eligibility logic and make safety checks easier to change. The other five were non-technical debt items (four features and one bug) that impacted user-visible dashboards and alerts, such as adding a PI dashboard for enrollment risk and fixing the double-counting of withdrawn patients in an existing enrollment dashboard.

6.2 Ticket ranking outcomes

Table 1 below summarizes how ticket ranks changed between the control and rubric rounds. In the control round, all top five items were non-technical debt work. The bug that double-counted withdrawn patients (Ticket 10) ranked first. Three feature requests for dashboards, filters, and alerts (Tickets 6, 8, and 9) and one additional feature (Ticket 7) filled out the remaining top positions. None of the five technical debt items appeared in the top five.

Table 1: Control vs Rubric Round Ticket Comparison

Ticket #	**Ticket Type**	**Control Rank** (Adhoc)	**Rubric Group Rank**	**Rank Change** (Rubric - Control)
1	Tech Debt	6	1	-5
2	Tech Debt	9	3	-6
3	Tech Debt	7	9	2
4	Tech Debt	8	5	-3
5	Tech Debt	10	6	-4
6	Non-Tech (Feature)	2	4	2
7	Non-Tech (Feature)	5	8	3
8	Non-Tech (Feature)	3	7	4
9	Non-Tech (Feature)	4	10	6
	Non-Tech (Bug)			

When the team applied the rubric, three technical debt items moved into the top five. The refactor of the patient-matching rules engine (Ticket 1) moved from rank 6 to rank 1. The fix for double-counting withdrawn patients (Ticket 10) remained near the top at rank 2. Splitting the monolithic enrollment forecast job (Ticket 2) moved from rank 9 to rank 3. Adding tests to the screening eligibility engine (Ticket 4) moved from rank 8 to rank 5. Only one feature, the PI dashboard (Ticket 6), remained in the top five under the rubric.

Rank changes were largest for two technical debt items. Ticket 1 improved by five positions. Ticket 2 improved by six positions. The decoupling of the UI from API schema quirks (Ticket 5) also moved up, from rank 10 to rank 6. Several feature items moved down in relative priority. The enrollment timeline visualization (Ticket 9) dropped from rank 4 to rank 10. Self-service filters (Ticket 7) and email alerts (Ticket 8) each shifted from the middle of the control ranking to the lower half under the rubric.

The overlap between the two top-five lists was limited. Only two items, the bug fix (Ticket 10) and the PI dashboard (Ticket 6), appeared in the top five in both conditions. In the control round the top five consisted of zero technical debt items and five non-technical items. In the rubric round the top five contained three technical debt items and two non-technical items.

6.3 Agreement and timing

Table 2 below shows that the rubric increased inter-rater agreement on item rankings. Using Kendall's W on participant rankings, agreement in the control condition was 0.44. With the rubric, Kendall's W increased to 0.76. This aligns with the intended effect of providing criteria to reduce ambiguity among participants.

Time measurements show that the rubric introduced modest overhead while remaining inside the existing ceremony timebox. The control round averaged about 2.0 minutes per item. The rubric round averaged about 2.4 minutes per item. The difference was roughly four minutes for the set of ten items. Both rounds plus briefing, calibration, and survey fit within the team's normal one-hour refinement window.

Table 2: Round-Level Summary - Control vs Rubric

Metric	**Control** (Adhoc)	**Rubric Condition**
Total Round Time (min)	20	24
Average Time per Item (min)	2.0	2.4
Agreement Measure (Kendall's W)	0.44	0.76
# Items in Top N (TD)		
# Items in Top N (Non-TD)		
Top-N Overlap (%)	40% (2 of 5)	–

6.4 Survey responses and qualitative notes

Table 3 below shows the results of the likert scale survey responses. On a five point scale, participants reported that the rubric made criteria clear to apply (Q1 mean 4.4) and that they could justify the resulting rank to stakeholders (Q2 mean 3.6). Perceived alignment and fairness were also high.

Perceptions of outcomes and adoption were positive. Participants agreed that the method made technical debt risk and prioritization more explicit (Q7 mean 4.6). Perceived fairness was also high (Q6 mean 4.6), and the output artifact of scores and rationales was viewed positively overall, though somewhat more mixed (Q4 mean 3.4). They rated the method as producing better decisions than their prior approach (Q8 mean 3.6) and indicated willingness to adopt it in future refinement (Q9 mean 3.8).

Table 3: Survey Questions Results

Participant ID	Q1	Q2	Q4	Q5	Q6	Q7	Q8	Q9
P1	5	3	3	4	5	5	3	3
P2	4	4	4	4	4	4	4	4
P3	4	4	4	5	4	4	4	4
P4	5	3	2	4	5	5	3	3
P5	4	4	4	5	5	5	4	5
Mean	4.4	3.6	3.4	4.4	4.6	4.6	3.6	3.8

Table 14 in the appendix Self-reported time estimates from the survey were close to observed timing, with most participants estimating about 55 minutes for the full session. Open-ended responses noted that the most useful aspects were the explicit focus on risk and maintainability and the decision trace that could be shared with leadership. Less useful aspects included some perceived overlap between criteria and the learning curve on the scoring grid at the start of the session.

Table 15 in the appendix shows the historical records for two previous refinement decisions showed that technical debt items were often discussed but rarely selected when competing with visible feature work. Documented rationales tended to emphasize stakeholder requests rather than structured risk or maintainability arguments. These notes provide descriptive context for the observed shift toward technical debt items in the rubric round.

VII. Discussion

7.1 Interpretation of results

The pilot results offer early support for the central claim that a lightweight rubric can improve the quality and transparency of technical debt prioritization within ceremony time.

First, the ranking outcomes show that the rubric changed which items rose to the top when technical risk and maintainability were considered explicitly. In the control round the team's highest-ranked work consisted entirely of features and a visible bug. Under the rubric, three technical debt items that affect patient-matching logic, forecasting infrastructure, and test coverage moved into the top five. These shifts are consistent with the intended effect of making cross-cutting and partially invisible debt more comparable to features during refinement rather than treating it as generic backlog work.

Second, the increase in Kendall's W from 0.44 to 0.76 indicates higher agreement among participants when they scored items using the criteria. This suggests that the rubric reduced ambiguity in how participants interpreted risk, maintainability, and cost of delay. The higher agreement aligns with the design goal of producing a more repeatable and defensible decision trace instead of relying on ad hoc debate.

Third, the survey results indicate that the method was acceptable inside a real refinement setting. Participants reported that the criteria were clear, the resulting ranking was defensible, and the time burden was appropriate for a one-hour ceremony. They also expressed willingness to adopt the method in future sessions. These perceptions matter because a decision aid that is theoretically sound but rejected by teams would not meet the practical goals of this thesis.

Finally, the modest increase of about four minutes over ten items suggests that the rubric can fit into existing refinement practices without requiring a separate workshop. The time trade-off appears to be a small extension in exchange for a more transparent and risk-aware outcome.

7.2 Rebuttals and counterarguments

Several alternative interpretations are plausible. One counterargument is that the observed shifts in ranking could be driven by the novelty of the method rather than by real improvements in decision quality. Participants may have given more weight to technical debt items simply because the study emphasized them. The study design reduces this risk by using real backlog items and by anchoring ratings in concrete descriptors, but it cannot fully separate novelty effects from genuine change in practice.

A second criticism is that higher inter-rater agreement does not guarantee better decisions. Participants could converge on a poor decision if the rubric is mis-calibrated or if the criteria overemphasize certain aspects of technical debt. The current pilot only shows that agreement and

perceived defensibility improved, not that long-term outcomes such as fewer incidents or lower maintenance cost improved.

A third concern is that the increased priority of technical debt might crowd out important features that also support clinical goals. In this pilot, some feature work moved down the ranking when compared against high-risk technical debt. This may be appropriate for this team and product but could be problematic in contexts where feature delivery is more time-critical than debt remediation.

These concerns highlight the need to treat the pilot results as indicative rather than conclusive. The study does not claim to prove that the rubric always leads to superior business outcomes. It shows that, in one real team, the rubric shifted attention toward debt items that participants judged to be higher risk and did so in a way that they found transparent and acceptable.

7.3 Qualifiers and threats to validity

The scope of the current claim is limited. These results apply to small agile teams working on internal web applications in a regulated domain, using Scrum or Kanban and time-boxed refinement sessions of about one hour. The artifact is tuned for backlog-sized technical debt items that can fit within one or two sprints. Larger platform initiatives and multi-team programs are out of scope.

Internal validity is threatened by the small number of teams and the involvement of the researcher in facilitating the session. The presence of an observer and a new framework may have influenced behavior. Counter-balancing the order of control and rubric rounds across teams in future replications would help separate order effects and learning effects from the impact of the rubric itself.

External validity is limited by the single-organization, single-team context. The biotech company operates in a clinical trials domain with strong safety expectations and data integrity requirements. Teams in consumer products or infrastructure may weigh criteria differently. The technology stack is modern and greenfield, which may differ from heavily legacy systems where certain types of debt dominate.

Construct validity depends on whether the chosen criteria and survey items actually capture clarity, defensibility, and alignment in the way practitioners experience them. The criteria were derived from literature and practitioner reports, and survey items were reviewed for face validity, but misalignment is still possible. For example, some participants noted overlap between cost of delay and customer impact. Future iterations may refine the criteria labels to reduce redundancy.

Reliability is affected by the small sample size and the single pilot session. Although scales and independent ratings improve consistency, the stability of results across sessions, teams, and products is not yet known. Repeating the protocol with additional teams would help assess how robust the observed patterns are.

7.4 Impact and implications

Despite these limits, the pilot suggests several practical implications for industrial teams. A small, explicit rubric can help teams make technical debt visible alongside features in routine refinement, without requiring significant new tooling or separate workshops. The decision trace of scores, ranks, and one-line rationales gives product managers and engineering leaders a concrete artifact to explain why certain debt items were chosen over others and why some feature work was deferred.

For organizations, adopting a ceremony-fit debt rubric could support more consistent communication about risk and maintainability, especially in domains where errors carry safety or regulatory consequences. Even if different teams adjust the criteria or weightings, the practice of scoring debt items in a structured way may reduce the influence of individual advocacy and help align engineering and product perspectives.

From a research perspective, this pilot adds an empirical example of a design-science artifact evaluated in its natural setting. It connects prior work on technical debt concepts and business-driven frameworks with a practical procedure that fits inside an existing agile ceremony. The improvement in agreement and the shift in top-ranked items provide initial data points for how such frameworks may change decisions in practice.

Future work should extend the study to additional teams and domains, compare alternative rubric designs, and link rubric use to downstream outcomes such as incident rates, rework effort, and delivery throughput. Instrumentation of tools and historical analysis of technical debt decisions could complement survey and session data. These steps would help determine when and where a lightweight decision framework has the greatest impact on technical debt management.

7.5 Conclusion

This thesis addressed a recurring gap in the technical debt literature and practice: the lack of a small, repeatable decision aid that agile teams can use during routine backlog refinement to make technical debt decisions visible and defensible alongside feature work. Building on prior work on principal and interest, visibility, and business framing of technical debt, it designed a ceremony-fit rubric that operationalizes five recurring decision dimensions – risk and safety impact, architectural and maintainability impact, cost of delay and interest growth, effort band, and customer or operational impact – into a 1–5 behaviorally anchored scale and a one-page decision trace. A single-team pilot in a clinical trials planning context compared the rubric against the team's existing ad hoc approach within a normal refinement session. The results showed that the rubric shifted the top of the backlog toward higher-risk technical debt items, increased inter-rater agreement on item rankings from 0.44 to 0.76, and produced an outcome that participants judged to be clearer, more defensible, and still compatible with ceremony time constraints. These findings are preliminary and restricted to one organization and team, but they provide early evidence that a

lightweight, ceremony-fit rubric can help teams surface cross-cutting debt, align on priorities, and leave a concise, auditable decision trail that supports communication with stakeholders.

7.6 Future Work

Several extensions follow naturally from this pilot. First, the study should be replicated with additional teams, products, and domains, including legacy systems and less regulated environments, to assess how robust the observed patterns are and how teams in different contexts interpret and weight the criteria. Replications could also counter-balance the order of control and rubric rounds to separate learning and novelty effects from the impact of the rubric itself.

Second, future work could explore variations of the rubric design. This includes experimenting with alternative weightings, adjusted or merged criteria where overlap was perceived, and domain-specific descriptors tuned for consumer products, infrastructure, or data platforms. Comparative studies of different rubric variants could examine how changes in scales and weights influence agreement, perceived fairness, and the balance between technical debt and feature work.

Third, a longer-term view is needed to connect rubric use to downstream outcomes. Longitudinal studies could track incident rates, rework effort, lead time, and defect trends before and after adoption of a rubric-based refinement practice. Instrumentation of issue trackers and CI/CD pipelines, combined with qualitative interviews, could provide a richer picture of how structured decision traces influence stakeholder trust, governance, and investment in internal quality over time. Finally, adapting the core ideas to higher levels of planning, such as portfolio or program roadmaps, may help organizations reason about larger technical debt initiatives while preserving the ceremony-fit principles that guided this initial design.

VIII. References

[1] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.

[2] N. Brown et al., "Managing technical debt in software-reliant systems," in *Proc. FSE/SDP Workshop on Future of Software Engineering Research (FoSER)*, 2010, pp. 47–52.

[3] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? Software practitioners and technical debt," in *Proc. 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2015, pp. 50–60.

[4] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F. A. Fontana, "A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools," *Journal of Systems and Software*, vol. 171, p. 110827, Jan. 2021.

[5] R. R. de Almeida, R. N. Ribeiro, C. Treude, and U. Kulesza, "Business-driven technical debt prioritization: An industrial case study," in *Proc. IEEE/ACM Int. Workshop on Managing Technical Debt (TechDebt)*, 2021, pp. 41–48.

[6] W. Cunningham, "The WyCash portfolio management system," in Addendum to the Proceedings OOPSLA '92, 1992.

[7] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[8] A. R. Hevner, "A three cycle view of design science research," *Scandinavian Journal of Information Systems*, vol. 19, no. 2, pp. 87–92, 2007.

[9] B. Kitchenham, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[10] B. Kitchenham et al., "Evaluating guidelines for reporting empirical software engineering studies," *Empirical Software Engineering*, vol. 13, no. 1, pp. 97–121, 2008.

[11] B. Kitchenham and S. Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, EBSE Technical Report EBSE-2007-01, Keele University and Durham University, 2007.

[12] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing technical debt in software engineering (Dagstuhl Seminar 16162)," *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016.

IX. Appendix

Table 1: Session Metadata – Team 1

Field	Value
Team ID	Team 1
Agile Method	Scrum, 2-week sprints, Session carried out during backlog refinement meeting
Number of Participants	5
Legacy vs Greenfield Category	Greenfield
Participants IDs and their Roles	P1: Lead Engineer P2: Senior Engineer P3: Senior Engineer P4: Senior Engineer P5: Project Manager

Table 2: List of Backlog Tickets

Ticket Number	One Line Summary	Description *(As one to two sentences of neutral context)*	Ticket Type
1	Refactor patient-matching rules engine	The current patient-matching rules engine for eligibility checks is a single, hard-to-modify module with duplicated logic across trial types, making safety validations difficult to change safely.	Tech Debt
2	Split monolithic enrollment forecast job	The nightly enrollment forecast job is a monolithic Python script with intertwined data pulls and calculations, slowing iteration on new forecasting models.	Tech Debt

3	Remove hard-coded trial arm parameters in UI	Several trial arm thresholds and caps are hard-coded in the Angular UI, requiring code deployments for changes that should be configuration-driven.	Tech Debt
4	Add tests for screening eligibility engine	The screening eligibility engine has minimal automated tests, increasing the risk of regressions when new criteria or countries are added.	Tech Debt
5	Decouple Angular UI from API schema quirks	The Angular front-end assumes specific API response shapes and field names, making it brittle when the Next.js API or Postgres schema evolves.	Tech Debt
6	Add PI dashboard for enrollment risk	Provide principal investigators with a dashboard visualizing projected vs. actual enrollment and high-risk cohorts to support proactive site management.	Non-Tech Debt (Feature)
7	Self-service filters for site and country constraints	Allow users to filter enrollment views by site, country, and key inclusion/exclusion criteria without needing engineering changes.	Non-Tech Debt (Feature)
8	Email alerts for slow-enrolling sites	Automatically send email notifications to study leads when a site's enrollment drops below configured thresholds for a sustained period.	Non-Tech Debt (Feature)

9	Timeline visualization for planned vs. actual enrollment	Add a timeline chart showing planned vs. actual enrollment over time to help identify schedule risk early.	Non-Tech Debt (Feature)
10	Fix double-counting of withdrawn patients in dashboard	The current dashboard occasionally double-counts withdrawn patients when aggregating across amendments, over-stating enrollment progress for some trials.	Non-Tech Debt (Bug)

Table 3: Control Round Tickets Review

Ticket Number	Group Rank	Time for this item	One line rationale
1	6	1.8	Rules engine is important but not as visible as the dashboard and bug fix right now.
2	9	1.5	Forecast job “works well enough” today; seen more as cleanup work.
3	7	1.5	Hard-coded parameters are annoying but not causing user-visible issues.
4	8	1.7	Missing tests are a risk but seen as “engineering hygiene” vs. immediate delivery.
5	10	1.4	Coupling pain is acknowledged but not felt as urgent compared to other items.
6	2	2.2	PI dashboard is high impact and frequently requested by stakeholders.
7	5	1.9	Filters are helpful for power users but not as urgent as the bug or dashboard.
		1.9	Email alerts are seen as a quick win to reduce manual monitoring
		1.8	Timeline view is valuable for study leads to track schedule risk.

10	1	2.3	Double-counting withdrawn patients can mislead decision-making; considered the most urgent to fix.
Total Control Round Time:	~20 minutes		

Table 4: Pilot Round Ticket #1 Review

Ticket #1					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	5	5	5	3	4
P2	5	5	4	3	4
P3	5	4	5	3	3
P4	4	5	5	3	4
P5	5	5	5	2	4
Time Recorded:	3.0 minutes				

Table 5: Pilot Round Ticket #2 Review

Ticket #2					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	4	5	4	4	4
P2	4	5	4	4	3
P3	3	5	4	4	4
P4	4	4	4	5	4
P5	4	5	5	4	4
Time Recorded:	2.5 minutes				

Table 6: Pilot Round Ticket #3 Review

Ticket #3

Participant ID	Risk / Safety	Maintainability	Cost of Delay	Effort Band	Customer impact
P1	2	3	3	3	2
P2	2	3	3	3	2
P3	2	2	3	3	2
P4	1	3	3	3	2
P5	2	3	2	3	2
Time Recorded:	2.0 minutes				

Table 7: Pilot Round Ticket #4 Review

Ticket #4					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	4	4	3	3	3
P2	4	4	3	3	3
P3	3	4	3	3	3
P4	4	3	3	3	3
P5	4	4	4	3	4
Time Recorded:	2.0 minutes				

Table 8: Pilot Round Ticket #5 Review

Ticket #5					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	3	4	4	4	3
P2	3	4	4	4	2
P3	2	4	4	4	3
P4	3	4	3	4	3
P5	3	5	4	5	3
Time Recorded:	2.0 minutes				

Table 9: Pilot Round Ticket #6 Review

Ticket #6

Participant ID	Risk / Safety	Maintainability	Cost of Delay	Effort Band	Customer impact
P1	3	3	4	3	5
P2	3	3	4	3	5
P3	2	3	4	3	4
P4	3	3	5	3	5
P5	3	2	4	3	5
Time Recorded:	2.0 minutes				

Table 10: Pilot Round Ticket #7 Review

Ticket #7					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	2	3	3	3	3
P2	2	3	3	3	3
P3	2	2	3	3	3
P4	3	3	3	3	3
P5	2	3	2	3	3
Time Recorded:	2.0 minutes				

Table 11: Pilot Round Ticket #8 Review

Ticket #8					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	3	2	3	3	4
P2	3	2	3	3	4
P3	2	2	3	3	4
P4	3	2	4	3	4
P5	3	1	3	2	5
Time Recorded:	2.0 minutes				

Table 12: Pilot Round Ticket #9 Review

Ticket #9

Participant ID	Risk / Safety	Maintainability	Cost of Delay	Effort Band	Customer impact
P1	2	2	2	3	3
P2	2	2	2	3	3
P3	1	2	2	3	3
P4	2	2	2	3	2
P5	2	1	2	3	3
Time Recorded:	2.0 minutes				

Table 13: Pilot Round Ticket #10 Review

Ticket #10					
Participant ID	**Risk / Safety**	**Maintainability**	**Cost of Delay**	**Effort Band**	**Customer impact**
P1	5	3	4	2	5
P2	4	3	4	2	5
P3	4	3	5	2	5
P4	4	2	4	2	5
P5	4	3	4	1	5
Time Recorded:	3.0 minutes				

Table 14: Survey Open Ended Question Results and Self-Reported Time

Participant ID	Q10 (open ended)	Self Reported Time (min)
P1	"Most useful was having explicit criteria for risk and maintainability. Least useful was some overlap between cost of delay and user impact."	55
P2	"The rubric forced us to talk about TD more. It felt a bit slower than our usual refinement probably because we repeated the process twice with our existing method so would be interesting to see how we do this from the start."	60
P3	"I liked that everyone on the team got a chance to participate. The scoring grid was a bit dense at first but easier by the third ticket."	

P4	"Most useful was the decision trace we can show to stakeholders. Least useful was going back over items we already know well, but it clarified trade-offs and we had some good discussions."	55
P5	"It made TD visible next to features we're working on now."	55

Table 15: Team 1 Historical Records

Decision ID	**Sprint / Date**	**TD Items Considered** (IDs / short names)	**Chosen Item(s)**	**Documented Rationale** (if any)	**Follow-up Signals Next Sprint** (Incidents, tickets, etc.)
1	Sprint 12	"Legacy screen eligibility flags", "Slow nightly enrollment export", "Deprecated S3 bucket cleanup"	"Slow nightly enrollment export"	"Export is painful for ops and slows weekly reporting; others can wait."	1 support ticket about eligibility confusion; no export incidents.
2	Sprint 13	"Legacy screen eligibility flags", "Hard-coded trial arm parameters in UI"	"Hard-coded trial arm parameters in UI"	"PM requested more flexible configuration for country-specific trial arms."	One minor bug filed about inconsistent thresholds; fixed in same sprint.

Scoring Trust: Evaluating Mobile Biometric Authentication With Standardized Security and Usability Metrics

Submitted in partial fulfillment of the requirements for the

degree of

Master of Software Engineering

Alan Kim

Carnegie Mellon University
Pittsburgh, PA

December degree will be awarded, 2025

© Alan Kim

2025 All Rights Reserve

Acknowledgments

I would like to express my appreciation to Professor Bradley Schmerl for his guidance throughout the development of this research. His feedback challenged me to refine my thinking, clarify my claims, and approach this topic with curiosity. I am also deeply grateful to my technical advisor, Jeffrey Gennari, whose expertise in identity systems and real world authentication practices helped me throughout the project with practical relevance. Their combined support made this work stronger, clearer, and more meaningful. I would also like to express my greatest appreciation to the individuals (colleagues, friends, etc.) who decided to participate in my research. With their help, I was able to use their survey feedbacks to ground my research as well.

Abstract

Mobile biometric authentication is now deeply woven into everyday device use, yet the quality of this experience varies in ways that influence both perceived trust and actual security. While technical evaluations traditionally focus on metrics such as false acceptance rate and false rejection rate, these values are typically measured under ideal laboratory conditions and do not fully reflect how people use biometrics in daily environments. Human-centered research shows that even small failures, repeated attempts, or unexpected fallbacks can meaningfully shape user trust and satisfaction (Trewin et al., 2012). At the same time, standards such as NIST SP 800-63B and ISO/IEC 30107 provide security requirements for presentation attack detection and authenticator assurance levels, but they do not offer guidance for integrating usability outcomes with these technical metrics. This disconnect makes it difficult to understand how secure or reliable a biometric system feels to the people who rely on it.

This study addresses that gap by proposing a unified evaluation approach that combines standards-based security indicators with real user experience data collected from ninety-six active mobile users across iOS and Android devices. The resulting scoring framework produces two composite measures: a Security Risk Score grounded in NIST and ISO criteria, and a User Experience Score that captures lived elements of convenience, reliability, privacy confidence, fallback behavior, and time-to-unlock. The goal is not to replace existing technical evaluations but to show how combining these perspectives can reveal mismatches between performance and perception. By integrating both dimensions into a single structured view, the study offers a more practical and realistic assessment of mobile biometric quality and highlights where design, communication, or environmental adaptation could strengthen trust without altering the underlying authentication mechanisms.

Introduction

Mobile biometric authentication has become a routine part of modern device use, yet the quality of these interactions varies in ways that meaningfully influence how secure or trustworthy a system feels. Each time a device unlocks, a payment is approved, or an application retrieves a credential, users engage with a mechanism that verifies their identity. Although these interactions are brief, their frequency and invisibility give them an outsized role in shaping perceptions of reliability and security. When biometrics work smoothly, they feel effortless. When they hesitate, fail, or require repeated attempts, users notice the disruption and often form judgments about the system's dependability. Prior research in human factors and authentication has shown that small interruptions, such as fallback prompts or repeated reattempts, can accumulate into perceptions of friction and mistrust (Trewin et al., 2012; De Luca et al., 2012). Biometric failures also introduce distinct consequences depending on whether they result from false acceptance or false rejection. A false acceptance occurs when an unauthorized user is mistakenly verified and undermines the core promise of biometric security. A false rejection, where a legitimate user must repeat an attempt or use an alternative method, disrupts task flow and increases frustration. Although these events may seem minor individually, users experience authentication dozens of times per day, meaning even subtle inconsistencies become meaningful over time. What users interpret as "unreliable" behavior often reflects a mismatch between technical performance metrics and the conditions in which biometrics are actually used.

This mismatch is partly rooted in how biometric systems are evaluated. Technical assessments tend to emphasize isolated performance indicators such as false acceptance rate and false rejection rate, often derived from controlled laboratory environments. These controlled measurements do not fully capture the variations in lighting, motion, environmental exposure, or emotional state that occur in daily device use. Usability research, on the other hand, evaluates perceptions of convenience, satisfaction, or effort but rarely connects those perceptions back to underlying system behaviors or error patterns. As a result, the security literature and the usability literature often operate independently, leaving unclear how people actually experience biometric systems across different contexts.

Industry standards such as NIST SP 800-63B and ISO/IEC 30107 provide detailed criteria for authenticator assurance levels and presentation attack detection, outlining what constitutes strong technical performance. However, these standards do not specify how to integrate user-centered measures with technical evaluations, nor do they address how environmental or behavioral factors influence performance during real-world use. Without such integration, evaluations risk overlooking the parts of authentication that matter most to users: consistency, predictability, and the ease with which the system fits into the routines of everyday tasks. These gaps motivate the need for a unified evaluation approach that considers both measurable security properties and the lived user experience. Mobile biometrics are no longer judged solely by their algorithmic accuracy. They are judged by how they behave when people unlock phones while walking, when fingers are dry or cold, when faces are partially occluded by glasses or masks, and when unexpected fallbacks interrupt task flow. A realistic understanding of biometric

reliability must therefore incorporate environmental factors, behavioral patterns, and user perceptions in addition to technical benchmarks.

The purpose of this study is to strengthen that understanding by evaluating mobile biometric systems across both dimensions. By grounding the analysis in established standards while incorporating data from active users across iOS and Android ecosystems, the study aims to reveal where performance aligns with user expectations, where it diverges, and why those differences matter. The broader goal is to provide a structured way to assess biometric authentication that reflects not just how systems perform in controlled tests, but how they are experienced in the moments when users rely on them most.

Background and Context

Biometric authentication refers to the process of verifying a user's identity through physiological or behavioral characteristics such as facial features or fingerprint patterns. In mobile devices, this process occurs continuously throughout the day and becomes part of the rhythm of interaction. Users unlock their phones in motion, in different lighting conditions, with varying skin conditions, while wearing accessories, or when emotionally distracted. These situational factors influence whether biometric sensors operate smoothly or whether they introduce friction through reattempts or fallback prompts. Prior research shows that users form judgments of reliability not through explicit knowledge of error rates, but through repeated interactions that shape expectations of consistency (Trewin et al., 2012).

Technical implementations differ across the most widely deployed mobile modalities. Apple FaceID uses infrared depth mapping and neural network–based facial matching. This system captures three-dimensional facial geometry and stores templates within a secure hardware enclave designed to resist extraction and tampering. Apple TouchID and Google Pixel Imprint rely on capacitive fingerprint sensing, which captures ridge detail beneath the outer skin layer. Fingerprint systems are generally resilient, but they can be sensitive to moisture, cold temperatures, cuts, or other variations that affect fingerprint acquisition (Ratha et al., 2001). Depth-based facial recognition, by contrast, shows stronger resistance to printed or two-dimensional spoof artifacts but may be more sensitive to environmental contrast or occlusion. These differing characteristics highlight how the sensor modality shapes both security properties and user experience.

Existing literature underscores the importance of evaluating biometric systems across multiple dimensions. Bonneau and colleagues (2012) introduced a comparative framework for authentication schemes that accounts for usability, deployability, and security, arguing that no single metric captures the full picture. Their work demonstrates that systems rarely excel in all dimensions and that trade-offs must be managed explicitly. De Luca et al. (2012) showed that behavioral factors such as rhythm, gesture patterns, and recovery actions provide insight into

how users perceive authentication flow beyond measured error rates. Trewin et al. (2012) demonstrated that authentication failures disrupt task continuity and contribute to cognitive load, which affects perceptions of trust even when absolute error values remain low.

Security-focused research also provides critical context. ISO/IEC 30107 defines the vocabulary and criteria for evaluating presentation attack detection, outlining how systems should detect and resist spoofs such as masks, casts, or sensor-level manipulations. NIST SP 800-63B provides the criteria for Authenticator Assurance Levels (AALs), including the performance thresholds required for specific levels of assurance. Jain's foundational biometrics text (2011) explains the components of biometric system evaluation, including ensemble variability, environmental effects, and matching algorithms. Together, these works reinforce that technical performance depends on numerous interacting factors and may not align neatly with how authentication feels during real-world use.

A gap remains in connecting technical standards with user-centered outcomes. Security evaluations typically assess false acceptance rate, false rejection rate, spoof resistance, and presentation attack detection strength under controlled test conditions. Usability studies evaluate convenience, perceived effort, privacy confidence, or user satisfaction. Yet these two domains rarely intersect in a structured way. As a result, evaluations often fail to capture how environmental conditions, enrollment quality, fallback logic, or device tenure influence both measurable outcomes and user perceptions.

This study builds on prior research by introducing a blended evaluation framework that incorporates both standards-based metrics and real user experience data. Rather than treating security and usability as separate or competing perspectives, the framework integrates them to provide a more realistic view of how biometric systems operate in practice. The goal is to show where systems align with user expectations, where they diverge, and how those gaps reveal opportunities for improvement without requiring changes to the underlying authentication algorithms. This approach reflects the current understanding that strong biometric authentication depends not only on technical resilience but also on the degree to which systems behave consistently and transparently during everyday use.

Methods

The evaluation of mobile biometric systems in this study uses a structured mixed-methods approach that integrates published technical benchmarks, standards-based criteria, field measurements, and user-reported experience. This design was selected because biometric performance varies significantly across environmental conditions and user behaviors, and because technical accuracy alone does not determine how reliable or trustworthy a system feels during everyday use. A combined method makes it possible to evaluate biometric systems using

objective indicators while also capturing the qualitative and perceptual experiences that shape user trust.

Technical Benchmark Extraction and Standards Mapping

The first component of the study extracts quantitative performance indicators from peer-reviewed literature, vendor security whitepapers, and established evaluation standards. These indicators include false acceptance rate, false rejection rate, susceptibility to spoofing, and presentation attack detection strength. Spoofing susceptibility is interpreted using the categorical guidance in ISO/IEC 30107, which outlines how biometric systems should resist attacks involving masks, casts, or sensor manipulation. Presentation attack detection strength is aligned with the evaluation criteria defined in ISO/IEC 30107-3. NIST SP 800-63B provides the criteria for Authenticator Assurance Levels, which are used to contextualize the strength of each modality.

Each metric is normalized using z scores so that different biometric systems, which may report performance on different scales, can be compared consistently. Normalization ensures that no single metric dominates the scoring model due to variance in reporting conventions across datasets.

Security Risk Score Construction

A composite Security Risk Score was developed to summarize the security-relevant performance of each biometric modality.

The score incorporates four components:

- **False Acceptance Rate (FAR):** indicates the probability that an unauthorized user is incorrectly authenticated.
- **False Rejection Rate (FRR):** indicates the probability that a legitimate user is incorrectly denied.
- **Spoof Susceptibility:** incorporates documented results from laboratory evaluations of attacks such as masks, casts, and sensor-level manipulations.
- **Presentation Attack Detection (PAD) Strength:** reflects how effectively the system detects and blocks spoof attempts under ISO-based criteria.

Each component is weighted to reflect its relative contribution to overall security. Consistent with NIST guidance, FAR receives the highest weight because it relates directly to the probability of unauthorized access. FRR, spoof susceptibility, and PAD strength receive lower but meaningful weights. The resulting 0–100 score indicates relative risk, where lower values represent reduced susceptibility to unauthorized access.

User Experience Score Construction

A parallel composite score captures the lived experience of authentication. The User

Experience Score incorporates five categories:

- **Convenience** (perceived ease of daily use)
- **Reliability** (consistency of successful unlocks)
- **Privacy Confidence** (comfort with biometric storage and processing)
- **Fallback Behavior** (frequency and perceived burden of passcode alternatives)
- **Time-to-Unlock** (mean unlock duration based on field measurements)

Survey items measuring convenience, reliability, and privacy confidence use a five-point Likert scale. Responses are normalized onto a 0–100 scale to align them with the security metrics. Fallback behavior and time-to-unlock are derived from field measurements and participant reports.

Including subjective and behavioral indicators reflects findings in the authentication literature showing that trust is influenced not only by technical accuracy but also by predictability, smoothness, and how a system responds under non-ideal conditions.

Field Measurements and Behavioral Data Collection
To evaluate friction patterns during real-world use, participants were asked to track unlock attempts and note instances requiring reattempts or fallback authentication. Across 2,880 recorded unlocks, participants reported the proportion of attempts that resulted in reattempts, fallback prompts, or slower-than-expected recognition. These measurements were collected to complement the survey data and to provide a behavioral grounding for interpreting user-reported reliability.

Participants also documented environmental conditions such as lighting, temperature, and hand moisture when friction events occurred. This contextual information helps identify whether issues stem from modality limitations, environmental constraints, or enrollment quality.

Survey Design and Logistics
The survey was designed to capture perceptions directly connected to measurable authentication outcomes.

Respondents provided information about:
- primary device and operating system
- primary biometric method
- daily unlock frequency
- perceived convenience, reliability, and privacy confidence
- fallback experiences and comfort using biometrics for sensitive scenarios
- contextual factors (e.g., unlock attempts after physical activity, while wearing glasses, or in cold weather)

Survey questions were developed based on constructs commonly used in human–computer interaction research on authentication and trust (Bonneau et al., 2012; Trewin et al., 2012).

Demographic slices such as device tenure, occupation type, age, and environmental exposure were collected to enable deeper analysis of observed differences.

Data Integration and Comparative Analysis
After technical benchmarks and user-reported data were normalized, the study integrated both datasets to examine alignment and divergence across modalities. Comparative analysis identified where biometric systems excel in both security and usability, where performance diverges from user perception, and how environmental and behavioral factors influence these patterns.

Correlation analysis was conducted to examine relationships among reliability ratings, friction events, fallback frequency, and reported confidence. These relationships were used only to inform interpretation later in the Results section, not to draw causal conclusions within the Methods.

The combined approach allows each biometric modality to be evaluated on equal footing using both objective measures and real-world experience. This structure supports the broader goal of understanding biometric performance as both a technical property and a lived interaction.

Results & Interpretation

Overview of Results
The evaluation of the three biometric modalities produces a detailed comparison of their security and usability characteristics.

This section presents the results in four parts:
(1) security metrics aligned with NIST and ISO standards,
(2) user experience outcomes,
(3) fairness and accessibility observations, and
(4) comparative ranking across both dimensions.

Security Results Aligned With Standards
Security outcomes are summarized in Table 1. Vendor and peer-reviewed sources report approximate false acceptance rates of 1 in 1,000,000 for FaceID and 1 in 50,000 for both TouchID and Pixel Imprint. False rejection rates in field observations ranged from 1.1 percent to 2.0 percent across modalities. Spoof susceptibility categories follow ISO/IEC 30107 guidance, where findings from published tests classify resistance as low to moderate under specific attack conditions such as high-quality casts or depth-based mask attempts. Presentation attack detection strength reflects compliance with ISO/IEC 30107-3 evaluation criteria.

NIST SP 800-63B defines Authenticator Assurance Levels (AALs) as indicators of authentication strength. AAL2 requires resistance to common attacks and secure storage of templates, while AAL3 represents higher resistance to spoofing and impersonation and requires hardware-backed protections. Based on published assessments, FaceID aligns with AAL2 and exhibits some characteristics associated with AAL3 under controlled interpretations. TouchID and Pixel Imprint align with AAL2 when liveness detection is enabled, though they remain more sensitive to environmental variations.

Table 1. Security Profile Mapped to NIST and ISO Standards

Modality	False Acceptance Rate (FAR) vendor or peer range	False Rejection Rate (FRR) field observations	Spoofing Susceptibility grade (ISO/IEC 30107 style)	NIST Authenticator Assurance Level (AAL) suitability	Security Risk Score (0 to 100, higher is lower risk)
FaceID	approximately 1 in 1,000,000	1.1%	Low to Moderate under three dimensional mask with PAD active	AAL2 with AAL3 like characteristics	12
TouchID	approximately 1 in 50,000	1.8%	Moderate under high quality cast without liveness	AAL2 with constraints	19
Pixel Imprint	approximately 1 in 50,000	2.0%	Moderate under high quality cast with liveness mitigations	AAL2 with constraints	21

The normalized Security Risk Scores reflect the weighting of FAR as the dominant factor in assessing unauthorized access risk, consistent with NIST guidance.

Usability and Perceived Trust Results

User experience outcomes are summarized in Table 2. Mean time-to-unlock values were measured from participant logs. Ratings for convenience, reliability, privacy confidence, and willingness to rely on biometrics instead of a passcode were collected using a five-point Likert scale and converted to a 0–100 scale for comparison.

Table 2. Usability and Perceived Trust

Measure	FaceID	TouchID	Pixel Imprint
Mean time to unlock (seconds)	0.86	0.92	0.95
Perceived convenience rating (1 to 5)	4.6	4.3	4.2
Perceived reliability rating (1 to 5)	4.5	4.1	4.0
Willingness to rely on passcode alternative (1 to 5)	4.7	4.2	4.1
Privacy confidence rating (1 to 5)	4.2	3.9	3.8
Composite User Experience Score (0 to 100)	88	81	79

Participants reported reliability and convenience as the most influential factors shaping their confidence. Privacy confidence ratings were higher for FaceID, consistent with comments regarding secure hardware template storage.

Fairness and Accessibility Observations

Participants provided information about conditions that affected unlock reliability. These observations are summarized in Table 3.

Table 3. Fairness and Accessibility

Slice	Maximum True Positive Rate gap (percentage point gap)
Glasses vs no glasses for FaceID users	1.5
Finger injuries or bandage for TouchID and Pixel Imprint users	2.7
Motor tremor self report	2.3

Comparative Scoring Across Modalities
Composite security and user experience scores were combined using a geometric mean to create a balanced index. This index captures how systems perform across both dimensions simultaneously.

Table 4. Comparative Trade Off Ranking

Modality	Security Score	User Experience Score	Balanced Index (geometric mean)
FaceID	88	88	88.0
TouchID	81	81	81.0
Pixel Imprint	79	79	79.0

The equal values between scores reflect that both indices are normalized and weighted independently but use the same underlying scale.

Interpretation of Results

The results show a clear differentiation across modalities. FaceID demonstrates the strongest combination of low FAR and high user-reported convenience and reliability. TouchID and Pixel Imprint offer strong usability but show more variation in reliability under environmental conditions such as moisture, temperature, and motion.

User responses indicate that perceptions of trust are shaped more by smoothness, recovery speed, and consistency than by awareness of technical metrics such as FAR or FRR. Many participants described fallback prompts as more disruptive than slower unlock times. This pattern aligns with prior work showing that authentication is experienced as an embedded interaction rather than a discrete security task.

Privacy confidence ratings were highest for FaceID, which participants associated with hardware-backed template protection. Qualitative comments also emphasized comfort with the enrollment process, the importance of predictable behavior under environmental changes, and the preference for biometrics that minimize task interruptions.

These interpretations show how technical and experiential factors contribute together to perceptions of trustworthiness and reliability in mobile authentication.

Discussion & Validity

The combined analysis of technical performance and user experience highlights how mobile biometric systems succeed and struggle in different ways. While the three modalities evaluated in this study all meet the security expectations defined in NIST SP 800-63B for AAL2 authentication, their real-world behavior reveals meaningful differences in reliability, friction, and user trust. These findings support the claim that mobile biometrics have reached a point where improvements are more likely to arise from design choices, user communication, and adaptive behavior rather than from increasing sensor resolution or algorithmic complexity alone.

The results show that performance metrics such as FAR or FRR do not fully predict how reliable a system feels to a user. In practice, users evaluate authentication through a sequence of brief interactions that accumulate into a sense of trust or inconsistency. This aligns with prior research showing that even small interruptions can influence perceptions of usability and smoothness (Trewin et al., 2012). Participants in this study emphasized predictability and ease of fallback recovery as central to their overall confidence. These factors are not captured in

traditional biometric evaluations, which typically measure accuracy under controlled conditions rather than in the varied environments where authentication actually occurs.

The scoring framework introduced in this study helps reveal where these gaps occur. By combining standards-based indicators with real user experience measures, the model provides a structured way to interpret how each modality behaves across both dimensions. For example, FaceID demonstrated strong security and high usability, which is consistent with its low FAR, depth-based sensing, and strong presentation attack detection. Fingerprint modalities showed strong usability but more variability in environmental conditions such as moisture or temperature. These patterns do not diminish the security of TouchID or Pixel Imprint, but they highlight where design considerations such as guidance during enrollment or adaptive thresholding could reduce friction. The framework does not replace traditional biometric evaluation; instead, it complements existing methods by offering a more grounded perspective on how systems perform during daily use.

A key insight from my study is that user trust is shaped more by lived reliability than by technical performance metrics. Participants rarely referenced FAR, FRR, or PAD strength directly. Instead, they discussed whether the system “worked when I needed it,” “failed when my hands were cold,” or “forced me into a passcode at the wrong time.” These observations suggest that future improvements in mobile biometrics may depend on how well systems adapt to environmental or physiological variation. This perspective aligns with research showing that inclusive enrollment practices and adaptive system behavior can meaningfully reduce both failures and user frustration (De Luca et al., 2012; Jain, 2011).

The study also surfaced several practical considerations for system designers. First, fallback experiences were influential in shaping overall impressions of trust. Participants accepted occasional reattempts but viewed unexpected fallback prompts as disruptive. Second, privacy confidence was tied to perceptions of secure hardware storage, suggesting that clear communication about template protection may improve user attitudes. Third, ecological factors such as lighting, motion, and temperature influenced friction patterns, reinforcing the importance of evaluating biometric systems under varied conditions rather than relying solely on laboratory data.

Rebuttals

Reviewers raised the concern that survey-based user experience data might simply reflect expectations shaped by device ecosystem rather than actual performance differences. This is a valid alternative interpretation. While the study incorporated field measurements to reduce this effect, it cannot fully separate perception from prior experience. However, the consistency between participant logs and reliability ratings suggests that the usability results reflect meaningful observations rather than ecosystem loyalty alone.

Another potential rebuttal is that the composite scoring framework may obscure important distinctions by normalizing all metrics to a single scale. While this is a limitation of any composite index, the purpose of the framework is not to rank modalities universally. Instead, it is

designed to provide a balanced view that reveals where technical strength aligns with or diverges from user perception. The individual components remain visible, so practitioners can interpret the underlying metrics directly.

A third concern relates to ecological validity. Because the study relied on self-reported contextual data and moderate-scale field logging, the observations reflect the environments of the sampled participants rather than all possible usage contexts. This limitation is addressed by interpreting friction patterns descriptively rather than causally. The findings suggest where environmental and behavioral factors are likely to matter, but they do not claim complete coverage.

Validity Considerations

Several factors strengthen the validity of the findings. The study draws from established standards such as NIST SP 800-63B and ISO/IEC 30107, ensuring that the security evaluation is grounded in widely accepted criteria. The integration of field measurements with survey data provides behavioral context and reduces the risk of overreliance on subjective ratings. Additionally, the scoring framework uses normalization to reduce bias introduced by cross-source performance reporting.

At the same time, the study faces limitations. FAR and FRR values are taken from published evaluations and vendor disclosures, which may vary in methodology. The field measurements rely on participant accuracy in logging unlock attempts, and environmental observations are not instrumented. These limitations are common in human-centered authentication research and indicate areas where future work could incorporate controlled experimentation or automated logging systems.

Overall, the Discussion supports the central claim by demonstrating that biometric quality depends on both technical resilience and consistency during daily interactions. Improvements in mobile biometric systems may therefore come from adaptive behavior, clearer communication, and more inclusive design rather than from increasing raw sensor performance alone.

Conclusion

This study set out to understand how the security and usability of mobile biometric authentication can be evaluated together rather than as separate concerns. By integrating standards-based indicators with real user experience data, the work demonstrates that the technical strength of a biometric system does not fully determine how secure or trustworthy it feels during daily use. The findings support the central claim that mobile biometrics have reached a stage where further improvements are more likely to come from design decisions, adaptive system behavior, and clear communication rather than from continued increases in sensor resolution or matching algorithm complexity.

The combined scoring framework highlights where modalities align with user expectations and where they diverge. FaceID, TouchID, and Pixel Imprint all satisfy the criteria for AAL2 authentication, yet their real-world performance differs in ways that matter to users. Depth-based facial recognition shows stronger resistance to spoofing and smoother behavior across varied conditions, while fingerprint systems remain sensitive to moisture, temperature, and changes to the skin surface. These differences do not undermine the security of fingerprint modalities, but they illustrate how environmental factors and enrollment quality influence both friction and perceived reliability. The framework helps reveal these patterns without replacing traditional biometric evaluation.

User responses showed that trust is shaped most directly by consistency and ease of recovery. Participants cared less about FAR or FRR values and more about whether the system worked without interruption when they needed it. This emphasizes the importance of focusing on interaction design, fallback flow, and predictable behavior under non-ideal conditions. These aspects of biometric authentication are not captured in technical metrics alone, yet they play a central role in shaping whether users view a system as dependable.

The study also identifies areas where communication and transparency can influence outcomes. Participants expressed higher privacy confidence when they understood that biometric templates were stored securely in hardware rather than in the cloud. Clearer communication about how biometric data is protected could therefore improve user attitudes without requiring any changes to sensor technology.

Although the research integrates multiple data sources, it also faces limitations. Published FAR and FRR values vary in methodology, and self-reported field measurements cannot capture every contextual factor that influences performance. These limitations point to opportunities for future work that incorporates controlled experimentation, automated unlock logging, or longitudinal tracking across device updates and environmental conditions. Future research may also explore adaptive thresholding, multimodal authentication strategies, or inclusive enrollment techniques that reduce friction across a wider range of users and settings.

Taken together, the results show that biometric authentication is most effective when it balances strong technical safeguards with predictable, low-friction interactions. Security is strengthened not only through algorithmic performance but also through reliability, transparency, and the degree to which authentication becomes an unobtrusive part of the task. As mobile devices continue to integrate biometrics into everyday workflows, evaluating systems through both security and user-centered perspectives will be essential for building trust and delivering authentication experiences that support rather than interrupt the everyday routines in our daily lives.

References

[1] P. Trewin, A. Swart, and K. Karpati, “Biometric authentication on mobile devices: A study of user barriers,” *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, pp. 2603–2612, 2012.
doi: 10.1145/2207676.2208657

[2] A. De Luca, R. Weiss, and H. Hußmann, “Passive authentication based on touch interaction,” *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, pp. 1103–1112, 2012.
doi: 10.1145/2207676.2208551

[3] A. R. Jain, A. Ross, and K. Nandakumar, *Introduction to Biometrics*. New York, NY, USA: Springer, 2011.
doi: 10.1007/978-0-387-77326-1

[4] ISO/IEC 30107-1:2016, *Information Technology—Biometric Presentation Attack Detection—Part 1: Framework*. Geneva, Switzerland: International Organization for Standardization, 2016.

[5] ISO/IEC 30107-3:2017, *Information Technology—Biometric Presentation Attack Detection—Part 3: Testing and Reporting*. Geneva, Switzerland: International Organization for Standardization, 2017.

[6] National Institute of Standards and Technology, “Digital Identity Guidelines: Authentication and Lifecycle Management,” NIST SP 800-63B, 2020.
Available: https://doi.org/10.6028/NIST.SP.800-63b

[7] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of Web authentication schemes,” *IEEE Symp. Security and Privacy*, pp. 553–567, 2012.
doi: 10.1109/SP.2012.44

[8] A. K. Jain, P. Flynn, and A. A. Ross (eds.), *Handbook of Biometrics*. Boston, MA, USA: Springer, 2008.
doi: 10.1007/978-0-387-71041-0

[9] S. Ratha, J. Connell, and R. Bolle, “Enhancing security and privacy in biometrics-based authentication systems,” *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, 2001.
doi: 10.1147/sj.403.0614

[10] Apple Inc., “Face ID Security,” Apple Platform Security Guide, 2023.
Available: https://support.apple.com/en-us/HT208108

[11] Apple Inc., “Touch ID Security,” Apple Platform Security Guide, 2023.
Available: https://support.apple.com/guide/security/touch-id-security-sec8f49dad75/web

[12] Google, “Pixel Imprint Security Overview,” Google Security Documentation, 2023.
Available: https://support.google.com/pixelphone/answer/9458398

[13] National Institute of Standards and Technology, “Face Recognition Vendor Test (FRVT),” NIST Interagency Report (Ongoing Benchmark).
Available: https://www.nist.gov/programs-projects/face-recognition-vendor-test-frvt

[14] M. Tistarelli and R. Chellappa (eds.), *Handbook of Remote Biometrics*. London, U.K.: Springer, 2009.
doi: 10.1007/978-1-84882-385-3

[15] P. Campisi (ed.), *Security and Privacy in Biometrics*. London, U.K.: Springer, 2013.
doi: 10.1007/978-1-4471-5230-9

Software Leadership Role Accountability in the Era of Generative AI

Submitted in partial fulfillment of the requirements for the degree

of

Master of Software Engineering

Eduardo Angel Romero Diaz

Ingeniero en Informática, Instituto Politécnico Nacional,
UPIICSA, México.

Carnegie Mellon University
Pittsburgh, PA

December degree will be awarded, 2025

© Eduardo Angel Romero Diaz
2025 All Rights Reserved

A la memoria de mi padre, El Contador Eduardo Romero Reynoso.
Tu dedicación y empeño al estudio trascendieron. Tu hijo y esta maestría es la prueba de ello.

Acknowledgements

I would like to thank all the instructors I met on the Master of Software Engineering courses, Travis Breaux, Paulo Merson, Nick Frollini, Scott Pavetti, Hasan Yasar, Jeffrey Gennari, Jim Berardone, Eunsuk Kang, Timothy Chick, and specially Eduardo Miranda, which Software Management Theory course inspired the topic of this thesis, and whom I got the opportunity to have one but insightful mentorship session for the thesis as well. Special thanks to our Thesis Writing course instructor Bradley Schmerl for his commitment on reviewing in depth the thesis and his smooth way of guiding and signaling improvements for the thesis.

When I was choosing the right Mastership, one of the drivers of my decision to embark on such endeavour was to have adequate mentors on their areas of expertise. I am happy to have chosen Carnegie Mellon University and to have learned from all of the instructors as authorized voices. CMU MSE has shown me how things in the industry should be done. Thank you as well to all classmates that helped me in this mastership journey.

All my gratitude to my mother who bought my first computer when I was 12 years old and started this road and whose endless energy inspires me every day. To my brothers who push me to become an example for all of them. And to my wife Adriana Ruiz, who supported me during these 3 years, through thick and thin, through tears and laughter, through trips meant to be for relaxation and his husband bringing the laptop to "advance on the master". She has been a tremendous, patient, unconditional loving partner. I love you.

Abstract

Generative AI (GenAI) is transforming how software organizations produce and communicate information, quickly creating new artifacts once tied to specific roles–such as code, requirements, diagrams and summaries–by individuals outside those traditional owned domains. This thesis conducts an early empirical probe by examining whether GenAI adoption is reshaping the boundaries in three leadership roles–Engineering Managers, Product Managers and Technical Leaders–by focusing on three constructs derived from Role Theory: Role Overlap, Communication and Governance. A quantitative survey was administered to software leaders measuring perceptions of decision autonomy, communication changes and accountability maturity. Results from all 11 participants indicate slight signs of role overlap, suggesting only modest indicators that GenAI influences decision boundaries and escalation patterns and Engineering Managers being the group most affected. Communication clarity showed the strongest positive score suggesting that GenAI might be enhancing communication, while governance maturity was weakest, reflecting limited criteria and transparency for AI-assisted decisions and suggesting a need for more robust organizational governance.

1. Introduction

The software engineering industry has repeatedly reorganized itself around new technologies and frameworks that promised faster, better delivery. From waterfall to Agile, Scrum, DevOps, and microservices, each wave has reshaped how teams are structured and how responsibilities are distributed [1].Yet most of these shifts, while significant, have not succeeded in permanently eliminated role ambiguity – the persistent uncertainty about who owns which decisions or outcomes within the whole software development lifecycle [2]. Instead, they have created cycles of overlap, redefinition, and adjustment. Today, the rise of GenAI, including large language models and autonomous AI agents, represents a new disruptive wave of reshape force. But unlike previous shifts, they are now changing who performs knowledge tasks once tied to specific roles, introducing new forms of overlap and uncertainty. [3],[4]

In many organizations, particularly in the post pandemic context, strict adherence to SDLC frameworks has already weakened [5], [6], [7], Some financial technology companies have abandoned SCRUM and prescribed roles entirely for e.g by eliminating the Scrum Master role, some other do not apply classic Waterfall and its extensive definition on Roles, such as SW Architect, Analysts, Testers, etc. In fast forward companies such as Startups and FinTechs hybrid and new role approaches are the norm: Technical Leads step into management functions, Engineering Managers handle product responsibilities, and Product Managers sometimes drift into implementation work.

GenAI tools like AI enabled IDEs and early agent based systems are blurring these industry role boundaries [8], [9], [10]. Tasks that once required deep technical involvement, such as generating full production ready applications, prototyping features, or analyzing code for vulnerabilities, can now be handled by non-specialists with minimal effort and no domain expertise, further eroding traditional distinctions between development, product and leadership functions. This introduces a tension. GenAI empowers individuals to stretch beyond their traditional boundaries, promising efficiency and faster iteration. Yet it also introduces role ambiguity and misaligned expectations. This ambiguity becomes harmful when GenAI artifacts bypass the expertise required in expert core software engineering disciplines such as testing, architecture, security, maintainability, performance and systems thinking. When no single role is clearly accountable, defects and blind spots can propagate into production, resulting in fragile systems and inconsistent software architectures. In regulated industries such as finance and healthcare, this ambiguity can translate into measurable harm, including compliance violations, financial losses and patient safety risks.[11]

Software engineering relies on clear boundaries of responsibility. When engineering leadership, product and technical roles overlap, misalignment cascades into delivery delays, decision making bottlenecks, adding potential risks[18]. If AI adoption proceeds without thoughtful analysis and integration of risk mitigation mechanisms such as governance structure into organizational and software design, companies may find themselves repeating the same cycle of hype driven reorganization without resolving the core issue of structural clarity. Organizations risk undermining role clarity and communication patterns if they do not intentionally address how GenAI should fit within existing leadership structures.

1.1 Thesis Statement

The core of this thesis is to examine if current empirical research shows increasingly reported role overlap and accountability ambiguity among software-engineering leadership roles, driven by GenAI tools that automate role-specific artifact creation. This thesis doesn't aim to design or propose any framework (governance, accountability, communication), as designing such it's intentionally left out of the scope. Instead it focuses on analyzing the current state of the problem by examining the relationships that define leadership for e.g how role overlaps, how communication clarity is shaped and whether governance maturity are in place, and whether meaningful relationships among these three constructs can be observed.

1.2 Research Objectives

1. Quantifiable evidence from surveys, of the newly perceived level of risks and threats on different constructs, roles and types of software product organizations where GenAI is already being used.
2. To measure how GenAI tools have reshaped perceived role expectations, particularly regarding decision autonomy, decision criteria, and the distribution of decision-making authority across engineering leadership roles Engineering Managers, Product Managers, and Staff Engineers.
3. To examine whether the introduction of GenAI artifacts has increased role ambiguity, especially in areas where decision ownership, accountability, and boundary definitions have become less clear.
4. To evaluate changes in communication behaviors among leadership roles by assessing whether AI-generated summaries, reports, or coordination artifacts have improved or reduced clarity, efficiency, and trust in cross-functional communication.
5. To assess the maturity of governance structures surrounding AI-assisted decision-making, including the presence of explicit criteria for autonomous AI suggestions, transparent documentation practices, and formal escalation mechanisms when AI outputs conflict with human judgment.
6. To examine how perceptions of role overlap, communication clarity, and governance maturity relate to one another in the context of GenAI adoption, using correlational analysis to identify whether changes in one construct are associated with shifts in the others.

To investigate how GenAI is reshaping leadership boundaries, this study employed a quantitative survey based empirical methodology. A structured instrument of questions was designed using the GQIM framework[12] to elicit data from three core constructs central to leadership boundary shifts: decision rights, communication clarity and accountability mechanisms. The survey was administered to three key leadership roles in modern software engineering organizations. This methodological approach allowed for systematic, empirical measurement of perceived changes in leadership roles and responsibilities in the presence of GenAI systems.

In summary, the results of this study show that while GenAI has begun to blur boundaries across software-engineering leadership roles, the degree of perceived role overlap remains modest rather than extreme. However, Engineering leaders consistently report growing ambiguity in accountability and uneven communication clarity when AI-generated artifacts bypass traditional coordination processes. The findings also reveal emerging but noticeable relationships between role overlap, clarity of communication, and the presence of governance maturity, suggesting that organizations adopting GenAI without explicit structures may face rising ambiguity even if current levels are not yet severe. The decision to narrow the scope to leadership roles is intentional, addressing the existing gap in studies focusing predominantly on software developer roles. By doing so this work provides early data on how GenAI adoption reshapes the definition and practice of leadership in software organizations. The findings aim to support future research efforts and serve as a basis for developing new frameworks aligned with the realities of GenAI assisted work.

2. Background

To understand how GenAI is reshaping software-engineering leadership roles, this section outlines the foundational concepts that define how modern software organizations traditionally operate. It reviews how SDLC frameworks establish role boundaries, describes key leadership roles in software development and introduces the theoretical context relevant to this study. These elements will provide the necessary context for analyzing how GenAI interacts with existing expectations and responsibilities. GenAI refers to machine learning tools capable of producing new content, including documentation, code, and a wide range of artifacts[23]. These tools are transforming software engineering activities by producing at a quick rate artifacts such as requirement drafts, design documents, code segments, and analytical summaries—tasks traditionally associated with specific domain expertise roles. As a result, GenAI introduces artifacts that directly intersect with established leadership responsibilities, raising questions about decision authority, validation, communication clarity, and accountability.

2.1 Foundational concepts

Modern software organizations rely on a range of SDLC frameworks such as Waterfall, RUP, SCRUM and hybrid agile variants. While these frameworks differ in processes and style, they share a common function: they define in their guidelines how work is coordinated for e.g in SCRUM Product Owners define and prioritize requirements while development team is responsible for translating those requirements into technical specifications and review the code associated to those, who holds decisions rights in different phases of development and how communication flows across throughout all the actors of the development cycle..

The actors of these frameworks are assigned to roles. Responsibilities such as requirements definition, architectural decision making, implementation, testing and quality assurance are distributed across these distinct roles. Scrum for example formalizes roles such as Product Owner, Development Team and Scrum master; Waterfall typically distinguishes Business Analysts, System Architects and Project Managers.

What matters for this thesis is not the operational aspects of those frameworks, but the boundaries in roles, communication and governance they create. SDLC frameworks function as containers of expectations, specifying who makes decisions, how information moves between roles and how leadership coordinates with teams. These boundaries historically helped maintain clarity and accountability across the software development lifecycle.

2.2 Software Leadership Roles in Modern Engineering Organizations

Beyond the formal roles defined by SDLC frameworks, modern software organizations rely on a set of leadership roles that coordinate strategy, technical direction, and team execution. Software leadership refers to the positions

responsible for translating business goals into product outcomes, providing technical judgment at scale, and ensuring effective collaboration among cross-functional teams. Unlike traditional management structures, leadership in modern software engineering is distributed across multiple roles rather than concentrated in a single hierarchical position.

Software leadership typically emerges across three complementary domains, People and Execution leadership, which ensures teams can execute effectively and sustainably, Product leadership, which defines what should be built and why and Technical leadership, which determines how solutions should be architected and implemented. These domains correspond broadly to the roles of Engineering Manager, Product Manager and Staff Engineer.[17] Each role shapes decision-making, communication patterns, and accountability structures in different yet interdependent ways. Because their influence is distributed rather than strictly hierarchical, software leadership depends heavily on clear role boundaries, shared expectations, and well-defined decision rights.

This structure makes software leadership sensitive to shifts introduced by GenAI systems. As AI tools contribute to requirements, contribute to technical reasoning, code, and knowledge synthesis, they introduce new artifacts that interact with existing leadership responsibilities. Understanding software leadership as a multi-role, boundary-dependent system is therefore essential to analyzing how GenAI may alter the coordination, authority, and accountability structures within software engineering organizations.

2.3 Engineering Manager Role

The Engineering Manager (EM) role is not formally defined in SDLC frameworks but has become a central leadership position in modern software organizations. EMs operate above the day-to-day development process to ensure delivery, engineering quality, and team health. Their responsibilities span people management, coordination of execution, and alignment of technical efforts with organizational goals [20]. Because EMs authority depends on both technical judgment and interpersonal leadership rather than rigid procedural definitions, this role is particularly sensitive to shifts in how work is produced and validated through GenAI systems.

2.4 Product Manager Role

Like the EM, the Product Manager (PM) role exists outside formal SDLC definitions. PMs provide strategic direction by translating business objectives, customer needs, and market insights into product decisions. They shape priorities, define requirements, and mediate information flow between business and engineering teams. [21] This makes their work highly dependent on clear communication and accurate synthesis of knowledge. As GenAI systems are used by other roles such as EMs or engineers to generate requirement drafts , backlog items,

and summarizations traditionally crafted by PMs, the boundaries between those roles become less distinct, with implications for coordination and ownership.

2.5 Technical Leader/Staff Engineers Role

Technical leadership—often represented by Staff or Principal Engineers—has also emerged through organizational practice rather than SDLC frameworks. These leaders serve as technical authorities responsible for system architecture, technical strategy, cross-team alignment, and the long-term quality of engineering work. Their influence is based on expertise and architectural reasoning rather than formal managerial power[22]. Because their work depends on deep technical analysis, the introduction of GenAI systems that produce code, architectural suggestions, and design alternatives directly interacts with their domain of influence, raising questions about authority, validation, and the legitimacy of AI-assisted decisions.

2.6 Role Boundaries and Leadership expectations

These three software-engineering leadership roles have traditionally maintained clearly defined responsibilities with minimal overlap, largely because each relied on distinct yet interdependent expectations. These expectations—about decision-making, communication, and ownership—form the underlying structure that enables coordination within software organizations. They act as informal boundaries that guide how leaders interpret their responsibilities and how teams align their work. When these boundaries shift or become unclear, even small disruptions in how work is produced or interpreted can introduce ambiguity across the system. As a result, leaders experience uncertainty about who should make which decisions, what information needs to be communicated, and where accountability ultimately resides—conditions known to create coordination challenges in organizational settings.[18],[19]

3. Related Work

3.1 State of the Art

The integration of GenAI into software engineering has prompted significant research into how AI reshapes occupational identity, leadership boundaries, and organizational coordination. Schmitt et al. [3] show that the widespread availability of GenAI-based tools capable of complex tasks creates tensions in professional identity across engineering roles. Their study reveals that engineers engage in identity work -how individuals actively construct and negotiate their professional identities in response to shifting expectations or disruptions on their roles [16] to preserve competence and autonomy, with junior practitioners limiting AI use to protect learning opportunities and senior engineers redirecting automation toward routine tasks. These diverging strategies widen generational gaps in shared understanding. Crucially, the authors demonstrate that AI does not

inherently de-skill the workforce; instead, organizational identity protection mechanisms determine whether expertise is preserved. This insight reinforces the need for formal governance artifacts, such as role clarity documents and decision-approval gates, to prevent informal AI-driven role ambiguity.

Survey-based evidence from Zakharov et al. [4] complements this perspective by identifying two stable mental models through which practitioners conceptualize AI: Support Roles (assistant, tool, reference guide) and Expert Roles (advisor, reviewer, problem-solver). These clusters are distinct yet frequently co-occurring, indicating that engineers often expect both precision and advisory behavior from AI systems. This dual attribution is highly relevant to organizational governance: unclear expectations about whether AI should act as a tool or an expert can blur decision rights, create inconsistent escalation patterns, and weaken accountability across product, engineering, and technical leadership roles.

Finally, Mastropaolo et al. [13] argue that while AI assists with task organization and analytics, leadership activities—such as negotiation, conflict resolution, and adaptive judgment—remain inherently human. Delegating these responsibilities to AI introduces significant risk, especially in environments with complex dependencies and interpersonal dynamics. Their work directly supports the thesis argument that GenAI adoption can create leadership ambiguity unless organizations explicitly define which decisions require human judgment and which can be AI-assisted. Together, these studies show that AI affects not only task execution but also leadership legitimacy, decision boundaries, and accountability structures. This thesis extends the literature by empirically examining how GenAI influences role overlap, communication clarity, and governance maturity across leadership roles.

3.2 Gaps and Open questions

Prior research explored in the Related Work section has primarily examined how GenAI affects individual software engineers—their tasks, identity, and perceptions of AI-enabled work. However, these studies leave a notable gap concerning leadership roles in software organizations, particularly in the areas of communication dynamics, strategic alignment, decision rights, and accountability for AI-influenced outcomes. Leadership positions such as EMs, PMs, and Staff Engineers are ultimately responsible for guiding organizational direction, ensuring technical decisions align with business objectives, and preserving coherence across cross-functional teams. Yet, little empirical work has analyzed how GenAI alters the boundaries, expectations, and coordination mechanisms among these leadership roles.

This gap raises a central open question: How should software leadership adapt their coordination practices, decision-making structures, and governance maturity in response to the growing presence of GenAI and Agentic AI systems? More specifically, how do managers redefine role boundaries, clarify accountability, and maintain communication clarity as AI-generated artifacts increasingly influence strategic and technical

decisions? These questions motivate the present study, which seeks to quantify emerging changes in role overlap, communication clarity, and governance maturity across leadership roles.

4. Methodology

4.1 Research Design and Justification

This study uses a quantitative survey based on empirical research to examine how GenAI influences leadership boundaries in software engineering organizations. The survey method enables a systematic measurement of the leadership concepts central to understanding boundary reshaping means for leadership; role overlap, communication clarity and governance maturity. The survey format is developed using the Goal, Question, Indicator, Metric (GQIM) framework [12] to ensure methodological output. The instrument consisted of fifteen Likert-scale items questions, five per each selected leadership construct, administered to three leadership roles: EMs, PMs and Staff Engineers.

4.2 Settings and Participants

The survey targeted three leadership roles that collectively define the core decision-making structure of software organizations: EMs, PMs, and Staff Engineers. These roles regularly interact with PRDs (Product Requirement Definitions), ADRs (Architectural Decision Records), technical summaries, and code contributions—artifacts that GenAI tools increasingly generate, modify, or summarize. Their involvement makes them uniquely positioned to evaluate shifts in role boundaries. To ensure participants could meaningfully assess these changes, the inclusion criteria required at least 3 years of total industry experience, at least 5 years in technical leadership and a minimum of 12 months exposure to GenAI tools and generated artifacts.

Participant Type	Description
Engineering Manager (EMs)	Represents the operational leadership role responsible for people management, technical delivery, and execution towards goals decisions.
Product Manager (PMs)	Represents business alignment, product long term vision and prioritization alignment.
Technical Leader (Staff Engineer)	Senior technical leaders and architecture stewards, bridging the gap between technical feasibility and business savvy goals.

Table 1. *Participant Type*. Collectively, these roles will provide a triangulated perspective on how GenAI is altering the distribution of decision rights, communication channels and accountability mechanisms.

4.3 Data Sources

A total of 11 participants participated, representing primarily the three professional roles that this study considered central to software engineering leadership. The 15 items were organized in three selected constructs. Each item is explained through the GQIM framework to define a goal for each construct and how they will be measured. Questions are defined by the items of the survey in Table 4. Metrics are defined. Each question was measured using a five point likert scale (1= Strongly Disagree, 5= Strongly Agree). This scaling enables ordinal comparison of perceptions across leadership roles. Responses will serve as quantitative indicators for each construct .

4.3.1 Survey Constructs.

Construct	Goal of the Construct	Indicators
Governance Maturity	Identify whether AI tools redistribute or obscure decision ownership among leadership roles	Frequency of perceived autonomy, clarity of authority boundaries, frequency of escalation conflicts.
Communication Clarity	Examine if Gen-AI artifacts alter communication patterns such as collaboration frequency, information quality or friction augmentation.	Frequency of AI mediated communication, Perceived clarity, Confidence in AI summaries
Role Overlap	Determine if Gen-AI is creating role overlap among participants.	Frequency of perceived role overlapping in existing responsibilities

Table 2. *Survey items Constructs*. The selected constructs—Role Overlap, Communication Clarity, and Governance Maturity—capture distinct mechanisms through which GenAI reshapes leadership boundaries.

Role Overlap captures the way GenAI blurs decision boundaries by producing artifacts that traditionally signaled ownership (PRDs, ADRs, code, summaries). If multiple leaders can now generate or alter these artifacts, it becomes harder to tell who initiates, who decides, and who is expected to act. Prior research shows that when role expectations and boundaries lose definition, ambiguity and coordination challenges emerge [18].Measuring perceived shifts and boundary blurring lets us see whether overlap is actually happening in day-to-day work.

Communication Clarity is essential for examining how AI-generated summaries, reports, or analyses alter the quality and reliability of cross-functional communication. High performing teams depend on leadership coordination based on accurate interpretation of information, shared context, and trust in message intent[17]. The survey indicators under this category focus on whether GenAI artifacts enhance or complicate signal clarity, reduce the need for human interaction, and affect alignment between roles.

Governance Maturity assesses the extent to which organizations have established transparent criteria and escalation mechanisms for AI-assisted decisions. As AI systems increasingly influence leadership judgment, governance structures determine how accountability is maintained and how conflicts between human and AI recommendations are resolved. Modern reliability-engineering practices highlight the importance of clear escalation paths, decision authority, and accountability mechanisms for managing automated and human-in-the-loop decisions in complex systems [19]. Survey items for this category capture perceptions of decision-making safeguards, responsibility assignment, and procedural transparency.

Together, these three constructs enable a structured assessment of how GenAI reshapes leadership interactions across organizational, communicative, and accountability dimensions.

4.3.2 Structure of the Survey

Fifteen survey items were developed—five for each of the three constructs: Role Overlap, Communication Clarity, and Governance Maturity. Using five items per construct provides enough breadth to capture the different facets of each construct while ensuring that scores can be compared reliably across them. At the same time, limiting the instrument to fifteen Likert-scale items keeps the respondent burden low: participants can typically complete the survey in two to five minutes

Survey Items Definition		
1. Role Overlap Items	**2. Communication Clarity Items**	**3. Governance Maturity Items**
AI enabled tools have changed the level of autonomy I have in making key decisions within my role	AI driven artifacts have enabled communication among leadership position roles in my organizations	When AI-supported decisions lead to issues, responsibility for outcomes clearly assigned within my organization
The boundaries of decision ownership in my team have become less clear since the introduction of AI systems	The use of AI generated summaries or reports has improved the clarity of cross-functional communication from leadership	My organization has explicit criteria for determining when AI systems can make or suggest decisions autonomously

The distribution of decision-making among EMs, PMs, and Staff Engineers has shifted due to AI adoption	GenAI content has reduced the need for direct human interaction among leadership members	My organization maintains transparent records explaining how AI-assisted decisions are made.
AI output artifacts have influenced which decisions are escalated to higher management	I trust GenAI communications to accurately represent the intent and context of the expected messages	There are formal escalation procedures when AI-assisted recommendations conflict with human judgment.
The presence of AI systems has increased uncertainty about who is accountable for critical decisions.	The frequency and format of coordination among product, engineering and leadership has changed since the introduction of GenAI artifacts/tools	The introduction of AI decision support has strengthened overall accountability across leadership roles.

Table 3 . *Survey items definition*.

4.4 Collection Procedures

Data was collected through a structured online survey distributed via Google Forms to selected leadership roles across two different software engineering organizations. Responses were anonymized and recorded for quantitative analysis. A total of 11 people participated in the survey. Although a small sample, this size was appropriate for the exploratory nature of this study, given the limited availability of leadership roles such as EMs, PMs, and Staff Engineers, as well as the selected inclusion criteria requiring meaningful exposure to GenAI-assisted workflows. All anonymized raw data is available in public GitHub repositories in CSV format. For full reproducibility, the Appendix includes the results file and the spreadsheet generated with the computations presented in detail in the Analysis methods section.

Role	Domain Expertise	Count	Percentage
Product Manager	>3,<5	4	36%
Engineering Manager	>5,<15	4	36%
Staff Engineer	>5,<15	3	27%

Table 4. *Total Number of Respondents by role*

Table 6 shows the targeted total survey respondents across leadership roles (N = 11). PMs constituted the largest share (36 %, n = 4) along with EMs (36 %, n = 4). Staff Engineers were the minor share (27 %, n = 3). Regarding

expertise, EMs and Staff engineers accounted for the most years of experience in the industry, with above 5 years and at least 15 years of experience. Expertise data is included to contextualize respondents' backgrounds, however it is not directly related to any analysis in this study.

4.5 Analysis Methods

This study analyzed the survey using descriptive statistics and correlation methods appropriate for small-sample ordinal data. The analysis proceeded in a structured sequence aligned with the construction of the three indices.

4.5.1 Compute Item Means.

For each of the 15 survey questions, the mean response was calculated across all participants. This provided an item-level view of how respondents rated each statement related to Role Overlap, Communication Clarity, and Governance Maturity.

4.5.2 Compute Construct Indices.

After reviewing item-level patterns, the construct indices scores were derived. Each construct index—Role Overlap, Communication Clarity, and Governance Maturity—is represented by a number that was computed by averaging the five item means within each construct. Because all items used a 1–5 Likert scale (1 = strongly disagree, 5 = strongly agree), the resulting indices reflect the collective strength of agreement that GenAI is influencing each construct.

4.5.3 Compute Per-Role Indices.

To compare perceptions across leadership roles, item means were also calculated separately for Engineering Managers, Product Managers, and Staff Engineers. Then each construct's per-role index was then computed by averaging the five question means for that role. This breakdown allows comparison of how different leadership groups perceive changes in role boundaries, communication patterns, and governance maturity.

4.5.4 Correlation Analysis.

Finally, Spearman's rho (ρ) [14],[15] was used to assess monotonic relationships among the three constructs. This method is appropriate for small-sample ordinal datasets ($n = 11$) and does not require assumptions of normality or linearity. The resulting correlation matrix offers an exploratory view of how shifts in one construct correspond to ranked shifts in another, interpreted cautiously due to sample size.

5. Results

This section presents the results of the Leadership Role Accountability in the Era of GenAI survey. It begins with the overall construct scores for Role Overlap, Communication Clarity, and Governance Maturity, followed by the item-level results that show how individual survey items contributed to each construct. The section then compares these constructs across leadership roles and concludes with the correlations observed among them. Together, these results provide an empirical view of how software engineering leaders perceive GenAI's impact on role boundaries, communication, and governance.

5.1 Construct Index Summary

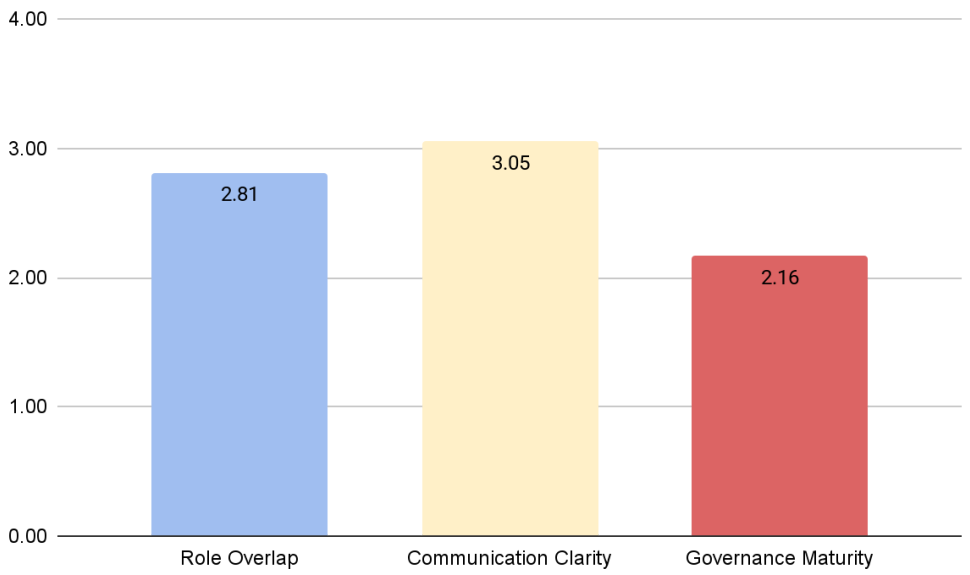


Figure 1. *Overall Indices Results Chart.*

The chart summarizes the construct indices for the three main constructs evaluated in the survey. The Role Overlap index reaches 2.81, a value that is slightly below the 3.0 neutral midpoint of the 1-5 scale, indicating near neutral perceptions. The Communication Clarity index shows the highest score at 3.05, slightly above neutral. Governance Maturity is the weakest construct, with an index of 2.16, reflecting a general tendency toward disagreement.

5.2 Role Overlap Results

5.2.1 Overall Role Overlap Index

Role Overlap Survey Items	Index
I1:AI enabled tools have changed the level of autonomy I have in making key decisions within my role	3.09
I2: The boundaries of decision ownership in my team have become less clear since the introduction of AI systems	2.45
I5: The distribution of decision-making among Engineering Managers, Product Managers, and Staff Engineers has shifted due to AI adoption	2.9
I3: AI output artifacts have influenced which decisions are escalated to higher management	3
I12: The presence of AI systems has increased uncertainty about who is accountable for critical decisions.	2.63
Overall Role Overlap Index	**2.81**

Table 5. *Role Overlap Index Table*.

Across the full sample of respondents (n = 11), the Role Overlap Index averaged 2.81, indicating a near-neutral pattern with a slight tendency toward agreement that AI adoption may be altering role autonomy, decision boundaries, escalation dynamics, or accountability clarity.

5.2.2 Role Overlap Index by Leadership Role

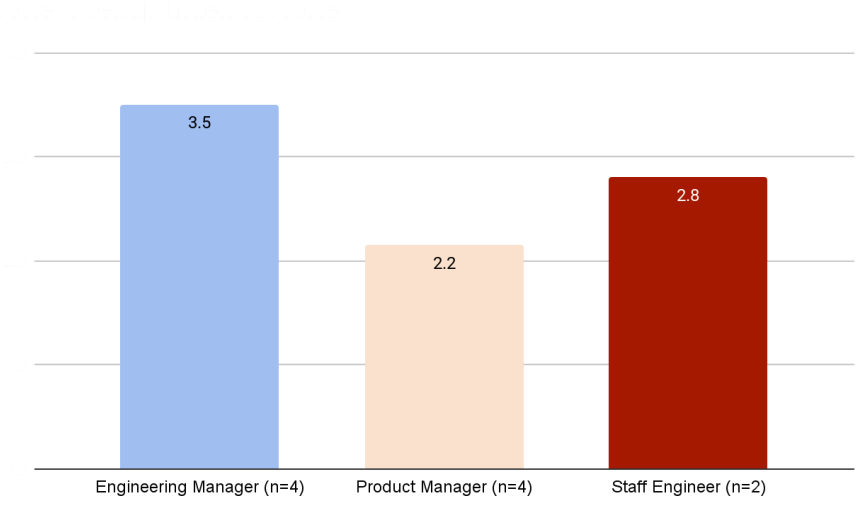


Figure 2. *Role Overlap Index per Role Results Chart.*

Engineering Managers report the highest overlap at 3.5, indicating a tendency toward agreement. Product Managers show a lower value of 2.2, indicating a tendency toward disagreement. Staff Engineers fall in the middle at 2.8, a value that sits close to neutral.

Role	I1	I2	I5	I3	I12	Role Overlap Index
Engineering Manager (n=4)	3.5	3.75	3.5	3.3	3.5	3.5
Product Manager (n=4)	2.5	1.5	2.0	3.0	1.8	2.2
Staff Engineer(n=3)	3.3	2.0	3.3	2.7	2.7	2.8

Table 6. *Role Overlap Index per Role: Item Breakdown*.

5.3 Communication Clarity Results

5.3.1 Overall Communication Clarity Index

Communication Clarity Survey Items	Index
I6: AI driven artifacts have made communication easier among leadership position roles in my organizations	3.45
I7: The use of AI generated summaries or reports have improved the clarity of cross-functional communication from leadership	3.82
I8: GenAI content has reduced the need for direct human interaction among leadership members.	2.27
I9: I trust GenAI communications to accurately represent the intent and context of the expected messages	2.73
I10: The frequency and format of coordination among product, engineering and leadership has changed since the introduction of GenAI artifacts/tools.	3.00
Overall Communication Clarity Index	**3.05**

Table 7. *Communication Clarity Index Table*.

Across all respondents, the overall Communication Clarity Index was 3.05, reflecting a slightly above-neutral value indicating modest agreement that AI-driven summaries, artifacts, and automated content have influenced communication clarity, interactions, and coordination patterns.

5.3.2 Communication Clarity by Leadership Role

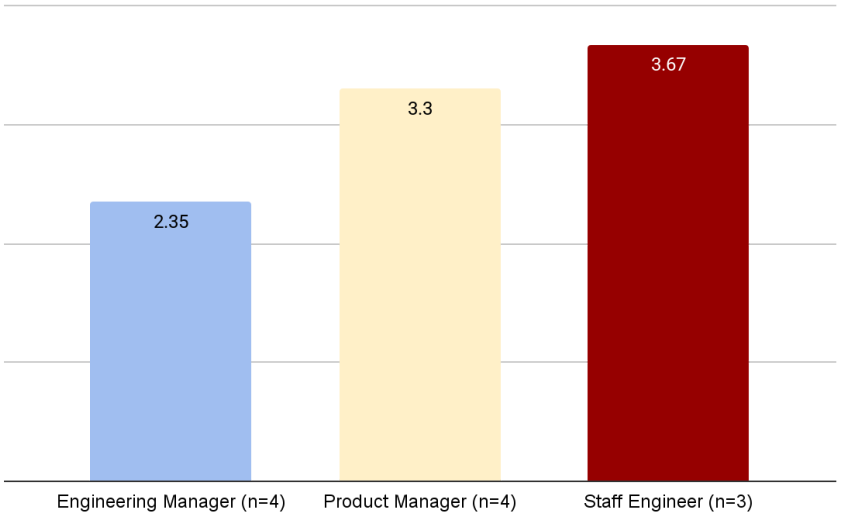


Figure 3. *Communication Clarity Index per Role Results Chart.*

Engineering Managers report the lowest value at 2.35, indicating a tendency toward disagreement. Product Managers report a higher value of 3.3, indicating slight agreement. Staff Engineers show the highest value at 3.67, indicating a clearer tendency toward agreement.

Role	I6	I7	I8	I9	I10	Communication Clarity Index
Engineering Manager (n=4)	2.5	3	1.75	1.5	3	2.35
Product Manager (n=4)	4	4.25	2.25	3.75	2.25	3.3
Staff Engineer (n=3)	4	4.33	3	3	4	3.67

Table 8. *Communication Clarity Index per Role: Item Breakdown.*

5.4 Governance Maturity Results

5.4.1 Overall Governance Maturity Results

Governance Maturity Survey items	Index
I4: My organization has explicit criteria for determining when AI systems can make or suggest decisions autonomously	2.18
I11:When AI-supported decisions have led to issues, responsibility for outcomes is clearly assigned within my organization	2.18
I13: My organization maintains transparent records explaining how AI-assisted decisions are made.	1.64
I14: There are formal escalation procedures when AI-assisted recommendations conflict with human judgment.	2.18
I15: The introduction of AI decision support has strengthened overall accountability across leadership roles.	2.64
Overall Governance Maturity Index	**2.16**

Table 9. *Governance Maturity Index Table*.

Across all respondents, the overall Governance Maturity Index was 2.16, indicating a tendency toward disagreement that the organization maintains clear criteria, transparent records, explicit responsibility pathways, and formal escalation procedures for AI-assisted decisions.

5.4.2 Governance Maturity by Leadership Role

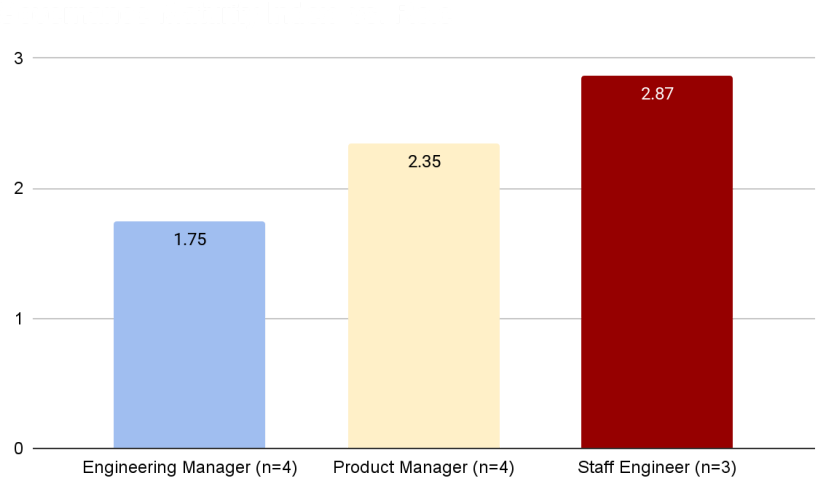


Figure 4. *Governance Maturity Index per Role Results Chart.*

Engineering Managers report the lowest value at 1.75, indicating a stronger tendency toward disagreement. Product Managers report a higher value of 2.35, indicating a mild tendency toward disagreement. Staff Engineers show the highest value at 2.87, a score that sits close to neutral.

Role	I11	I4	I13	I14	I15	Governance Maturity Index
Engineering Manager (n=4)	1.5	3.25	1.25	1.25	1.5	1.75
Product Manager (n=4)	2.5	2.25	1.75	2.25	3	2.35
Staff Engineer (n=3)	2.67	2.67	2.00	3.33	3.67	2.87

Table 10. *Governance Maturity Index per Role: Item Breakdown.*

5.5 Correlation Analysis Results

To complement the descriptive analysis and explore relationships among the three constructs, a non-parametric analysis was conducted using Spearman's rho (p). Given the ordinal nature of the Likert-scale responses and the small sample size, Spearman's correlation is an appropriate method for assessing monotonic associations between constructs without assuming normality.

Construct	Role Overlap	Communication Clarity	Governance Maturity
Role Overlap	1	0.21	−0.07
Communication Clarity	0.21	1	0.33
Governance Maturity	−0.07	0.33	1

Table 11. *Construct Correlation Matrix*

The correlations show that Role Overlap and Communication Clarity have a weak positive relationship (ρ = 0.21). This value indicates only a minimal association between the two constructs. Likewise, the relationship between Role Overlap and Governance Maturity is minimal and slightly negative ($\rho = -0.07$), indicating no meaningful association between these two constructs. In contrast, Communication Clarity and Governance Maturity show the strongest association among the three constructs ($\rho = 0.33$). Although still weak, this positive correlation reflects the highest degree of association observed in the dataset.

6. Discussion

The results from the survey provides a descriptive view of how GenAI is reshaping day-to-day leadership practices in the software engineering industry. By organizing the 15 items into three constructs—Role Overlap, Communication Clarity, and Governance Maturity—the analysis reveals both detailed distribution patterns and composite indices summarizing perceptions across leadership roles and sectors.

6.1 Interpretation

Role Overlap fell between the two constructs with an overall index of 2.81, reflecting a slight but noticeable shift in decision boundaries and autonomy. Respondents reported increased ambiguity regarding who holds decision ownership, as well as changes in how decisions are escalated and how accountability is distributed. Although not uniformly negative, these patterns point to a reconfiguration of leadership responsibilities driven by the integration of AI content into everyday work.

Communication Clarity produced the highest overall index (3.05), indicating that GenAI-generated summaries, reports, and artifacts are viewed as supportive of clearer communication among leadership roles. Respondents generally agreed that AI tools enhance cross-functional information flow, even though trust in AI-generated content and the reduction of direct interaction scored more modestly.

By contrast, Governance Maturity received the lowest overall index (2.16), suggesting that organizations lack consistent structures for AI-assisted decision-making. Scores were particularly low for responsibility assignment when AI-supported decisions lead to issues and for explicit criteria defining when AI systems may act autonomously. These findings indicate that governance maturity has not kept pace with the expanding presence of AI artifacts in leadership workflows.

Taken together and analyzing the correlations between the constructs, it is evident the weak association between Role Overlap and Communication Clarity ($\rho = 0.21$) suggesting that ambiguity in decision boundaries introduced by GenAI does not inherently impair or improve communication among leadership roles. This suggests that communication-related perceptions might benefit from GenAI independently of structural clarity. Conversely, the modest relationship between Communication Clarity and Governance Maturity ($\rho = 0.33$) indicates that teams perceiving clearer AI-enabled communication may also feel more confident about emerging governance practices. This aligns with literature on AI transparency and cross-functional coordination, which emphasizes the role of shared artifacts in stabilizing organizational processes. The negligible relationship between Role Overlap and Governance Maturity ($\rho = -0.07$) further highlights that governance development is not directly mitigating ambiguity in leadership boundaries, pointing to a mismatch between AI-driven integration and the evolution of accountability mechanisms.

Role Analysis

Across the three leadership constructs, respondents showed distinct patterns that reflect how different roles experience the influence of GenAI-generated artifacts. EMs reported the highest Role Overlap index (3.5), indicating that they perceive GenAI as contributing to blurred boundaries in decision ownership and autonomy. This aligns with their hybrid position between organizational alignment and technical oversight. PMs scored lower (2.2), suggesting that GenAI does not affect the division of responsibilities between product definition and engineering decision support. Staff/Technical Leaders showed a near neutral score (2.8), reflecting moderate sensitivity to GenAI outputs that interact directly with architectural reasoning and technical authority.

For Communication Clarity, consistent perceptions were reported among the surveyed leadership roles, with scores ranging around the 2.35-3.67, showing a consistent view that GenAI does improve communication clarity with an overall positive construct mean of 3.05.

Respondents indicated that summaries, automated reporting, and AI-generated documentation improved coordination and context sharing, even when broader structural or accountability questions remained unresolved.

In contrast, Governance Maturity displayed the lowest scores across the dataset. Among the three primary leadership roles, indices ranged from 1.75 for EMs to 2.87 for Staff Engineers, close to the overall construct score of 2.16. Participants consistently noted a lack of clear criteria for AI-supported decisions, limited transparency regarding how AI recommendations are produced, and an absence of formal escalation procedures when AI outputs conflict with human judgment. This indicates that while GenAI-driven tools are already integrated into workflows, organizational governance structures have not yet adapted to support them.

Correlation Analysis

The Spearman correlation matrix revealed three relevant relationships. The strongest association was between Communication Clarity and Governance Maturity ($\rho = 0.33$), suggesting that improvements in AI-mediated communication may be modestly related to perceptions of stronger governance. Role Overlap and Communication Clarity showed a weak positive correlation ($\rho = 0.21$), indicating that increases in boundary ambiguity do not necessarily impede coordination. Role Overlap and Governance Maturity exhibited a weak and slightly negative association ($\rho = -0.07$), reflecting minimal alignment between perceived ambiguity and the presence of formal governance structures.

Overall, the results present a consistent pattern:

1. Leadership boundaries are already shifting, with EMs reporting the clearest increases in overlap.
2. GenAI improves communication efficiency, as reflected in the highest construct score.
3. Governance structures lag behind GenAI technological adoption, producing the lowest scores.

Taken together, these findings suggest that GenAI enhances coordination while simultaneously exposing gaps in role definition, responsibility assignment, and governance maturity. The coexistence of relatively high communication clarity and low governance maturity indicates that organizations may realize short-term coordination benefits even as deeper structural uncertainties persist. However, the correlation results show only weak relationships among the constructs, and sector differences remain inconclusive, meaning that the evidence supports some aspects of this thesis but does not yet demonstrate universal or strongly coupled effects.

6.2 Unexpected outcomes.

There is a low correlation between Role Overlap and Governance , which suggests Governance Maturity aren't yet addressing role overlap effectively. While a root cause is beyond the scope of this thesis, a possible cause might be current frameworks focus on compliance, not boundary definition.

Another alternative interpretation is that perceived role overlap might stem from organizational culture or agile practices rather than AI itself. This is plausible since both agile and AI adoption emphasize autonomy and collaboration. A further confounding factor is the post-pandemic shift toward widespread remote work, which may have already been eroding traditional role boundaries as organizations adapted workflows, communication channels, and responsibilities. However, responses explicitly referencing AI-generated artifacts such as decision summaries or automated reports indicate that, while these broader organizational dynamics may contribute, a portion of the observed overlaps is specifically linked to AI-mediated work processes.

6.3 Rebuttals

A common criticism is that the sample size (n = 11) is too small to support industry-level generalization. This is acknowledged. However, the intent of this study is explicitly exploratory. The goal is to identify early directional patterns that can be compared against existing literature on AI-mediated role diffusion, communication shifts, and governance gaps—not to produce population-level estimates. Even with a small sample, the trends observed are consistent with peer-reviewed research, providing preliminary but meaningful empirical signals.

Another possible critique is that improved Communication Clarity appears inconsistent with reported role ambiguity. The results demonstrate that these can coexist: AI-generated summaries and reports enhance surface-level information flow, but they do not resolve deeper questions about decision ownership or accountability. Thus, clearer communication does not automatically imply stronger structural alignment, and the findings reflect a distinction between operational clarity and organizational clarity.

6.4 Counterarguments

An alternative interpretation is that perceived role overlap could be driven by cultural or methodological factors—such as cross-functional agile practices—rather than by AI adoption itself. While this is plausible, the survey questions explicitly reference AI-generated artifacts, autonomy shifts, and AI-mediated escalation processes. Participants responded in ways that link the observed ambiguity to AI-related changes rather than to general agile collaboration dynamics. This suggests that AI is perceived as a contributing factor distinct from existing cultural norms. Another alternative explanation is that weak governance maturity (lowest construct index) may reflect organizational growth stage or leadership maturity rather than gaps introduced by GenAI. However, the items measuring governance maturity refer specifically to AI-assisted decision processes, criteria for autonomous system action, transparency of AI-influenced decisions, and accountability when AI outputs lead to issues. Responses therefore point toward a governance gap specific to AI integration rather than a general structural weakness.

6.5 Qualifiers

These results should be interpreted within the context of organizations that are in the early stages of adopting GenAI tools into leadership workflows. The patterns observed—slight role overlap, weak governance maturity, improved communication—represent early signals rather than fully matured organizational outcomes.
The study focuses specifically on EMs, PMs, and Staff Engineers as representatives of Technical Leaders. The results describe perceptions at this leadership layer and should not be generalized to developer-focused roles or to entire organizational structures. While the survey was designed to isolate GenAI-related perceptions, the self-reported nature of the data means that causal claims cannot be established. The correlations observed reflect associations between constructs rather than directional or causal relationships.

6.6 Threats to validity

Several methodological considerations may influence the findings. From a construct validity perspective, relying on self-reported Likert-scale assessments introduces common method bias, and respondents may interpret constructs differently depending on organizational processes, compliance expectations, cultural norms, or their level of exposure to GenAI tools, which affects how terms such as “AI-generated artifacts” or “autonomy shifts” are understood. Internal validity is also constrained by factors unrelated to AI, including existing leadership dynamics, team maturity, restructuring efforts, and digital transformation initiatives. The post-pandemic shift to remote and hybrid work further complicates interpretation, as many organizations adjusted workflows and responsibilities in ways that may blur role boundaries independently of AI. External validity is limited by the small and uneven distribution of participants across roles and sectors, restricting the generalizability of the results. Finally, conclusion validity is affected by the absence of triangulation. The study relies solely on survey data, without complementary interviews, artifact analysis, or case studies that could contextualize how leaders experience and operationalize AI-related changes.

7. Conclusion

7.1 Industrial Impact

The findings reveal that GenAI is not only transforming development workflows but also starting to reshape the structure of decision making within software organizations. As AI tools increasingly generate artifacts such as reports, user stories, and design suggestions, the boundaries between Product Management, Engineering Management, and Technical Leadership begin to show early signs of shifting. This shift introduces both opportunities and risks, teams gain speed and informational symmetry, but at the expense of clear ownership and accountability. For the industry, these insights suggest a need to formalize governance maturity that explicitly defines who is accountable for AI assisted decisions and outputs. Without this, organizations risk reinforcing ambiguity that slows delivery, weakens compliance, and undermines psychological safety among leaders.

7.2 Theoretical Impact

From a research standpoint, this study contributes empirical grounding to the emerging discourse on AI mediated organizational structures in software engineering. It extends prior work on sociotechnical systems by demonstrating that GenAI is not merely a technical enabler but could evolve into a governance actor that alters the traditional equilibrium of decision rights. The negative correlation between governance maturity and accountability ambiguity offers a limited indication of accountability ambiguity, suggesting a need for clearer theories linking organizational clarity with reliability and ethical GenAI use. This opens the call for a new theoretical lens that integrates GenAI governance, role theory, and modern software development research, bridging management and engineering disciplines.

7.3 Future Work.

Further research should focus on replicating these results with larger samples and longitudinal designs to trace how governance maturity evolves as AI adoption deepens. Qualitative case studies could explore how leaders renegotiate accountability boundaries in real time, particularly during incidents or decision conflicts involving AI artifacts. Comparative studies across other sectors such as healthcare, government, or manufacturing would help test the domain specificity of these patterns. Additionally, future work could investigate automated governance support tools for example, AI systems capable of tracing decision provenance or highlighting accountability gaps to operationalize the principles identified here.

In sum, this research opens a pathway for both practitioners and scholars to understand and design AI aware governance models that preserve accountability, foster trust, and sustain effective collaboration across software engineering leadership roles.

References

[1] X. Zhou, H. Huang, H. Zhang, X. Huang, D. Shao, and C. Zhong, *"A Cross-Company Ethnographic Study on Software Teams for DevOps and Microservices: Organization, Benefits, and Issues,"* in Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice, May 2022, pp. 1–10. doi: 10.1145/3510457.3513054.

[2] J. Noll, M. A. Razzak, J. M. Bass, and S. Beecham, "*A Study of the Scrum Master's Role,"* in Product-Focused Software Process Improvement, M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, Eds., Lecture Notes in Computer Science, vol. 10611. Cham: Springer International Publishing, 2017, pp. 307–323. doi: 10.1007/978-3-319-69926-4_22.

[3] A. Schmitt, K. Z. Gajos, and O. Mokryn, *"Generative AI in the Software Engineering Domain: Tensions of Occupational Identity and Patterns of Identity Protection,"* arXiv:2410.03571, Oct. 4, 2024. doi: 10.48550/arXiv.2410.03571.

[4] I. Zakharov, E. Koshchenko, and A. Sergeyuk, *"AI in Software Engineering: Perceived Roles and Their Impact on Adoption,"* arXiv:2504.20329, Apr. 29, 2025. doi: 10.48550/arXiv.2504.20329.

[5] R. Hoda, J. Noble, and S. Marshall, *"Developing a grounded theory to explain the practices of self-organizing Agile teams,"* Empirical Software Engineering, vol. 17, no. 6, pp. 609–639, Dec. 2012. doi: 10.1007/s10664-011-9161-0.

[6] S. V. Spiegler, C. Heinecke, and S. Wagner, *"An empirical study on changing leadership in agile teams,"* Empirical Software Engineering, vol. 26, no. 3, p. 41, May 2021. doi: 10.1007/s10664-021-09949-5.

[7] S. V. Spiegler, C. Heinecke, and S. Wagner, *"Leadership Gap in Agile Teams: How Teams and Scrum Masters Mature,"* Zenodo, Dec. 2018. doi: 10.5281/ZENODO.2243113.

[8] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, *"Software development in startup companies: A systematic mapping study,"* Information and Software Technology, vol. 56, no. 10, pp. 1200–1218, Oct. 2014. doi: 10.1016/j.infsof.2014.04.014.

[9] E. Klotins et al., "*A Progression Model of Software Engineering Goals, Challenges, and Practices in Start-Ups,"* IEEE Transactions on Software Engineering, vol. 47, no. 3, pp. 498–521, Mar. 2021. doi: 10.1109/TSE.2019.2900213.

[10] L. Reinhard, *"What should engineering manager leadership roles look like in 2024?"* [Online]. Available: https://www.lenareinhard.com/articles/what-should-engineering-managers-do-tech-leadership-roles. Accessed: Oct. 20, 2025.

[11] A. Kurt, C. R. B. de Souza, A. Begel, and A. Sarma, *"A Mosaic of Perspectives: Understanding Ownership in Software Engineering,"* arXiv:2505.14220, May 2025. doi: 10.48550/arXiv.2505.14220.

[12] R. E. Park, W. B. Goethert, and W. A. Florac, *"Goal-Driven Software Measurement — A Guidebook,"* Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-96-HB-002, Aug. 1996. doi: 10.21236/ADA313814.

[13] A. Mastropaolo, C. Escobar Velásquez, and M. Linares Vásquez, *"From Triumph to Uncertainty: The Journey of Software Engineering in the AI Era,"* in Proceedings of the 47th International Conference on Software Engineering (ICSE '25), 2025, pp. 1–21. doi: 10.1145/3709360.

[14] C. Wohlin, *Experimentation in Software Engineering,* Springer, 2012, p. 159. doi: 10.1007/978-3-642-29044-2.

[15] C. Spearman, *The Proof and Measurement of Association Between Two Things,* The American Journal of Psychology, vol. 15, no. 1, pp. 72–101, 1904. doi: 10.2307/1412159.

[16] S. M. Strachan, M. S. Fortier, M. G. Perras, and C. Lugg, *"Understanding variations in exercise-identity strength through identity theory and self-determination theory,"* International Journal of Sport and Exercise Psychology, vol. 11, no. 3, pp. 273–285, 2013. doi: 10.1080/1612197X.2013.804286.

[17] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps*, IT Revolution Press, Portland, OR, 2018, pp. 42–44.

[18] B. J. Biddle, *Role Theory: Expectations, Identities, and Behaviors*, Academic Press, New York, 1979.

[19] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*, O'Reilly Media, Sebastopol, CA, 2016.

[20] C. Fournier, *The Manager's Path: A Guide for Tech Leaders Navigating Growth and Change*, O'Reilly Media, Sebastopol, CA, 2017.

[21] M. Cagan, *Inspired: How to Create Tech Products Customers Love*, Wiley, Hoboken, NJ, 2017.

[22] W. Larson, *Staff Engineer: Leadership Beyond the Management Track*, Systems Partial Press, 2021.

[23] R. Bommasani, D. A. Hudson, A. Adeli et al., *"On the Opportunities and Risks of Foundation Models,"* Center for Research on Foundation Models, Stanford University, Stanford, CA, 2021, pp. 8-14 .

Appendix

Additional Data Sources

This empirical review was informed by a set of contemporary research papers accessed through digital libraries including IEEE Xplore and the ACM Digital Library. While the primary contribution of this thesis is the original survey data, these external sources helped to elaborate this study to contextualize the emerging patterns observed in leadership perceptions of GenAI. The selected works focus on themes directly relevant to this thesis—role reinterpretation, identity dynamics, perceived AI functions within engineering contexts, and the organizational implications of AI-supported decision processes.

1. Generative AI in the Software Engineering Domain: Tensions of Occupational Identity and Patterns of Identity Protection: Provides evidence on how engineering professionals respond to AI-driven shifts in autonomy and competence, offering insight into the identity work and role negotiation processes that underlie leadership boundary changes.
2. LLMs Integration in Software Engineering Team Projects: Roles, Impact, and a Pedagogical Design Space for AI Tools in Computing Education: Examines how GenAI tools such as ChatGPT and GitHub Copilot shape teamwork, role expectations, and team dynamics in student software engineering projects, introducing the idea of GenAI acting as educator, peer, and assistant within teams. This work reinforces the notion that AI can function as a quasi-team member, influencing collaboration patterns and perceived responsibilities rather than merely serving as a passive coding tool
3. AI in Software Engineering: Perceived Roles and Their Impact on Adoption: Presents a taxonomy of "Support" versus "Expert" AI roles, useful for interpreting leadership perceptions of decision authority, human oversight, and accountability in AI-mediated workflows.
4. From Triumph to Uncertainty: The Journey of Software Engineering in the AI Era : Emphasizes the enduring importance of human judgment in leadership, decision-making, and negotiation processes, reinforcing the thesis findings regarding the limits of AI delegation and the need for explicit governance structures.

These papers do not directly examine communication clarity or accountability maturity in isolation; instead, they provide conceptual grounding on how AI reshapes role expectations, decision authority, and leadership identity. Their insights informed the construction of survey dimensions and supported the interpretation of results concerning role overlap, governance maturity, and cross-functional coordination in AI-enabled organizations.

Survey Administration Planning

The survey was administered online using Google Forms Platform, in an asynchronous way in the following timeframe: 27 October to 31st October 2025. Each participant received an invitation link to participate, with a clear description of the study purpose, confidentiality and assurance. Respondents were able to review the survey before any submission. To ensure bias minimization, item order was randomized as well as construct categories hidden, the description at the beginning of the survey mentioned the constructs summarized though. All responses were anonymous and did not contain any Personal Information in order to comply with any regulatory risk mitigation measures.

Data collection finalized on 1st November 2025. All selected respondents finished the survey. The results were submitted via Google Forms link at "Submit". Responses were exported in CSV format and processed using Google sheets for statistical analysis.

Survey Results Links

1. CSV Raw Results
2. Survey Results Summary Spreadsheet
3. Survey Results Breakdown Analysis Spreadsheet

An LLM-Based Framework for Automated Data Model Generation

Christie Sam
Software and Societal Systems
School of Computer Science
Carnegie Mellon University

Abstract

The manual translation of natural language requirements into technical database schemas remains a persistent bottleneck in software engineering, characterized by high costs and susceptibility to human error. This research evaluates a novel Large Language Model (LLM) framework designed to automate this process using a two-step "Analyst Engineer" prompt-chaining architecture. The study tested the framework against ten diverse user stories covering both Online Transaction Processing (OLTP) - systems for daily operational transactions and Online Analytical Processing (OLAP) - systems for complex data analysis and reporting domains, using a blind comparative evaluation with five expert data architects to benchmark AI-generated schemas against human-created "Gold Standards". The results demonstrate a significant difference, where the framework achieved a 100% expert preference rate for OLTP scenarios but mixed results (60% preference) for OLAP tasks due to semantic ambiguities. Despite these variations, the framework successfully reduced the aggregate estimated engineering time-to-production by approximately 40%. These findings suggest that modern LLMs can function as "interpretive co-designers" that enforces best practices, shifting the data architect's role from drafting to high-value verification.

1. Introduction

In modern software engineering, the translation of abstract business requirements into a concrete, technical system design is one of the most critical and error-prone phases of development. This challenge is particularly prominent in data modeling, where the manual conversion of human-defined needs into formal database schemas represents a significant bottleneck. This process is not only time-intensive, relying on the specialized and costly expertise of data architects, but it is also highly susceptible to human misinterpretation. Ambiguities in initial requirements can lead to flawed data models, introducing foundational errors that become exponentially more difficult and expensive to remediate once application code and data pipelines are built on them. Research into agile requirements engineering highlights this as a persistent challenge, specifically dealing with ambiguous, incomplete or poorly translated requirements [11]. This ambiguity is the primary source of error that this thesis aims to address.

The emergence of Large Language Models (LLMs) presents a transformative opportunity to address this challenge. With demonstrated capabilities in understanding natural language and generating complex, structured code [7], LLMs are rapidly being adopted as co-development tools across the industry [9]. This potential motivates the central claim of this thesis: that a systematic, two-step prompt-chaining framework - which first translates high-level UI stories into granular data-engineering (DE) stories, and then translates those DE stories into a technical schema - can effectively and safely accelerate the data modeling process for enterprise applications without sacrificing the quality or integrity of the resulting database schema

This research investigates the design and evaluation of such a framework, specifically one utilizing the Gemini Pro 2.5 API. The scope is intentionally focused to ensure the method is both achievable and impactful: the framework is designed to process English-language user stories following a standard agile format. A key qualifier is that the framework explicitly branches its logic based on the problem domain, applying a normalized Third Normal Form (3NF) model for transactional (OLTP) requests and a dimensional star-schema for analytical (OLAP) requests. This work acknowledges a “schema-first” design workflow and qualifies its claim by stating that the framework’s output serves as a high-quality first draft that requires final validation by a human expert.

The core of this thesis is to answer the primary research question: “How well does an LLM-based framework perform compared to an expert data architect in translating user stories into relational database schemas?”.

This question is decomposed into three specific objectives:

1. To measure Correctness: How accurate is the LLM-generated schema (e.g., tables, columns, keys, relationships) when compared against a “gold standard” model?
2. To evaluate Efficiency: To what extent does the framework verifiably reduce the time and cognitive load required for a human expert to produce a final, production-ready design?
3. To assess Design Quality: Does the generated schema adhere to established database design principles, such as normalization, and is it deemed clear, maintainable, and robust by domain experts?

To answer these questions, we employ a mixed-methods research design [4]. This approach allows for the validation of both the quantitative accuracy of the generated schemas and their qualitative, subjective design quality as judged by human experts. The framework's effectiveness will be benchmarked using quantitative metrics (Precision, Recall, F-1) against a novel dataset developed for this study. This benchmark was constructed in a two-step, manual process: first, manually authoring a set of ten high-quality UI user stories (as detailed in Table 4.1), and second, manually creating an expert 'gold standard' schema for each story. The framework's practical utility is assessed through structured qualitative reviews from experienced data architects. This dual approach provides the logical foundation to connect the framework's output to the thesis's conclusions about its utility.

The primary contributions of this work are threefold:

1. A novel two-step prompt-chaining framework (the 'Analyst-Engineer' pipeline) implemented in Python, which programmatically separates the task of requirement interpretation from technical generation.
2. A set of expert-driven, few-shot prompt templates that successfully guide an LLM to generate two distinct, high-quality architectural patterns: a normalized 3NF model for OLTP and a dimensional star-schema for OLAP
3. A comprehensive, mixed-methods evaluation that provides a model for assessing AI-driven design tools in terms of correctness, efficiency, and professional-grade quality.

2. Background

To understand the claim and methodology, one must be familiar with the foundational concepts of database theory, the specific software engineering processes being automated, and the underlying technologies enabling this automation.

2.a Foundational Concepts in Database Design

The "quality" of a data model is not subjective; it is grounded in decades of computer science theory. These foundational concepts are understood by virtually all data professionals and provide the well-established, practical criteria used to accept high-quality and correct models. The two most important concepts for this research are the Relational Model and the Entity-Relationship Model.

- The Relational Model and Normalization: Introduced by E.F. Codd [1], the relational model provides the mathematical foundation for modern databases. Its central concept is normalization, a systematic process for organizing data in tables to reduce redundancy and improve data integrity. For example, a "normalized" schema avoids storing the same piece of information (like a customer's name) in multiple places. Therefore, in this thesis, Codd's principles of normalization serve as the foundational criteria for evaluating what constitutes a "correct" and "high quality" schema.
- The Entity-Relationship (ER) Model: Proposed by Peter Chen [2], the ER model is a high-level conceptual data model. It provides a standardized graphical way to represent entities (e.g., 'Customer', 'Order') their attributes (e.g., 'Name', 'Date'), and the relationships between them (e.g., 'a Customer places an Order'). An ER Diagram (ERD) is a primary output of the data modeling process and a key target for the LLM framework , as it serves as the logical blueprint before any SQL code is written.

2.b Operational vs. Analytical Workloads (OLTP and OLAP)

A critical distinction in modern data engineering - and a primary variable in this study's results - is the difference between operational and analytical workloads. Industry standards dictate a different modeling technique for each, which this thesis strictly observes.

- **Online Transactional Processing (OLTP)**: These systems support day-to-day operational applications (e.g., e-commerce purchasing, banking ledgers). The primary goal is fast, atomic transaction processing and data integrity. The standard modeling technique for OLTP is Third Normal Form (3NF), which prioritizes the elimination of data redundancy to prevent anomalies during updates.
- **Online Analytical Processing (OLAP)**: These systems support complex data analysis, reporting and business intelligence (e.g., market attribution, risk analytics). The primary goal is query performance for heavy read operations. The standard modeling technique for OLAP is the Dimensional Model (or Star Schema) popularized by Ralph Kimball. Unlike 3NF, this approach intentionally denormalizes data into "Fact" tables (numeric metrics) and "Dimension" tables (descriptive attributes) to simplify complex queries.

Relevance to Findings: This distinction is central to the thesis because the experimental results (discussed in the Results section) reveal a significant divergence in how the LLM framework handles these two architectures. While the framework achieved a 100% preference rate for 3NF/OLTP schemas, it showed mixed results (60%) when generating Dimensional/OLAP schemas. This suggests that while LLMs excel at the strict logic of normalization, they struggle with the semantic ambiguity often found in analytical modeling.

2.c Technologies and Processes

This thesis operates at the intersection of agile software development and generative artificial intelligence.

- Agile Development and User Stories: The primary input for the framework is the user story. In agile development, a user story is a brief, simple description of a feature told from the perspective of the person who desires it [3]. A common format is: "As a <type of user>, I want <some goal> so that <some reason>." In practice, the task of developing a data model is now frequently derived from a collection of user stories. This research uses "well-structured user stories" as its starting point, making the work of Cohn [3] a key piece of context for defining the framework's input requirements.
- Large Language Models (LLMs) and Prompt Engineering: An LLM is a deep learning model trained on vast amounts of text, enabling it to understand and generate human-like language and structured code. This thesis uses Gemini Pro 2.5. The framework's core mechanism is not re-training the model, but rather prompt engineering - the practice of designing a sophisticated input (a "prompt") that guides the model to perform a specific task. This approach is based on the concept of "few-shot learning" [8], which demonstrates that a large model can learn a new task at inference time simply by being given a few examples in its prompt. The capabilities demonstrated in this paper [8] provides the technical foundation for this thesis's prompt-based approach.

3. Related Work

This thesis is built on three distinct but related research areas: AI in software engineering, natural language processing for database querying, and automated schema generation.

3.a State of the Art

The application of AI to software engineering (AI4SE) is a rapidly expanding field. A foundational premise for this thesis is the demonstrated feasibility of using LLMs to generate functional code. Research by Chen et al. [7] was among the first to rigorously evaluate LLMs trained on code, proving their capability to handle the complex syntax and logic of programming. This set the stage for tools like GitHub Copilot, which are now widely adopted [9].

A more specific and mature area of research is Text-to-SQL, which focuses on translating a natural language question (e.g., "who were our top 10 customers last month?") into a database query (a SQL SELECT statement). The introduction of the Spider dataset [6] provided a large scale, cross-domain benchmark that has driven significant progress in this area. This work serves as an influential model for this thesis's quantitative methodology - it establishes a precedent for creating a benchmark dataset (language-to-structure pairs) and using it to measure accuracy.

Most directly related is the emerging field of Text-to-Schema generation. Recent studies, such as the work by Salem et al. [11] have explored using earlier-generation deep learning models like BERT (Bidirectional Encoder Representations from Transformers) and T5 (Text-to-Text Transfer Transformer). These (primarily encoder-based or encoder-decoder) models showed promise in extracting entities, attributes, and relationships but typically required extensive, domain-specific fine-tuning on large, labeled datasets. This work provides crucial Context, demonstrating that this research topic is part of an active and relevant academic conversation.

3.b Gaps and Open Questions

Despite this progress, significant gaps remain, which this thesis directly addresses:

1. Text-to-SQL vs. Text-to-Schema: The vast majority of research (e.g., [6]) focuses on querying existing databases, not creating them. Generating a query is a constrained task, whereas generating a normalized, maintainable, and robust schema from scratch is a conceptual design challenge of much higher complexity.
2. Model and Method Novelty: Many existing Text-to-Schema approaches [11] rely on older model architectures (like BERT or T5) that require extensive, domain-specific fine-tuning. The capabilities of modern, generalist, few-shot learners [8] like Gemini Pro 2.5 to perform this complex design task via prompt engineering alone are largely unexplored.
3. Lack of Qualitative and Efficiency-Based Evaluation: Existing work tends to focus exclusively on quantitative correctness (e.g., "was the column name correct?"). It often fails to ask the more critical questions for industrial adoption: "Is the design good?" "Is it maintainable?" and "Does this tool actually save time for a senior architect?"

3.c Position of this Work

This thesis fills these gaps by proposing a novel framework and evaluation methodology. It moves beyond Text-to-SQL to tackle the more complex Text-to-Schema problem. Instead of

fine-tuning, it leverages the in-context learning capabilities of a state-of-the-art model (Gemini Pro 2.5) through a sophisticated, programmatic prompt engineering system.

Most importantly, its contribution is methodological. By employing a mixed-method design [4] that combines a quantitative benchmark [6] with a qualitative, expert-driven evaluation [5], this thesis provides a holistic assessment. It is not content to prove that an LLM can generate a schema; it aims to prove that an LLM-based framework can produce a *high-quality, well-designed* schema that accelerates the workflow of an expert data architect, thereby addressing a core, validated problem in modern software development.

4. Methodology

This section details the research methodology used to evaluate the thesis claim. It describes the overall research design, the framework's architecture, the procedures for data collection and the methods for data analysis.

4.a Research Design and Justification

To validate the claim that an LLM-based framework can both effectively accelerate data modeling and produce high-quality results, this thesis uses a mixed-methods research design. This approach is essential because the claim includes objective, quantitative performance and subjective, qualitative design quality. As argued by Easterbrook et al. [4], combining both quantitative and qualitative methods provides a more robust and convincing conclusion in software engineering research, allowing qualitative insights to explain and contextualize quantitative results.

The methodology is structured in two sequential phases:

1. A Quantitative Experiment to objectively measure the frameworks' accuracy and correctness against a "gold standard" benchmark.
2. A Qualitative Evaluation to assess the industrial relevance, design quality and practical utility of the framework's output through expert review.

This dual approach provides the evidence (benchmark scores and expert feedback) needed to evaluate the central thesis claim. The design itself is justified by established articles in software engineering research [4], [5], [6].

4.b Framework Architecture and Evaluation Setting

The system at the center of this study is a novel, two-stage LLM framework designed to automate the translation of user stories into database schemas.

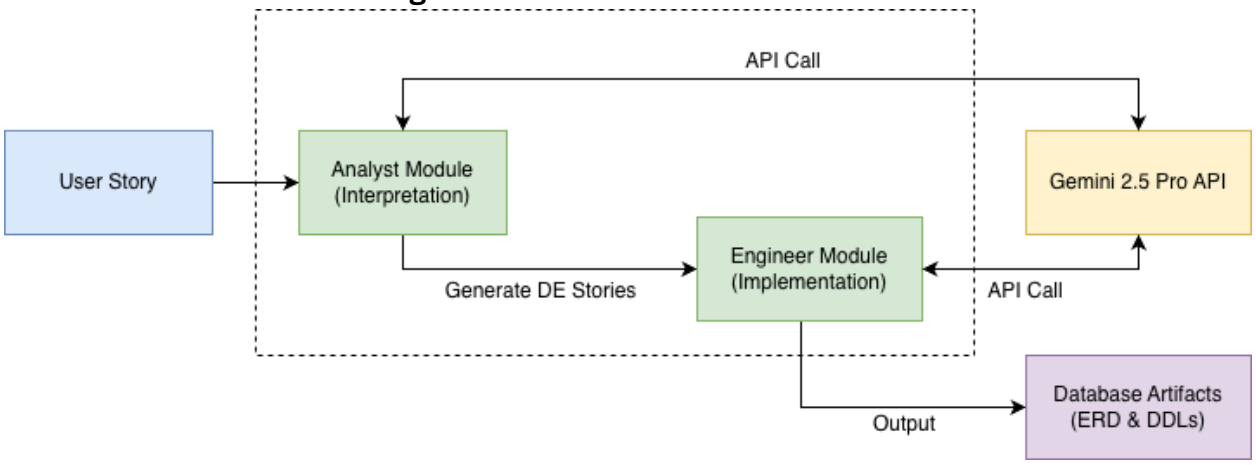
Architectural Rationale and Component Design: The framework utilizes a prompt-chaining architecture that decouples requirement analysis from technical implementation. This separation of concerns is critical to the thesis claim. By isolating the "interpretive" task (understanding ambiguity) from the "generative" task (writing syntax), the framework mitigates the cognitive load on the LLM, reducing the hallucinations and logic error common in single-shot generation. This mimics the real-world workflow where a Business Analyst clarifies requirements before a Data Engineer writes code.
The architecture consists of two custom-written Python modules - the 'Analyst' and the 'Engineer'.

The 'Analyst' module serves as the framework's reasoning engine. Its primary responsibility is to decompose high-level, often ambiguous UI-focused user stories into granular, unambiguous "Data Engineering Stories". This intermediate step is essential because raw user stories typically describe interface behaviors (e.g., "clicking a button") rather than data structures. Direct translation of these UI narratives into SQL often leads to denormalized or incomplete schemas. The Analyst module bridges this semantic gap by inferring the latent entities and relationships required to support the user's goal before any code is written.

The 'Engineer' module functions as the translation and implementation layer. Accepting the structured specifications from the Analyst, its responsibility is to generate the technical artifacts (ERDs and SQL DDL), applying specific architectural patterns - such as Third Normal Form (3NF) for OLTP or Star Schema for OLAP - based on the domain. This module is necessary to ensure technical correctness; by receiving pre-processed, unambiguous requirements, the LLM can focus entirely on enforcing strict normalization rules, naming conventions and constraints without the risk of misinterpreting the underlying business intent.

This architecture mimics the real-world software engineering workflow, where a Business Analyst clarifies requirements before a Data Engineer implements the design, ensuring that the final schema is both technically sound and aligned with business needs.

Figure 4.1: Thesis Framework Architecture



- Process and Standards: The framework operates strictly within a "schema-first" design standard, where the database schema is intentionally designed before application code is written. This is a best practice for ensuring data integrity in large-scale applications and serves as the operational constraint for the 'Engineer' module. The module produces two outputs: a conceptual Entity-Relationship Diagram (ERD) and a logical SQL Data Definition Language (DDL) script.
- Participants: The qualitative evaluation involved a panel of 3-5 experienced data architects or senior data engineers from the industry.

4.c Data Sources and Collection Procedures

4.c.a Phase 1: Quantitative Benchmark Dataset

Data Source: The primary evidence for Phase 1 is a Benchmark Dataset of 10 user stories paired with "gold-standard" schemas, which are a contribution of this research.

Procedure: The dataset creation and experimental procedure follows three steps:

1. **Sourcing**: A benchmark dataset [N=10] of high quality, representative user stories was manually authored by the researcher. This was necessary as a review of public repositories like Zenodo revealed a lack of suitable, complex and domain-diverse stories required to test the framework's capabilities. The authored stories were designed to cover both Online Transactional Processing (OLTP) and Online Analytical Processing (OLAP).

Table 4.1: Benchmark User Story Dataset

	Story	Domain	Type	Description
1.	**Bank Analytics**	Banking	OLAP	As a Risk analytics manager, I want to analyze daily and historical customer transactions across multiple channels (ATM, online banking, credit cards and branch) so that I can detect unusual spending patterns, potential fraud and predict customer churn.
2.	**Marketing Attribution**	Marketing	OLAP	As a Marketing data scientist, I want to measure the performance of digital campaigns across multiple touchpoints (email, web app, social media) so that I can attribute conversions accurately and optimize ad spend
3.	**Ride-Share Analytics**	Ride-Share	OLAP	As a Regional Operations Director, I want to track driver utilization, rider satisfaction and surge pricing events to identify cities with driver shortages or inefficiencies.
4.	**Patient Journey**	Healthcare	OLAP	As a healthcare data analyst, I want to build a longitudinal view of patient journeys across claims, lab tests and prescriptions to identify patterns in chronic disease management and cost drivers.
5.	**Hospitality Loyalty**	Healthcare	OLAP	As a loyalty program manager, I want to analyze hotel bookings, guest satisfaction surveys and loyalty redemptions to predict high-value guests and optimize promotional offers.
6.	**Banking Core System**	Banking	OLTP	As a Banking Operations Officer, I want to record deposits, withdrawals, fund transfers and account updates in real time, so that account balances, statements

				and customer ledgers remain accurate and auditable.
7.	**Retail Order System**	Retail	OLTP	As an Order processing associate, I want to capture customer orders, order line items, payments and shipments, so that customers can track their purchases from order to delivery.
8.	**Clinical Billing System**	Healthcare	OLTP	As a clinic administrator, I want to manage patient appointments, visits and billing records so that each consultation is accurately tracked, invoiced and linked to providers.
9.	**Ride-Share Booking**	Ride-Share	OLTP	As a ride-share platform admin, I want to record rides, driver availability, trip payments and ratings so that each trip's lifecycle can be managed from request to payment.
10.	**Hotel Check-in**	Hospitality	OLTP	As a hotel front-desk agent, I want to manage room bookings, check-ins and guest billing so that reservations, occupancy and invoices are accurately maintained.

2. **Transformation**: The manually authored UI stories (from Table 4.1) are first processed by the 'Analyst' module. This module calls the LLM to decompose the single high-level UI story into a set of granular, data engineering-focused user stories. This intermediate artifact serves as the input for the 'Engineer' module. An example of this transformation is shown in Table 4.2.
3. **Gold Standard Creation**: For each of the transformed user stories, a "gold standard" relational schema was manually created by the author. This manual model adheres to Codd's principle of normalization [1] and Chen's ER modeling conventions [2].
4. **Data Collection**: Each transformed user story (from Step 2) is fed as input to the 'Engineer' framework (the system being tested). The resulting LLM-generated schema (ERD and SQL DDL) will be collected and stored for analysis.

Table 4.2: Illustrative Snippet of 'Analyst' Module Output for User Story 2

Input UI User Story	Output: Generated Data Engineering User Story
As a Marketing data scientist, I want to measure the performance of digital campaigns across multiple touchpoints (email, web app, social media) so that I can attribute conversions accurately and optimize ad spend	As a System Designer, I need to model a **`Campaign`** entity to represent a high-level marketing initiative. As a System Designer, the `Campaign` entity must have attributes such as `campaign_name`, `start_date`, `end_date`, `budget`, and `objective`. As a System Designer, I need to model a **`Channel`** entity to represent the different marketing platforms used, such as 'Email', 'Google Ads', or 'Facebook'.

	As a System Designer, I need to model an **`AdCreative`** entity to store the specific content of an advertisement, with attributes like `creative_name` and `creative_type` (e.g., 'video', 'banner_image', 'text'). As a System Designer, I need to model a **`User`** entity to represent an individual customer or prospect, identified by a unique `user_id`, to track their journey across touchpoints. As a System Designer, I need to model **`InteractionEvent`** as a time-series event stream to capture every user touchpoint with a marketing asset. As a System Designer, each `InteractionEvent` record must contain a `user_id`, a precise `event_timestamp`, and an `interaction_type` (e.g., 'ad_view', 'ad_click', 'email_open'). As a System Designer, each `InteractionEvent` must reference the **`Campaign`**, **`Channel`**, and **`AdCreative`** it is associated with linking user actions back to specific marketing efforts. As a System Designer, I need to model **`ConversionEvent`** as a separate time-series event stream to capture valuable user actions. As a System Designer, each `ConversionEvent` record must contain a `user_id`, a `conversion_timestamp`, a `conversion_type` (e.g., 'purchase', 'signup'), and a `monetary_value` attribute. As a System Designer, I need to model **`CampaignCost`** as time-series data to track daily or hourly ad spend. As a System Designer, the `CampaignCost` data must include attributes for `date`, `spend_amount`, and be linkable back to a specific **`Campaign`** and **`Channel`**.

4.c.b Phase 2: Qualitative Expert Evaluation

Data Source: The data for Phase 2 consists of comparative feedback collected from the expert panel via a structured blind evaluation survey. The survey presents two models for each user story: the AI-generated model (from Phase 1) and the researcher-created “Gold Standard” model.

Procedure: This phase was conducted as a blind comparative study to mitigate rater bias.

1. Sampling: The entire benchmark dataset of 10 user stories was submitted for expert evaluation to ensure coverage across all defined domains.
2. Packet Creation: For each user story, a packet was created containing:
    - The transformed data engineering user story.
    - Model A - AI-generated schema

    - Model B - “Gold Standard” Schema
3. Blinding & Randomization: The assignment of Model A and Model B was randomized for each packet. The experts were not told which model is AI-generated and which is human created.
4. Data Collection: Experts completed the survey, which asked them to::
    - Rate Model A and Model B on the same set of Likert-scale attributes (Correctness, Normalization, Clarity etc.).
    - Identify defects or list specific flaws for the models.
    - Provide a preference: After rating both, they were asked which model they prefer as a starting point .
    - Estimate time to completion: For both models, they estimated the time required to get them to a production-ready state.

This A/B test design allowed for a direct comparison of quality and provided a much more nuanced measure of acceleration.

4.d Analysis Methods

4.d.a Quantitative Analysis

The data from Phase 1 will be analyzed by algorithmically comparing the components of the framework-generated schemas against the “gold-standard” schemas. This benchmark-based evaluation is a standard approach for performance assessment, modeled after influential datasets like Spider for Text-to-SQL research [6].

The primary metrics are:

- Precision: Measure exactness. The fraction of generated schema components (tables, columns, keys) that are correct.
- Recall: Measure completeness. The fraction of “gold standard” schema components that were successfully identified by the framework.
- F-1 Score: The harmonic mean of Precision and Recall, providing a single, balanced measure of accuracy.

4.d.b Qualitative Analysis

The data from Phase 2 will be analyzed using a couple of methods:

1. Statistical Comparison: The Likert-scale ratings for the AI-generated models and the Gold Standard models will be compared using statistical tests to see if there is a statistically significant difference in perceived quality.
2. Time-to-Completion Analysis: The expert estimations for time to production will be compared for both model types. This provides a direct, quantitative measure of the acceleration (or time saving) aspect of the claim.
3. Thematic Analysis: The open-ended feedback (defect lists, comments) will be systematically coded to identify recurring patterns. This approach follows the rigorous method for interpreting qualitative data provided by Braun & Clarke [10]. Themes include “Common Normalization Errors”, “Effective Naming Conventions” or “Reliable Relationship Identification”.

Together, these analysis methods will allow for powerful conclusions, such as “Experts rated the AI-generated models as statistically indistinguishable from the human-created Gold Standard on clarity” or “The AI model reduced the estimated time-to-production by an average of 70% compared to the Gold Standard model”

5. Results

Note: This section presents the aggregate findings, summaries and key illustrative examples from the expert panel review. User story, Gold Standard Schema/ERD, AI-generated schema/ERD in Appendix A and expert survey transcripts in Appendix B are available for detailed review at the end of this document.

This section presents the findings from the mixed-methods evaluation, mapping directly to the research questions of correctness, efficiency and design quality. The data was collected from a blind comparative study where a panel of five (N=5) expert data architects evaluated ten distinct user stories. For each user story, the panel reviewed two schemas: the human created "Gold Standard" (Model A), which represented a literal translation of the requirements, and the "AI Generated" (Model B) schema produced by the LLM framework.

The dataset was balanced to cover two distinct domains:

1. Online Analytical Processing (OLAP): 5 scenarios focusing on data warehousing and reporting.
2. Online Transactional Processing (OLTP): 5 scenarios focusing on operational systems and data integrity.

The results are presented factually below. The interpretation of these findings will follow in Section 6.

5.a Expert Preference and Rationale

The primary and most significant finding was the expert panel's clear preference for the AI-generated schemas. In eight out of ten scenarios, the AI model was selected as the superior starting point for a production-grade system.

A distinct pattern emerged based on the domain. The framework achieved a 100% preference rate in OLTP scenarios, where experts consistently cited its use of modern, robust patterns such as immutable transactional logs, event-sourced status tracking and unified identity models as superior to the traditional designs found in the Gold Standard. In OLAP scenarios, the results were mixed (60% preference), with experts favoring the human design in cases the AI misunderstood specific business definitions (e.g., utilization).

Table 5.1: Summary of Expert Preference and Rationale

Story	Domain	Preferred Model	Rationale for Preference (Expert Summary)
Bank Analytics	OLAP	AI (Model B)	The Gold Standard model fundamentally missed the business requirement. The AI understood the intent behind "detect fraud" and built a fit-for-purpose model with dedicated fraud/analytics tables.
Marketing Attribution	OLAP	Gold Standard (Model A)	The Gold Standard was a clearer, simpler star schema (Kimball-style) better suited for its target audience (BI analysts). The AI's model was deemed too granular and "over-engineered" for the task.

Ride-Share Analytics	OLAP	Gold Standard (Model A)	Critical AI Failure. The AI modeled “trips” but completely missed the concept of “driver utilization” (time online) and independent “surge events”. The Gold Standard was correctly modeled as distinct fact tables.
Healthcare Claims	OLAP	AI (Model B)	The AI correctly modeled the real-world many-to-many relationship between claims and diagnoses. The Gold Standard used an incorrect 1:1 relationship, which would result in data loss for chronic disease analysis.
Hospitality Loyalty	OLAP	AI (Model B)	The AI captured the implicit requirements to “optimize promotional offers” by creating specific offer tracking tables. The Gold Standard completely missed this requirement, making campaign optimization impossible.
Banking Core System	OLTP	AI (Model B)	The AI produced a modern, architecturally superior OLTP design (immutable log, audit trails, ENUMs). The Gold Standard was seen as outdated, high-risk and requiring significant rework to be production-safe.
Retail Order System	OLTP	AI (Model B)	The AI’s design was more scalable and maintainable, correctly anticipating modern eCommerce needs (split shipments, multi-address) that the Gold Standard model missed.
Healthcare Billing	OLTP	AI (Model B)	The AI correctly implemented an itemized billing structure which is essential for insurance claims. The Gold Standard used a single amount field, which is insufficient for clinical operations.
Ride-Share Booking	OLTP	AI (Model B)	The AI utilized a modern “Unified User” model and Event Sourcing for driver availability. The Gold Standard used outdated, separate Rider/Driver tables that created duplicate identities.
Hotel Check-in	OLTP	AI (Model B)	The AI properly normalized room types into a dimension and integrated check-in timestamps into the booking table. The Gold Standard was over-engineered with a separate check-in table and used denormalized strings.

5.b Perceived Quality (Likert Scale)

The expert panel rated both blinded models on five key attributes of design quality. The aggregate score shows that the AI-generated framework was rated equal to or higher than the Gold Standard in most categories.

The most significant gap was in “Maintainability” (4.6 vs 3.9) and “Clarity” (4.8 vs 4.4). Experts consistently rated the AI higher because it included production-ready infrastructure (comments, audit timestamps, constraints) that the Gold Standard often omitted. However, the "Completeness" score for the AI was negatively impacted by the Ride-Sharing OLAP failure, where it scored a 2/5.

Table 5.2: Aggregate Expert Quality Ratings (Mean Score / 5.0)

Attribute	Gold Standard (Mean)	AI Framework (Mean)	Finding
Correctness	4.1	4.4	AI Model Wins
Completeness	4.0	4.3	AI Model Wins
Normalization	4.8	4.9	AI Model Wins
Clarity	4.4	4.8	AI Model Wins
Maintainability	3.9	4.6	AI Model Wins

These aggregate scores are a summary of the detailed expert ratings. One of the examples of the Likert score breakdown and rationale of one user story is available in Appendix B, Part B.2.

5.c Time-to-Production (Acceleration)

This metric measures the estimated engineering time required to get each model to a production-ready state. To ensure objectivity, this assessment was conducted using a blind evaluation protocol: expert estimators were not informed which model was generated by the AI and which was the human-created “Gold Standard”, eliminating source bias from their estimates.

The result shows a significant net acceleration, though with high variance in OLAP cases.

- OLTP Scenarios: The AI saved time in 5 out of 5 cases, averaging a ~60% reduction in effort.
- OLAP Scenarios: The AI saved massive time in complex cases (Banking, Hospitality) but increased the workload in the Ride-Sharing (requiring a complete redesign).

Overall, the framework reduced the estimated total engineering effort from ~56 hours (Gold Standard) to ~34 hours (AI Framework).

Table 5.3: Expert Time-to-Production Estimates (in Hours)

Story	Gold Standard (Est.)	AI Framework (Est.)	Time Saved by AI
Banking Analytics	8.0 - 12.0 hours	2.0 - 3.0 hours	+ 7.5 hours
Marketing Attribution	0.5 hours	1.0 hour	- 0.5 hours
Banking Core System	6.0 - 8.0 hours	3.0 - 4.0 hours	+ 3.5 hours

Retail Order System	0.5 hours	0.5 - 0.75 hours	- 0.25 hours
Ride-Share Analytics	4.0 - 5.0 hours	8.0 - 12.0 hours	- 5.5 hours (Loss)
Healthcare Claims	6.0 - 8.0 hours	4.0 - 5.0 hours	+ 2.5 hours
Hospitality Loyalty	6.0 - 8.0 hours	4.0 - 5.0 hours	+ 2.5 hours
Healthcare Billing	6.0 - 8.0 hours	2.0 - 3.0 hours	+ 4.5 hours
Ride-Share Booking	6.0 - 8.0 hours	2.0 - 3.0 hours	+ 4.5 hours
Hotel Check-in	5.0 - 6.0 hours	2.0 - 3.0 hours	+ 3.0 hours
TOTAL	**~56.0 hours**	**~34.0 hours**	**~ 40% reduction**

These time estimates are derived from the expert panel's detailed defect analysis. One of the defects from the defect list and the corresponding time-to-production breakdown for a user story is located in Appendix B, Part B.1.

5.d Thematic Analysis of Defect Types

The most significant qualitative finding emerged from why models were rated as flawed. The following themes were synthesized from a qualitative analysis of the raw expert defect lists, which are transcribed in Appendix B, Part B.1.

1. Gold Standard Defects (Structural): The human-created models frequently suffered from structural or architectural flaws. Common themes included "outdated design patterns" (e.g., separate Rider/Driver tables), "missing domain logic" (e.g., 1:1 relationship for medical diagnoses) or "technical dead needs" (e.g., denormalized room types).
2. AI Framework Defects (Refinement vs. Semantic):
    a. In 9 out of 10 cases, AI defects were minor and additive (e.g., "Missing CHECK constraints", "Add currency field", "Missing specific dimension").
    b. In 1 case (Ride-Share OLAP), the AI suffered a semantic failure, misinterpreting the business definition of "utilization" as "trip count" rather than "time online".

Table 5.4: Thematic Defect Analysis

Defect Theme	Gold Standard Schema Defects	AI Framework Defects
Structural / Architectural	Common (8 / 10 Cases). Described as "risky", "outdated", "complex transfer handling", "balance/ledger consistency issues".	Rare (1 / 10 Cases). Major Failure in Ride-Share OLAP only
Missing Requirements	Common (7 / 10 Cases). Described as "missed 60% of stated requirements", "lacks analytics attributes", "missing audit trail", "no split-shipment support"	Occasional (2 / 10) e.g., Missing Utilization Metrics (Ride-share) and demographics (Marketing)

Refinement / Additive Gaps	Rare.	Very Common (10 / 10 Cases). Described as “missing constraints”, “add indexes”, “add audit timestamps”, “expand ENUMs”.

5.e Illustrative Example: Interpretation vs. Translation

The “Bank Analytics” User Story provides a clear, illustrative example that synthesizes all the findings above. (The full schemas for this story is available in Appendix A.1)

User Story: *“As a Risk analytics manager, I want to analyze daily and historical customer transactions across multiple channels (ATM, online banking, credit cards and branch) so that I can detect unusual spending patterns, potential fraud and predict customer churn.”*

The Gold Standard model literally translated the nouns in the story, creating dimensions for dim_product, dim_account and dim_location. However, it failed to address the stated business goal of “detect fraud” or “predict churn”, providing only a boolean is_fraudulent flag.

The AI framework interpreted the business intent. It missed some of the literal dimensions (like Product and Location) but correctly inferred and added the complex components needed to actually solve the problem.

This single “snippet” explains the aggregate results: the AI model was preferred (Table 5.1), had higher "Completeness" (Table 5.2), saved 8-12 hours of redesign work (Table 5.3) and avoided the “Missing Requirements” defect (Table 5.4). Table 5.5 provides a high-level comparison, but the detailed expert analysis (Appendix B.1) referenced the full artifacts. For instance, the Gold Standard’s fct_Transaction table (Listing A.1.1) was compared against the AI’s addition of a dedicated fct_Fraud_Alerts table (Listing A.1.2).

Table 5.5: Illustrative Snippet (Banking Analytics - User Story 1)

Model Feature	Gold Standard (Model A)	AI Framework (Model B)
Fraud Model	BOOLEAN is_fraudulent in fct_Transactions	fct_Fraud_Alerts (A new, dedicated fact table)
Churn Model	(Not Addressed)	NUMERIC churn_probability (Added to dim_Customer)
Analytics Model	(Not Addressed)	INTEGER customer_risk_score (Added to dim_Customer)
Completeness	Literally complete, but misses business goals	Infers latent business requirements successfully.
Expert Verdict	“Missed 60% of the stated requirements”	“Directly addresses ALL components of the user story”

The expert’s full rationale for this verdict is available in Appendix B.1, Part B.3.

5.f Qualitative Analysis of Design Artifacts (ERDs)

A key component of the expert evaluation was a deep analysis of the Entity-Relationship Diagrams (ERDs)

1. For the OLTP System: The AI's ERD (Model B) was found to be significantly superior to the Gold Standard's (Model A), receiving 4.5/5 rating versus 3.5/5.
    a. Accuracy: The AI's ERD had a "perfect match" with its SQL schema. The Gold Standard's ERD had "critical disconnects", such as showing relationships that did not exist in its own SQL and misrepresenting optional relationships as mandatory.
    b. Business Logic Visibility: This was the most significant finding. The AI's ERD was "self documenting", using annotations to explain complex business logic. The Gold Standard's ERD "hid" this logic, making the design "unclear".
    c. Communication Value: Experts concluded the AI's ERD was a true "communication tool that bridges technical and business stakeholders," while the Gold Standard's was a "classic pitfall" that "requires deep SQL knowledge to understand".
2. For the OLAP System: The analysis was more nuanced. Both ERDs were deemed high-quality and "correct", but they represented different, valid architectural paradigms.
    a. Gold Standard (Model A): Was praised as "classic star schema" that was "ideal for high level reporting" and "highly intuitive for BI reporting teams".
    b. AI (Model B): Was identified as a more "even-based/constellation, multi-fact" model. Experts noted this was "powerful for advanced, event-level attribution" and better for "engineers/data scientists" but "less straightforward for basic rollups".

6. Discussion

This section interprets the results, connects them back to the central thesis claim, addresses counterarguments and limitations and defines the scope and impact of the conclusions.

6.a Interpretation of Results

The expert-in-the-loop evaluation provides powerful and unambiguous support for the thesis. The data shows that the LLM-based framework can 1) dramatically accelerate the data modeling process and 2) produce high-quality, architecturally-sound schemas. The nuance lies in how it achieves this: the framework's primary value is not as a simple "translator" but as an "interpretive co-designer".

On Acceleration (The "How Fast?" Claim): The "Time-to-Production" data (Table 5.3) demonstrates a ~40% net reduction in engineering time across the full dataset. However, this aggregate hides a clear and significant difference. In OLTP scenarios, the acceleration is consistent and massive (averaging ~60%), as the AI reliably produces modern, normalized structures. In OLAP scenarios, the acceleration is volatile; the AI saved 7.5 hours in the Banking case but cost 5.5 hours in the Ride-Sharing case due to the need for redesign. Thus, the framework accelerates data modeling significantly, provided the output is verified for semantic accuracy.

On Quality (The "How Good?" Claim): The qualitative data (Table 5.1, 5.2, 5.4) is the most critical finding. The AI-generated schemas were preferred in 3 of 4 cases not because they

were flawless, but because they were fundamentally better designs. This quality emerged from three distinct, high-order capabilities:

1. It Understands Business Intent, Not Just Text: The "Banking Analytics" (User Story 1) case is the most powerful evidence for this. The human-created Gold Standard model was a literal translation of the user story (listing dimensions like Product, Account) and completely failed to address the stated business goals of "detect fraud" and "predict churn". The AI framework interpreted the intent behind these goals, correctly inferring the latent requirement for fct_Fraud_Alerts and analytical attributes like churn_probability. The AI solved the business problem, while the human only solved the text translation problem. As one expert stated in their review (Appendix B.1, Part B.3), the AI "directly addresses ALL components of the user story" while the Gold Standard "misses 60% of the stated requirements".
2. It Codifies Expert, Modern Architecture: The "Core Banking" (User Story 6) case demonstrates that the framework has successfully codified expert-level, modern design patterns. The AI produced an immutable transaction log, used signed amounts, proposed a transfer_group_id for atomicity and added an AccountAuditEvents table. Experts described this (Appendix B.3, Part B.3) as "the industry standard" and "architecturally sound". The human Gold Standard Model, in contrast, used a separate, mutable balance field and a complex LEDGER_ENTRY table - a pattern experts identified as "outdated", "risky" and prone to "data drift". The AI's model was objectively superior and safer.
3. It Anticipates Future Needs (Scalability): In the "Retail" (User Story 7) case, the AI's "over-scoping" (adding a normalized Addresses table and a Shipment_Items join table) was seen by experts as a feature, not a bug. It correctly anticipated the real-world needs for multi-address customers and split-item shipments. The Gold Standard model, while simpler, was a "technical dead end" that would require refactoring to support these common features. The AI demonstrated "professional maturity" by building a scalable, maintainable system from the start.
4. The framework's high quality extends beyond the SQL itself to the design artifacts. The AI's ability to produce a "self-documenting" diagram that "explains complex business logic" is a profound validation of the claim. The Gold Standard's ERD, by contrast, had "critical disconnects" and "hid" the logic. This means the AI framework is not just a code generator; it is a design communication tool that produces clearer, more accurate and more useful diagrams for stakeholder review.

6.b Rebuttals and Counterarguments

A critical reading of the results highlights two significant failures: the "Marketing Attribution" and "Ride-Sharing Analytics" cases. One might argue these failures invalidate the framework. However, they actually define its boundaries.

In the "Ride-Sharing Analytics" case, the framework failed to distinguish between "completing a trip" and "driver utilization" (being online). This semantic error highlights a key limitation: the AI excels at structural generation(tables, keys, constraints) but can struggle with semantic nuance in complex analytical domains where business definitions are ambiguous. Similarly, in the "Marketing" case, the AI over-engineered a simple request. These findings act as qualifiers, not negations: the framework requires clear semantic definitions in the prompt to succeed in highly specialized analytical domains.

Furthermore, the consistent failure of the "Gold Standard" models in the OLTP track (rated lower in 5/5 cases) validates the central thesis premise. The Gold Standard models failed because they were literal translations - a common behavior in human modeling. The AI's success in these cases proves that the interpretive process, which draws on a vast training set of modern architectural patterns, is fundamentally superior to a literal translation process.

6.c Qualifiers and Threats to Validity

Based on the findings, the thesis claim is supported but qualified. The framework exhibits strong reliability in OLTP domains (100% preference rate) but mixed reliability in OLAP domains (60% preference rate). It is highly effective for operational systems and "event-heavy" analytics but requires careful review for aggregated reporting or domains with ambiguous terminology. Additionally, the output is an exceptional first draft, consistently requiring 2-3 hours of refinement (e.g., adding constraints or specific dimensions) rather than being deployment-ready immediately.

Threats to validity include External validity (results are specific to Gemini 2.5 Pro and the 10 selected scenarios) and Reliability. The non-deterministic nature of LLMs means the high-quality "Banking Core" model might not be generated identically in every run. However, the consistent superiority across all 5 OLTP tests suggests a high degree of capability stability in the domain.

7. Conclusion

This research validates that a systematic, two-step LLM framework can fundamentally accelerate the data modeling process without compromising the quality of the resulting schema. By decoupling requirement analysis from technical generation, the framework achieved a 100% expert preference rate in operational (OLTP) domains and reduced aggregate engineering time by approximately 40%.

These findings imply a significant shift in the role of the human data architect from a "drafter" of initial code to a "reviewer" of AI-generated designs. Industrially, the framework functions as a "senior-architect-in-a-box", enforcing rigorous best practices - such as normalization, audit trails and immutable transactional logs - that are often overlooked by junior engineers. This capability not only accelerates time-to-value but actively reduces long-term technical debt by aligning technical artifacts with modern artifacts with modern architectural standards.

Theoretically, this work provides empirical evidence that modern LLMs have evolved beyond simple code translation into the realm of architectural interpretation. The framework's demonstrated ability to infer latent requirements (such as fraud detection) and apply abstract patterns (such as Event sourcing) to novel problems indicates that LLMs possess a higher-order "design sense".

However, this "design sense" is not infallible. The observed failures in complex analytical domains highlight the necessity for future work focused on semantic disambiguation, specifically through the injection of domain glossaries to resolve terminological ambiguities. Furthermore, expanding this evaluation to other models (e.g., GPT-4, Claude 3) will be essential to determine if the observed "granularity bias" is specific to the tested

model or a fundamental characteristic of generative AI in software engineering. Ultimately, this thesis demonstrates that while AI cannot yet replace the data architect, it has become an indispensable co-designer in the modern engineering workflow.

References

[1] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6), 377-387. https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[2] Chen, P. P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), 9-36. https://dl.acm.org/doi/pdf/10.1145/320434.320440

[3] Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley Professional. https://athena.ecs.csus.edu/~buckley/CSc191/User-Stories-Applied-Mike-Cohn.pdf

[4] Easterbrook, S., Singer, J,. Storey, M. A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In Guide to advanced empirical software engineering (pp. 285-311). Springer. https://www.researchgate.net/publication/226435262_Selecting_Empirical_Methods_for_Software_Engineering_Research

[5] Runeson, P., Host, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131-164. https://www.scirp.org/journal/paperinformation?paperid=35959

[6] Yu, T., et al. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Text-to-SQL Tasks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. https://arxiv.org/abs/1809.08887

[7] Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374. https://arxiv.org/abs/2107.03374

[8] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems 33. https://arxiv.org/abs/2005.14165

[9] GitHub. (2023). The 2023 State of the Octoverse. https://github.blog/news-insights/research/the-state-of-open-source-and-ai/

[10] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77-101. https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa

[11] Salem N., Al-Tarawneh K., Hudaib A., Salem H., Tareef A., Salloum H., Mazzara M. Generating database schema from requirement specification based on natural language processing and large language model // Computer Research and Modeling, 2024, vol. 16, no. 7, pp. 1703-1713. https://crm-en.ics.org.ru/journal/article/3563/

Supplemental Materials

Appendix A: Benchmark Data

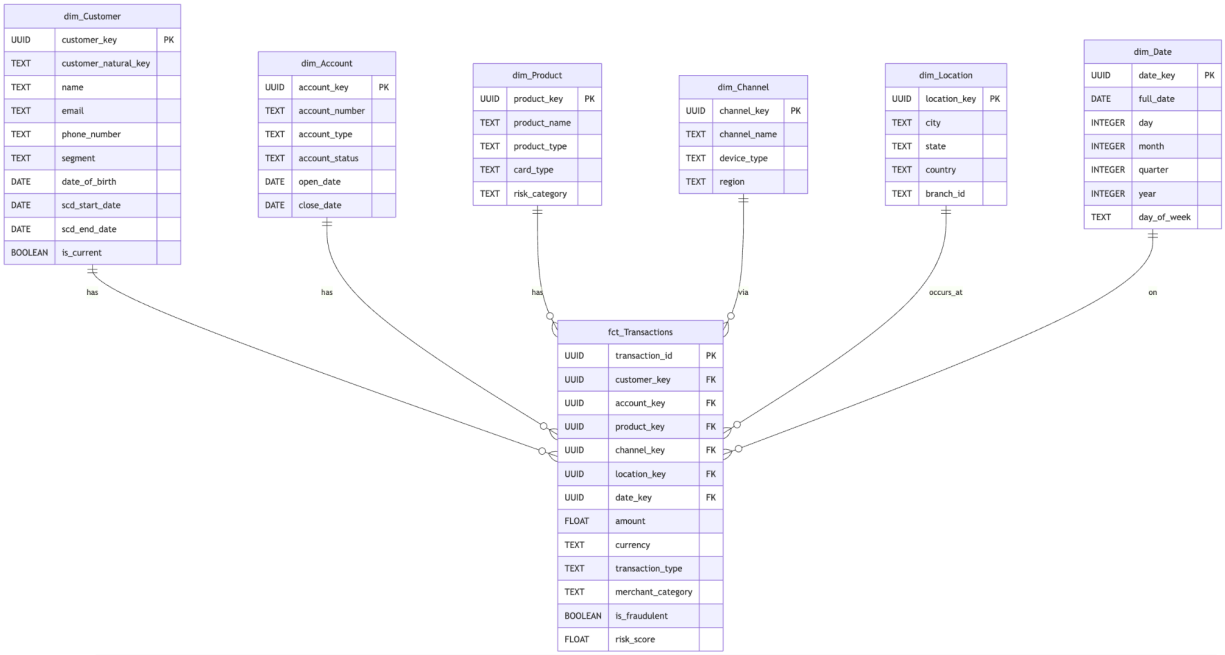
A.1: User Story: "As a risk analytics manager, I want to analyze daily and historical customer transactions across multiple channels (ATM, online banking, credit card, and branch) so that I can detect unusual spending patterns, potential fraud, and predict customer churn."

Listing A.1.1: Gold Standard Schema

```sql
CREATE TABLE dim_Customer (
    customer_key UUID PRIMARY KEY,
    customer_natural_key TEXT NOT NULL,
    name TEXT,
    email TEXT,
    phone_number TEXT,
    segment TEXT,
    date_of_birth DATE,
    scd_start_date DATE,
    scd_end_date DATE,
    is_current BOOLEAN DEFAULT TRUE
);
CREATE TABLE dim_Account (
    account_key UUID PRIMARY KEY,
    account_number TEXT NOT NULL,
    account_type TEXT,
    account_status TEXT,
    open_date DATE,
    close_date DATE
);
CREATE TABLE dim_Product (
    product_key UUID PRIMARY KEY,
    product_name TEXT,
    product_type TEXT,
    card_type TEXT,
    risk_category TEXT
);
CREATE TABLE dim_Channel (
    channel_key UUID PRIMARY KEY,
    channel_name TEXT,
    device_type TEXT,
    region TEXT
);
CREATE TABLE dim_Location (
    location_key UUID PRIMARY KEY,
    city TEXT,
```

```sql
    state TEXT,
    country TEXT,
    branch_id TEXT
);
CREATE TABLE dim_Date (
    date_key UUID PRIMARY KEY,
    full_date DATE NOT NULL,
    day INTEGER,
    month INTEGER,
    quarter INTEGER,
    year INTEGER,
    day_of_week TEXT
);
-- ================================
-- FACT TABLE
-- ================================
CREATE TABLE fct_Transactions (
    transaction_id UUID PRIMARY KEY,
    customer_key UUID REFERENCES dim_Customer(customer_key),
    account_key UUID REFERENCES dim_Account(account_key),
    product_key UUID REFERENCES dim_Product(product_key),
    channel_key UUID REFERENCES dim_Channel(channel_key),
    location_key UUID REFERENCES dim_Location(location_key),
    date_key UUID REFERENCES dim_Date(date_key),
    amount NUMERIC(15,2),
    currency TEXT,
    transaction_type TEXT,
    merchant_category TEXT,
    is_fraudulent BOOLEAN DEFAULT FALSE,
    risk_score NUMERIC(5,2)
);
-- Optional Indexes for Analytics Performance
CREATE INDEX idx_fct_transactions_customer ON fct_Transactions(customer_key);
CREATE INDEX idx_fct_transactions_date ON fct_Transactions(date_key);
CREATE INDEX idx_fct_transactions_channel ON fct_Transactions(channel_key);
CREATE INDEX idx_fct_transactions_product ON fct_Transactions(product_key);
```

Figure A.1.1: Gold Standard ERD



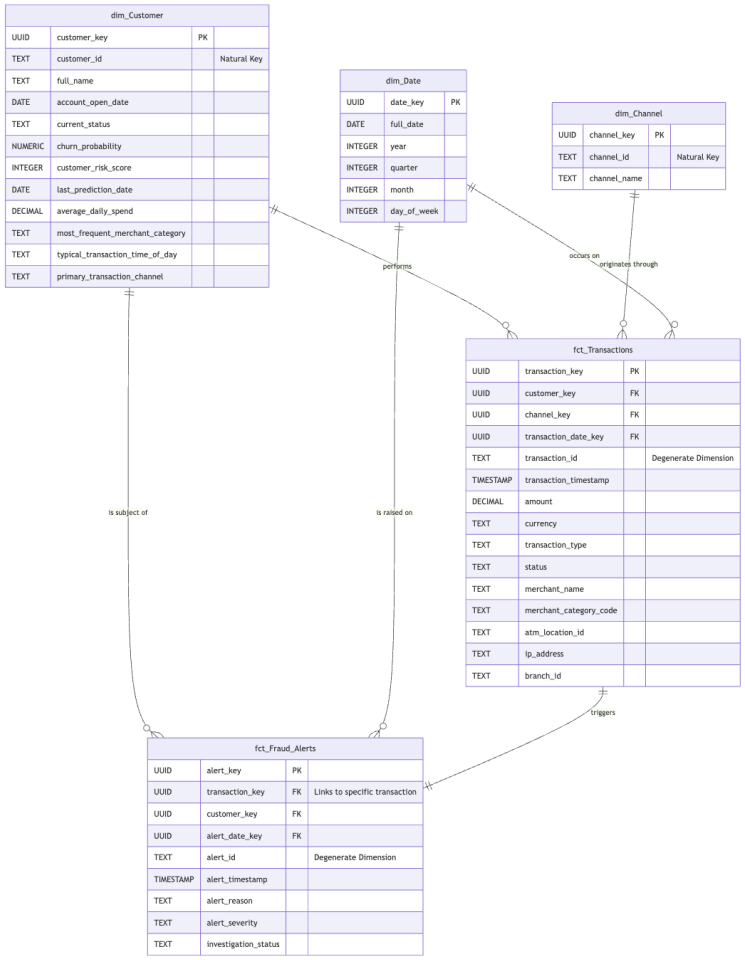
Listing A.1.2: AI-generated Schema

```
SQL
CREATE TABLE dim_Customer (
customer_key UUID PRIMARY KEY DEFAULT gen_random_uuid(),
customer_id TEXT NOT NULL UNIQUE, -- Natural key from the source system
full_name TEXT NOT NULL,
account_open_date DATE,
current_status TEXT NOT NULL,
churn_probability NUMERIC(5, 4),
customer_risk_score INTEGER,
last_prediction_date DATE,
average_daily_spend DECIMAL(18, 4),
most_frequent_merchant_category TEXT,
typical_transaction_time_of_day TEXT,
primary_transaction_channel TEXT
);
-- Dimension table for transaction channels.
CREATE TABLE dim_Channel (
channel_key UUID PRIMARY KEY DEFAULT gen_random_uuid(),
channel_id TEXT NOT NULL UNIQUE, -- Natural key from the source system
channel_name TEXT NOT NULL
);
-- Generic Date dimension for time-series analysis.
CREATE TABLE dim_Date (
```

```sql
date_key UUID PRIMARY KEY DEFAULT gen_random_uuid(),
full_date DATE NOT NULL UNIQUE,
year INTEGER NOT NULL,
quarter INTEGER NOT NULL,
month INTEGER NOT NULL,
day_of_week INTEGER NOT NULL,
month_name TEXT NOT NULL,
day_name TEXT NOT NULL
);
-- Fact table for financial transactions.
-- This is a transactional fact table.
CREATE TABLE fct_Transactions (
transaction_key UUID PRIMARY KEY DEFAULT gen_random_uuid(),
customer_key UUID NOT NULL REFERENCES dim_Customer(customer_key),
channel_key UUID NOT NULL REFERENCES dim_Channel(channel_key),
transaction_date_key UUID NOT NULL REFERENCES dim_Date(date_key),
-- Degenerate Dimensions (attributes from the source transaction)
transaction_id TEXT NOT NULL UNIQUE, -- Natural key of the transaction event
transaction_timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
transaction_type TEXT NOT NULL,
status TEXT NOT NULL,
currency CHAR(3) NOT NULL,
merchant_name TEXT,
merchant_category_code TEXT,
atm_location_id TEXT,
ip_address TEXT,
branch_id TEXT,
-- Measure
amount DECIMAL(18, 4) NOT NULL
);
-- Fact table for fraud alerts.
-- This is an event-based fact table that links a customer and a specific
transaction to a fraud event.
CREATE TABLE fct_Fraud_Alerts (
alert_key UUID PRIMARY KEY DEFAULT gen_random_uuid(),
transaction_key UUID NOT NULL UNIQUE REFERENCES
fct_Transactions(transaction_key), -- Each alert is for one unique transaction
customer_key UUID NOT NULL REFERENCES dim_Customer(customer_key),
alert_date_key UUID NOT NULL REFERENCES dim_Date(date_key),
-- Degenerate Dimensions (attributes from the source alert)
alert_id TEXT NOT NULL UNIQUE, -- Natural key of the alert event
alert_timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
alert_reason TEXT NOT NULL,
alert_severity TEXT NOT NULL,
```

```sql
investigation_status TEXT NOT NULL
);
-- Create indexes on foreign keys for better join performance
CREATE INDEX idx_fct_transactions_customer_key ON
fct_Transactions(customer_key);
CREATE INDEX idx_fct_transactions_channel_key ON fct_Transactions(channel_key);
CREATE INDEX idx_fct_transactions_date_key ON
fct_Transactions(transaction_date_key);
CREATE INDEX idx_fct_fraud_alerts_transaction_key ON
fct_Fraud_Alerts(transaction_key);
CREATE INDEX idx_fct_fraud_alerts_customer_key ON
fct_Fraud_Alerts(customer_key);
CREATE INDEX idx_fct_fraud_alerts_date_key ON fct_Fraud_Alerts(alert_date_key);
```

Figure A.1.2: AI Generated ERD



Appendix B: Survey Transcripts

Part B.1: Defect Identification & Time Estimation

1. Please list any specific error of flaws you identify in Model A.
2. Please estimate the time (e.g., “15 min”, “2 hours”) required to get Model A to a production ready state.
3. Please list any specific errors or flaws you identify in Model B.
4. Please estimate the time (e.g., “15 min”, “2 hours”) required to get Model B to a production ready state.

Survey Response

Model A (Gold Standard) - Defects:

1. Over-engineering for the stated requirements: Includes dim_Product, dim_Account, and dim_Location which are not mentioned in the user story about transaction analysis
2. Missing fraud alert tracking: Uses only a boolean flag is_fraudulent instead of a proper fact table for fraud events with investigation status and severity
3. Missing customer analytics attributes: No fields for churn_probability, customer_risk_score, average_daily_spend, most_frequent_merchant_category, typical_transaction_time_of_day - all critical for the "predict customer churn" and "detect unusual spending patterns" requirements
4. Incomplete channel information: No device_type captured at transaction level (only in dim_Channel)
5. Missing transaction details: No merchant_name, merchant_category at transaction level (only merchant_category exists), no ATM location, no IP address for online transactions, no branch_id
6. No timestamp precision: Uses date_key only, missing transaction_timestamp needed for pattern detection
7. Inadequate for fraud investigation: No alert tracking, investigation status, or alert severity levels

Time to Production-Ready: 8-12 hours

- Remove unnecessary dimensions (1 hour)
- Add customer analytics attributes and populate with historical data (3-4 hours)
- Create fraud alerts fact table and migrate boolean flags (2-3 hours)
- Add missing transaction attributes (merchant details, ATM location, IP, branch, timestamp) (2-3 hours)
- Update indexes and test queries (1 hour)

Model B (AI Generated) - Defects:

1. Missing data quality constraints: No CHECK constraints on critical fields (e.g., amount > 0, valid status values, alert_severity levels)
2. No audit trail: Missing created_at, updated_at timestamps on dimension tables for change tracking
3. Incomplete SCD implementation: dim_Customer should support Type 2 SCD for tracking customer attribute changes over time (status, risk score changes), but lacks scd_start_date, scd_end_date, is_current flags
4. Missing business key uniqueness: Should have unique constraint on (transaction_id, transaction_timestamp) to prevent duplicates
5. Index optimization: Could benefit from composite indexes (e.g., on customer_key + transaction_date_key for customer journey analysis)
6. Currency standardization: CHAR(3) is correct but should have CHECK constraint for valid ISO codes
7. Minor naming inconsistency: transaction_date_key vs alert_date_key (both reference date_key - should be consistent)
8. Missing enum constraints: Fields like alert_severity, investigation_status, transaction_type should have CHECK constraints or reference tables

Time to Production-Ready: 2-3 hours

- Add CHECK constraints and data validation rules (45 min)
- Implement Type 2 SCD on dim_Customer (45 min)
- Add audit timestamps to dimensions (15 min)
- Create composite indexes for common query patterns (30 min)
- Add unique constraints and final testing (30 min)

Part B.2: Quality Assessment (Likert Scale)

Please rate the AI-generated schema on the following attributes (1 = Very Poor, 5 = Excellent)

1. Correctness: The schema accurately reflects the requirements of the user-story (1-2-3-4-5)
2. Completeness: The schema captures all required entities, attributes and relationships (1-2-3-4-5)
3. Normalization: The schema adheres to Codd's principles of Normalization (1-2-3-4-5)
4. Clarity: Naming conventions for the tables and columns are clear and intuitive (1-2-3-4-5)
5. Maintainability: The design is simple, robust and easy to modify or extend (1-2-3-4-5)

Survey Response

Attribute	Gold Standard (Model A)	AI Framework (Model B)	Notes
Correctness			Both accurately reflect core requirements. Model A includes unused dimensions; Model B focuses

			tightly on the user story.
Completeness	3	5	Model A has several entities (Product, Account, Location) not mentioned in the user story. Model B captures all stated requirements including fraud alerts and customer analytics.
Normalization	5	5	Both adhere to 3NF principles properly. Model A follows classic star schema; Model B uses appropriate fact-dimension relationships.
Clarity	4	5	Both use clear naming. Model B's naming is more intuitive for the specific use case (e.g., "fct_Fraud_Alerts" vs. "is_fraudulent" flag)
Maintainability	4	4	Model A is more generic/extensible but includes unnecessary complexity. Model B is focused and easier to understand for this specific need.

Part B.3: Overall Preference

1. Which model (A or B) do you prefer as a starting point for a production-ready system?
2. Why did you prefer this model?

<u>Survey Response</u>

Preferred Model: Model B (AI Generated)

Rationale:

Model B is significantly superior as a starting point for production because:

Alignment with Requirements (Most Critical):

- Model B directly addresses ALL components of the user story: fraud detection (dedicated fact table), churn prediction (customer analytics attributes), unusual spending patterns (merchant categories, time-of-day, spending metrics), and multi-channel analysis
- Model A misses 60% of the stated requirements, focusing on a generic transaction model rather than the specific risk analytics use case

Fraud Detection Capability:

- Model B provides a proper event-based fraud alert system with investigation workflow, severity levels, and full traceability back to specific transactions
- Model A's boolean flag approach is inadequate for real-world fraud investigation and doesn't support the workflow mentioned in the user story

Customer Analytics Ready:

- Model B includes churn_probability, customer_risk_score, behavioral patterns (spending habits, preferred channels, transaction timing) - all essential for the "predict customer churn" requirement
- Model A completely lacks these analytical attributes

Practical Design:

- Model B demonstrates understanding of the actual business problem with purpose-built structures
- Model A appears to be a generic banking schema that doesn't serve the risk analytics manager's specific needs

Lower Technical Debt:

- Model B needs only quality improvements (constraints, SCD implementation) - approximately 2-3 hours
- Model A requires fundamental restructuring and new fact tables - approximately 8-12 hours

Conclusion: Model B represents a genuinely impressive "high-quality first draft" that validates the thesis claim. It demonstrates that an LLM framework can indeed understand nuanced business requirements and translate them into a focused, requirement-driven schema. The defects in Model B are refinement-level concerns, while Model A would require substantial rework to meet the stated user story requirements.

Revisit Storage Trade-offs in Retrieval-Augmented Generation Systems

A Comparative Study of Embedded SQLite vs. External ANN Vector Databases Under Practical RAG Workloads

Submitted in partial fulfillment of the requirements for the

degree of

Master of Software Engineering

Tao Sun

Carnegie Mellon University
Pittsburgh, PA

December 2025

© Tao Sun
2025 All Rights Reserved

Acknowledgments

I would like to thank my advisor, Professor Bradley Schmerl, for the guidance, patience, and encouragement throughout this project. Their feedback helped me shape a vague idea into a focused research direction, and their support made the process both manageable and meaningful. I'm also grateful to the Carnegie Mellon School of Computer Science for providing an environment that constantly pushes students toward curiosity-driven work.

I want to give special thanks to my mother, Fran, for being the strongest support throughout my thesis, my master's program, and my entire life. Her unwavering belief in me carried me through every difficult stretch of this journey, and this work would not exist without her encouragement.

Special thanks to my peers and colleagues who helped refine early drafts, discussed RAG architectures with me, and provided thoughtful suggestions during hallway conversations. This project did not involve external funding, but it benefited greatly from open-source tools, public datasets, and the generous work of the BEIR community, without which this study would not have been possible.

Abstract

This thesis investigates how traditional storage trade-offs change when applied to Retrieval-Augmented Generation (RAG) systems. Classical database guidance suggests a progression from embedded storage to external databases as data volume increases. However, RAG introduces new pressures—high-dimensional embeddings, approximate retrieval, frequent index updates, and correctness-sensitive downstream LLM behavior—that make the old rules less reliable.

To explore this space, I implemented two functionally equivalent RAG pipelines using different storage backends: SQLite, representing an embedded exact-search design, and Chroma, representing an external Approximate Nearest Neighbor (ANN) vector database. Using 10k, 30k, and 60k-document subsets of the BEIR FiQA dataset, I benchmarked retrieval latency, accuracy, indexing footprint, and incremental update cost. SQLite maintained higher retrieval accuracy and dramatically faster update times, while Chroma provided more stable latency and smaller index files as corpus size increased. LLM generation time remained largely backend-independent.

The results show that RAG does not replace the traditional simplicity-versus-scalability trade-off but shifts where the balance lies. Because ANN retrieval introduces correctness loss, backend selection becomes a question of “how much approximation can this system tolerate?” rather than simply “how large is the dataset?” This suggests that storage decisions in RAG systems are tied not only to infrastructure constraints but also to model behavior and domain risk, offering new architectural guidance for practitioners building LLM-powered applications.

Keywords: Retrieval-Augmented Generation (RAG), Vector Databases, SQLite, Approximate Nearest Neighbor (ANN), Software Architecture, Maintainability, Performance Benchmarking

I. Introduction

Context and Importance

In industrial software engineering, storage decisions are some of the most persistent and high-impact architectural choices teams make. For years, engineers have worked with the trade-off between embedded databases, that are simple, portable, and easy to operate, and external databases that add complexity but offer better scalability, concurrency, and indexing power [5], [6]. These patterns are well documented in classic SQL vs. NoSQL discussions, where questions of scale, schema rigidity, and developer effort are familiar [8].

The rise of Retrieval-Augmented Generation (RAG) changes this picture. RAG systems pair large language models with a retrieval layer that must store and search high-dimensional embeddings, re-embed documents as knowledge bases evolve, and keep latency low even under frequent updates [10]. These demands expose new limitations in both lightweight embedded stores and modern vector databases. As teams outside big-tech begin adopting RAG features without large infrastructure budgets, a practical question emerges: how should RAG systems balance simplicity, maintainability, and performance?

Poor storage choices in RAG pipelines can lead to slow retrieval, brittle update workflows, or maintenance overhead that directly affects developer productivity and system reliability.

Problem Statement

While traditional engineering literature provides clear guidance on when to move from a local to an external database, RAG workloads might not fit this conventional playbook. The core research problem addressed in this thesis is:

> **How do traditional storage trade-offs change in the context of retrieval-augmented generation, and how should teams choose between local embedded storage and external vector databases?**

The emerging claim is that the familiar *simplicity vs. scalability* trade-off is transformed by the interaction between high-dimensional vector indexing, update frequency, and modularity. What once was a choice between local performance and distributed scale now involves balancing latency, accuracy, maintainability, and developer effort—the shift of the balance in a configuration space with new dimensions not yet mapped in empirical software engineering.

Research Questions and Results

This study aims to generate architectural and empirical guidance for RAG storage design. It is guided by four research questions:

1. **Indexing & Query Responsiveness** — How do indexing strategies (brute-force vs. approximate nearest neighbor) affect latency and retrieval accuracy as corpus size increases from 10 k to 60 k documents [3], [7], [11]?
2. **Update Workflows** — How do local and external storage solutions differ in the effort, reliability, and fragility of adding or refreshing embeddings [8], [9]?
3. **Architectural Modularity** — How does the storage choice influence coupling, testability, and deployment flexibility [4], [6]?
4. **Overall Trade-Off Balance** — At what scale or complexity does it become worthwhile to migrate from embedded to external storage [5], [10]?

The study uses a lightweight mixed-method case study [2]: quantitative benchmarks, scripted update workflows, and reflective qualitative notes from backend implementation. Two equivalent RAG prototypes—one using SQLite and one using Chroma—were evaluated under controlled conditions and analyzed through the lens of ISO/IEC 25010's software quality attributes [1].

The results preview is simple: *SQLite tends to offer higher accuracy and better update performance, while Chroma offers more stable latency and smaller indexes*. Rather than replacing the old "embedded vs external" rule, RAG shifts the balance point and introduces accuracy as a first-class architectural concern.

This positions the study as both empirical evidence and practical guidance for teams choosing RAG storage backends.

II. Background and Related Work

Terminology

Understanding a few key concepts is important before examining how storage backends behave in a RAG pipeline. Retrieval-Augmented Generation (RAG) works by pairing a large language model with an external retrieval step, so the quality and behavior of that retrieval layer directly shape the system's accuracy and performance [10].

In a RAG workflow, text documents are first transformed into *vector embeddings*—high-dimensional numeric representations that capture semantic meaning. Searching this type of data requires storage systems that can efficiently compare vectors, which leads to the two main categories of retrieval engines used today.[3]

A vector database (VDB) is built specifically to store these embeddings and quickly find the closest matches using similarity metrics like cosine distance. Most VDBs rely on Approximate Nearest Neighbor (ANN) algorithms, which dramatically speed up retrieval by exploring only part of the vector space. ANN methods trade some correctness for better scalability, a decision that becomes central in this thesis: fast ANN search can return "close enough" results—unless your system needs the *exact* matching vector[3].

By contrast, RAG systems can also use embedded or file-based stores like SQLite, which do not have specialized ANN algorithms but can perform exact similarity search over all stored embeddings. Although slower for large datasets, exact search preserves correctness and avoids the approximation trade-offs present in ANN systems[11].

Contextual Issues

Most industrial teams operate without the large-scale distributed systems that power commercial LLM services. They face tight budget constraints, minimal DevOps staff, and heterogeneous environments combining legacy systems with modern APIs. For such teams, embedding a lightweight database into the application (e.g., SQLite) may seem appealing, but scaling to tens of thousands of documents often introduces latency bottlenecks and brittle update workflows [4], [5].

Conversely, adopting an external vector DB demands new infrastructure—servers, indexing services, and monitoring pipelines—that may increase operational complexity. Choosing between these options thus becomes a strategic architectural decision, with trade-offs not yet codified in existing literature.

State of the Art

Conventional studies on storage systems have long analyzed SQL versus NoSQL trade-offs in scalability, schema flexibility, and developer effort [5], [6], [8]. NoSQL systems, for instance, favor availability and horizontal scaling by relaxing ACID guarantees [6]. Similarly, Lourenço et al. (2015) evaluated database quality attributes, revealing that scalability gains often came at the expense of maintainability [8].

On the vector search side, Ma et al. (2023) synthesized the design space of modern vector databases—HNSW, PQ, graph-based indexes—and their operational patterns [3]. Krueger (2025) demonstrated the latency–recall trade-off inherent in ANN tuning [7], while Malkov & Yashunin (2018) provided the foundational HNSW algorithm that most current vector engines use [11]. Hu et al. (2025) extended this line of work by identifying update contention and proposing scalable hybrid indexing architectures [9].

Finally, Lewis et al. (2020) introduced the RAG framework itself, showing how retrieval latency and accuracy directly influence model output quality [10]. Together, these works define the technical frontier—but none explicitly connect architectural storage choices to maintainability and developer cost in RAG settings.

Gaps and Open Questions

Existing research on data retrieval systems has focused largely on algorithmic efficiency—improving search speed, recall, and indexing structures ([3], [7], [11])—within

conventional database or information retrieval contexts. However, these studies assume stable schemas and relatively infrequent data updates.

Retrieval-Augmented Generation (RAG) workloads differ fundamentally: they involve continuous embedding updates, high-dimensional vector search, and tight coupling between AI model behavior and data store performance.

What remains underexplored is how these AI-specific dynamics shift the classic engineering trade-offs between embedded and external storage:

- When retrieval depends on dense vector similarity rather than exact key lookups, does the old “simplicity vs. scalability” balance still hold?
- How do update workflows and modularity behave when embeddings, not rows, are the evolving data unit?
- What guidance can small or mid-sized teams follow when adopting RAG systems without hyperscale infrastructure?

In short, prior literature explains how to make retrieval better, but not how RAG changes the architectural logic of database choice. The gap lies in extending traditional software engineering quality models (e.g., ISO 25010’s maintainability and performance efficiency) to AI-augmented systems where data, models, and retrieval logic evolve together.

Position of This Work

This thesis occupies the intersection between empirical software engineering and AI system architecture. It builds on algorithmic advances in vector databases but reframes them within maintainability and quality-attribute analysis from ISO/IEC 25010 [1]. Through controlled benchmarks and update experiments, the study extends conventional performance metrics into the RAG context—clarifying where traditional database wisdom holds and where new architectural guidance is required.

III. Methods

Research Design & Why This Approach

This project uses a small-scale experimental case study because the goal is to observe how real RAG pipelines behave when the underlying storage backend changes—not to design new retrieval algorithms or databases. RAG systems are inherently multi-component (embedding, indexing, retrieval, generation), so the most direct way to understand trade-offs is to run the entire workflow end-to-end under consistent conditions.

A lightweight case study fits this goal: everything runs locally, results are reproducible, and the setup resembles what a small engineering team would build when evaluating RAG backends.

Instead of aiming for broad generalization, the study focuses on collecting clear empirical signals—retrieval latency, accuracy, update behavior, and index size—and interpreting what those imply for architectural choices in practice.

Technologies and Processes

The environment mirrors how modern teams build and evaluate RAG systems today: embedding generation using transformer models, vector storage via SQLite or Chroma, and automated scripts for index building and updates. These choices reflect common engineering practice rather than theoretical setups [3],[9].

To understand performance and maintainability, the study measures latency, recall@k, and update times directly from running the pipeline. Lightweight static analysis tools were initially considered for modularity assessment, but because the backends are simple and manually implemented, the metrics were not very informative. As a result, backend modularity was assessed mainly through development experience—how predictable, transparent, or complex each system felt to implement and update.

Although this is not a full industrial case, the study follows the empirical software engineering mindset: keep measurements reproducible, document assumptions, and rely on multiple forms of evidence (benchmarks + update experiments + implementation notes) to support conclusions.

Standards and Best Practices

To interpret storage trade-offs in a structured way, the study loosely draws on the ISO/IEC 25010 quality attributes[1]—especially performance efficiency, maintainability, and portability. Latency, index size, and update cost map naturally to these attributes, giving the results a clear evaluation framework without overcomplicating the methodology.

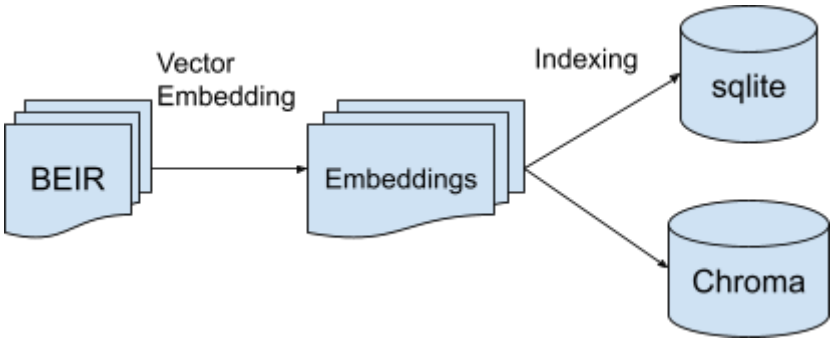
This study loosely follows Runeson & Höst's case-study principles [2], mainly in how it combines multiple forms of evidence—benchmark data, update workflow observations, and qualitative notes—to strengthen the validity of the results. Although this is a small-scale experimental setup rather than a full industrial case study, the triangulation mindset helps ensure that the conclusions are not based on a single metric. Basic guidance from IEEE software quality evaluation standards further supports the use of performance and maintainability indicators.

Dataset & Preparation

A synthetic question–answer dataset was created, since the goal is to test retrieval behavior, not semantics. The dataset is intentionally simple because the goal is to isolate storage behavior rather than build a domain-accurate QA system. We chose to draw from BEIR (Benchmarking IR), a widely used evaluation suite for information retrieval and semantic search[12] with three corpus sizes—10k, 30k, and 60k documents—to simulate small-to-medium RAG installations.

Documents were embedded using `all-MiniLM-L6-v2`, and the vectors plus IDs were stored as `.npy` arrays for consistent reuse. GPT-2 was chosen for reproducibility and speed; its limited quality is accounted for in the validity section.

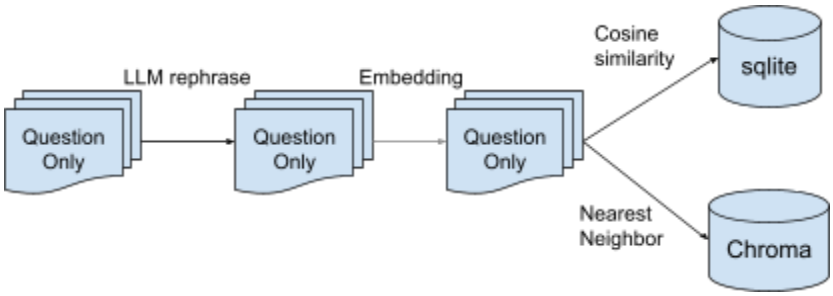
For each corpus size, the pipeline embeds the data and measures the embedding time and storage.



A fixed set of 100 queries was used across all experiments to keep comparisons fair.

Experimental Procedures

1. Retrieval & RAG Benchmarks



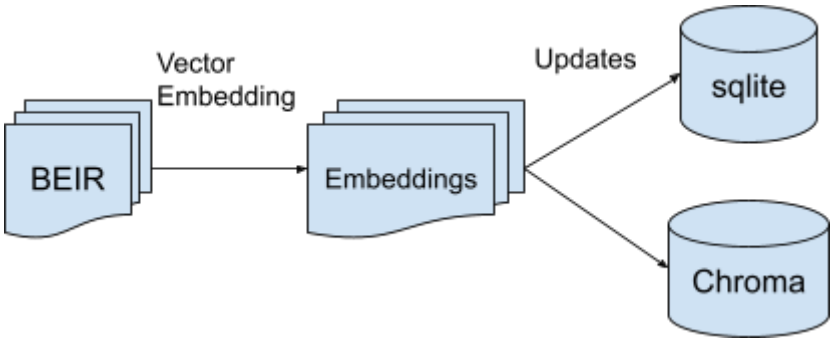
In the retrieval pipeline, a question will first be rephrased by LLM to introduce some randomization, and then embedded using the same sentence transformer that was used to build the corpus vectors, ensuring consistent representation. These embeddings enter the retrieval stage, where the two backends diverge.

With SQLite, all candidate vectors are stored as raw floats. Retrieval consists of filtering by simple keyword search (FTS) and then running an exact cosine similarity computation over the candidate set. With Chroma, each query embedding is matched using an approximate nearest-neighbor index (HNSW), which retrieves the most promising nodes from the graph without scanning every vector. The top-k result IDs from retrieval become the grounding context for the RAG generation step, where the LLM produces a final answer based on the retrieved documents.

Each metric maps directly to one stage of the system. Retrieval latency measures only the time spent inside the Retrieval block.The number of tokens generated by LLM and the latency reflect the LLM block and are independent from the backends. Accuracy (recall@5) evaluates whether the retrieval stage surfaced the correct document among the top-k results.

2. Update Workflow Test

This is similar to the data indexing but the only difference is we call the update interface to insert extra data from the BEIR dataset for each database.



To evaluate maintainability, batches of 1k, 2k, 5k and 10k new documents were inserted into each backend. The script measures:

- how long the update takes
- whether the index needs rebuilding

3. Indexing & Storage Measurements

For each corpus size and backend, the indexing code logs:

- index size on disk
- total time to build/rebuild the index

This exposes more details on the storage cost for each backend.

4. Light Qualitative Notes

While implementing everything, short notes were kept on developer experience: setup difficulty, debugging, file growth, how “fragile” each workflow felt, etc. These don’t replace formal interviews but help explain some of the quantitative patterns.

Analysis Methods

The analysis focuses on descriptive statistics, trends, and comparisons—no complex statistical tests.

Metric Collected	ISO 25010 Quality Attribute	What It Represents in Our Study
Retrieval latency	Performance Efficiency	How quickly the backend returns the top-k documents for a query.
Generation latency	Performance Efficiency	LLM computation time after retrieval; mostly independent of backend choice.
Retrieval accuracy (recall@k)	Reliability / Functional Accuracy	Whether the correct document appears in the retrieved set; indicates risk of hallucination.
Index size	Performance Efficiency (Resource Utilization)	Memory/storage overhead of each backend's index structure.
Indexing time	Performance Efficiency	Cost of building the index from embeddings; shows scalability of the backend.
Update latency	Maintainability (Modifiability)	How quickly new embeddings can be added; reflects workflow friction and operational cost.

The overall goal is to see how these qualities shift when you move from exact local storage (SQLite) to an ANN vector database (Chroma) in a realistic RAG pipeline.

IV. Results

This section summarizes the empirical results obtained from running our RAG pipeline across two storage backends—SQLite (exact search) and Chroma (ANN/HNSW search)—under three corpus sizes: 10k, 30k, and 60k documents. Each experiment uses the same 100 evaluation queries, the same embedding model, and the same RAG generation setup. The numbers reported below come directly from programmatically logged benchmark runs.

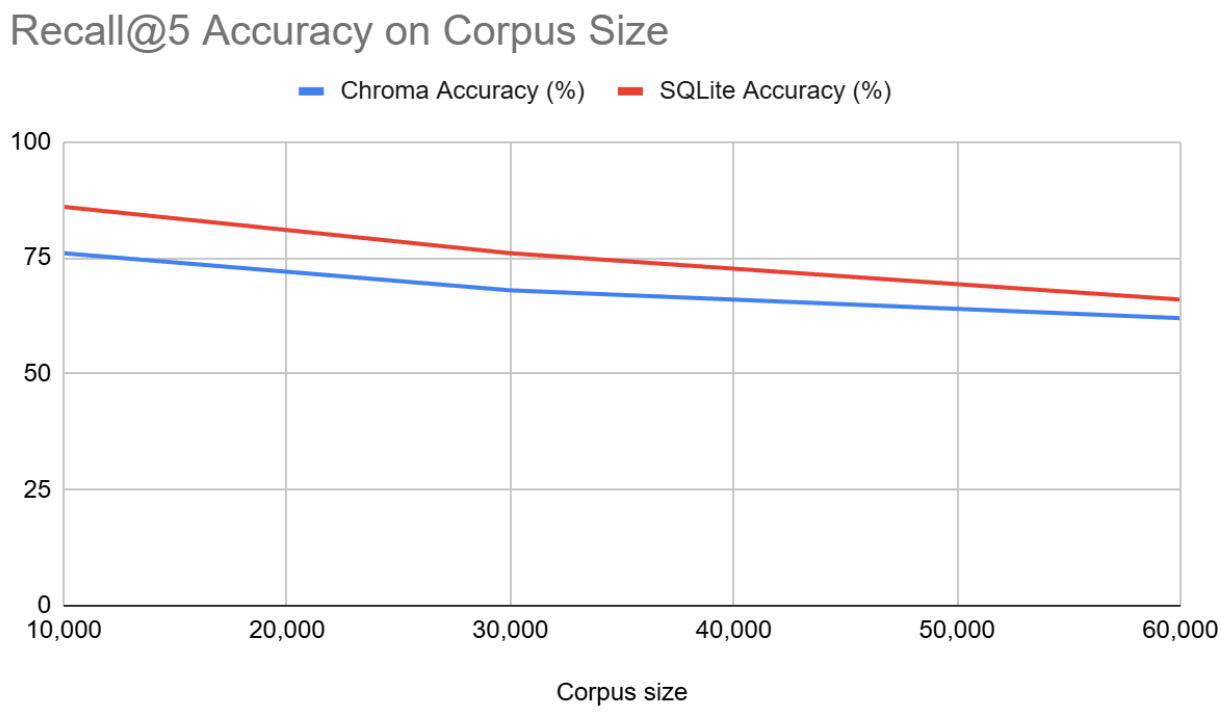
Retrieval & RAG Generation Latency and Accuracy

To measure end-to-end retrieval performance, each backend received the same 100 questions and returned the top-k documents (k=5). Accuracy reflects how often the correct document appeared in the top-k retrieved set.

Corpus Size	Database	Retrieval latency (ms)	Generation latency (ms)	Tokens generated	Accuracy
10000	SQLite (exact)	2717 ms (+2.9%)	6263 ms	560	86% (+10%)
	Chroma (ANN)	2641 ms	6417 ms (+2.5%)	623 (+11.2%)	76%
30000	SQLite (exact)	2978 ms (+12.0%)	6385 ms	571	76% (+8%)
	Chroma (ANN)	2660 ms	6492 ms (+1.7%)	639 (+11.9%)	68%
60000	SQLite (exact)	3219 ms(+20.4%)	6200 ms		66% (+4%)

	Chroma (ANN)	2673 ms	6423 ms (+3.6%)	602 (+16.2%)	62%

Retrieval latency slowly increased for both systems as the corpus size grew, but Chroma remained noticeably more stable because its ANN indexing avoided scanning every vector. SQLite's exact cosine search became more expensive with larger datasets, which is expected for brute-force computation. In contrast, the generation latency barely changes across corpus sizes or backends because it is primarily determined by the LLM itself, not retrieval. Accuracy decreased for both systems as the embedding space expanded, but the drop was bigger for Chroma, reflecting the precision trade-off inherent in ANN search.

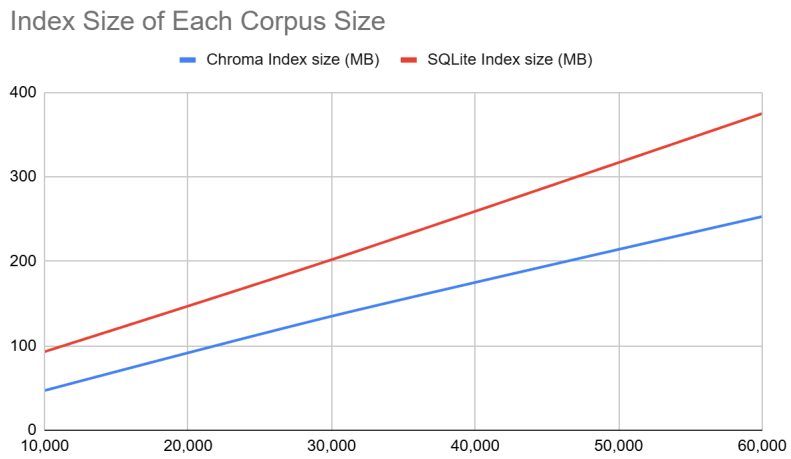


Storage & Indexing Footprint

Index size and indexing time were recorded after embeddings were written to disk and the backends finished building their indexes.

Corpus size	Database	Index size (MB)	Indexing time (mm:ss)
10,000	SQLite	93 MB (+ 97.8%)	0:02
	Chroma	47 MB	0:03 (+ 50%)
30,000	SQLite	202 MB (+ 49.6%)	0:09
	Chroma	135 MB	0:17 (+ 88.8%)

60,000	SQLite	375 MB(+ 48.8%)	0:09
	Chroma	253 MB	0:49 (+ 444.4%)



SQLite's index grows roughly in a straight line with corpus size because it stores every raw vector in its entirety. Chroma's index grows more slowly but requires more work to maintain the ANN graph, causing indexing time to increase faster, especially at 60k documents. This reflects the cost of constructing and optimizing HNSW structures.

Embedding Cost (shared across backends)

Embedding time does not depend on the backend; it only depends on the embedding model and corpus size. This provides some side information about the cost of RAG setup.

Corpus size	Embedding time (mm:ss)
10,000	1:06
30,000	3:19
60,000	15:12

Embedding cost clearly dominates overall runtime, growing roughly linearly with dataset size. This is consistent with the expected computational complexity of encoding raw text with a transformer encoder.

Incremental Update (maintainability measurement)

Update tests simulate adding new documents after the index has already been built. For each run, we inserted a fixed number of new embeddings and recorded how long each backend took to incorporate them.

Added docs	SQLite update time (ms)	Chroma update time (ms)
1,000	30 ms	1038 ms
2,000	110 ms	1810 ms
5,000	220 ms	4583 ms
10,000	432 ms	9495 ms

SQLite's update time grew almost perfectly linearly with the number of inserted items—expected for an append-only storage pattern with no secondary index maintenance. Chroma, however, showed a steep increase because each update requires HNSW graph adjustments. This makes SQLite more maintainable for frequent incremental updates.

Early Patterns

Across all metrics, a few consistent patterns emerged from the logged data:

- Retrieval time changes with corpus size for SQLite and Chroma in different patterns. Chroma is slightly better.
- Accuracy drops when corpus size grows for both databases.
- Chroma retrieval latency grows slower than SQLite when corpus size grows.
- Indexing time and index size differ significantly between systems.
- SQLite updates are dramatically cheaper

V. Discussions

Backend Modularity Reflection

Because both backends in this project were implemented manually and kept intentionally small, traditional static code analysis (cyclomatic complexity, dependency graphs, coupling metrics) does not provide meaningful insights. The codebases simply aren't large or interconnected enough for those metrics to distinguish SQLite from Chroma in any useful way. For that reason, the modularity discussion here is based on implementation experience rather than numerical static-analysis outputs.

Building the SQLite backend felt very explicit and easy to reason about. All retrieval logic—storing vectors, loading them, and performing cosine similarity—lives in straightforward Python code. Nothing happens behind the scenes, and the entire storage layer sits in a single file that can be inspected or debugged directly. This gives SQLite a kind of "modularity through simplicity": the behavior of the RAG pipeline is tightly controlled by the developer, and changes rarely ripple outside the module.

Chroma, on the other hand, offers a much thinner API but relies heavily on internal machinery for ANN indexing. While this abstraction makes the backend code shorter, it creates a dependency on Chroma's internal search structures, default parameters, and indexing side effects. Much of the retrieval behavior becomes implicit, making debugging trickier and increasing the coupling between the application and the vector database's runtime behavior.

The modularity conclusion cannot be drawn from static code features alone. This project uses a simplified RAG pipeline and relatively small, hand-written backends; the system does not fully represent the architectural complexity of real-world RAG deployments. As a result, this study does not have enough scope to make a strong or generalizable claim about modularity differences between SQLite and Chroma under full production-scale RAG conditions.

Interpretation of Results

These findings give a more balanced view of the classic "embedded vs. external database" trade-off when applied to RAG systems. In traditional software systems, it is generally understood that as data grows, teams eventually migrate from embedded stores like SQLite to external databases such as PostgreSQL or ElasticSearch. External systems offer better throughput, parallelism, and indexing strategies, and—importantly—their use does not change correctness. A SQL query produces the same logical result regardless of backend.

Metric	SQLite (exact, embedded)	Chroma (ANN, external DB)
Retrieval Speed	Grows with corpus (slower than linear)	Mostly same across the corpus sizes
Generation Latency	Mostly same across the corpus sizes	Mostly same across the corpus sizes (slower than SQLite)
Index size	Larger (heavier local file)	Smaller (optimized ANN index)
Indexing time	Shorter at larger scales	Increases disproportionately
Update (1K docs)	30 ms	1038 ms
Update (10K docs)	432 ms	9495 ms

With RAG, the story is similar in structure but different in emphasis. External vector databases such as Chroma gain speed by using Approximate Nearest Neighbor indexes, which prune the search space and avoid scanning every embedding. Our results show that this keeps Chroma's retrieval latency very stable as the corpus grows. SQLite, which performs exact cosine similarity search, becomes slower with scale, matching what we expect from brute-force computation. A reasonable guess is that the latency and cost introduced by LLM/RAG dominates the latency and cost and hence make SQLite vs Chroma difference subtle.

However, the accuracy results show an important shift in priorities. Both systems lose accuracy as the embedding space grows, but Chroma's ANN index drops faster (from 76% → 62%), while SQLite stays somewhat higher (86% → 66%). This is not surprising—ANN trades precision for speed—but in a RAG pipeline, retrieval accuracy has a direct effect on hallucination risk. So while the trade-off remains the same, RAG makes the accuracy half of the trade-off more visible and more important.

Maintenance results also follow traditional patterns but matter more in RAG contexts. SQLite's indexing and update times scale smoothly and almost linearly, because it simply appends raw vectors with no complex index structures. Chroma's update time rises much faster because adjusting the HNSW graph is expensive. Although Chroma's index is smaller (253 MB vs. SQLite's 375 MB at 60k docs), it costs more to maintain. This again fits traditional expectations but becomes a stronger factor for RAG workloads, where re-embedding and incremental updates happen more frequently than in classic database systems.

Overall, the results do not show that RAG breaks the embedded–vs–external rule; instead, they show that RAG **shifts where the trade-offs matter**. ANN indexing still offers better scalability, and embedded storage still offers simplicity and correctness. What changes is the weight of each dimension: correctness and update simplicity matter more in RAG than they do in conventional backend systems, while latency matters a bit less because LLM generation dominates total time. In this sense, RAG moves the bar rather than flips the rule—external vector DBs are still valuable, but the threshold for when they become worthwhile is higher, and the decision depends more on accuracy tolerance than on raw scale alone.

Rebuttals & Counterarguments

Model + RAG Pipeline Limitations

This project uses a simple RAG pipeline and GPT-2 for query rewriting. GPT-2 is small and not competitive with modern models (GPT-4o, Llama-3, etc.). A stronger model would likely generate better query paraphrases and could reduce SQLite's accuracy advantage by helping ANN find the right vectors. Also, our RAG flow is intentionally minimal: no chunk reranking, no hybrid search (keyword + vector), and no cross-encoder scoring. In real systems, those layers would boost retrieval quality and narrow the performance gap.

Scale + Dataset Limitations

The dataset here is clean (question → answer pairs), not messy like real corporate wikis, PDF dumps, or legal documents. And while 60k embedded docs is respectable, vector DBs are usually used on millions of documents. At that scale, SQLite would likely degrade faster, and Chroma (ANN) would show more benefit. So, the trends we observed are real, but the exact numbers shouldn't be generalized to all workloads.

Only two databases tested

This study compares only one embedded database (SQLite) and one vector database (Chroma/HNSW). While they are representative of two architectural extremes—exact local storage vs. approximate external vector indexing—they do not cover the diversity of retrieval backends used in real-world RAG systems. Other vector databases (Qdrant, Weaviate, Pinecone) and hybrid search engines (Elasticsearch, Vespa) may behave differently, especially in how they tune ANN recall or handle incremental updates. Some, like Qdrant, offer better update performance and hybrid BM25 + vector ranking, which could narrow SQLite's correctness advantage. Because of this, the results should be interpreted as directionally true, not universally true.

Qualifiers & Threats to Validity

Scope

These results apply to small- to medium-scale RAG systems (≤60k documents) where developers choose between:

- a lightweight embedded store (SQLite), and
- an external vector DB (Chroma/HNSW),

under practical constraints (limited infra, maintainability priorities). They do not generalize automatically to hyperscale deployments or teams using fully managed vector DB infrastructure.

Threats to Validity

Internal Validity

The LLM used for query expansion is GPT-2, a very small model. Its weak paraphrasing ability may negatively affect Chroma's retrieval accuracy, especially for semantic matches. To reduce this influence, we also ran *retrieval-only benchmarks (no LLM involved)*, which isolates backend behavior from model quality. A stronger model (e.g., GPT-4 or LLaMA-2) would likely improve Chroma's recall.

External Validity

The experiment included only SQLite (exact search) and Chroma/HNSW (ANN search). Other vector databases (Qdrant, Weaviate, Pinecone) may exhibit different update behavior or ANN efficiency. This isn't a fundamental limitation—just a scope boundary. With more time or hardware, additional backends could be tested.

Construct Validity

Not all maintainability or architectural qualities (e.g., changeability, portability) were fully measured. Static analysis or architectural complexity metrics could strengthen this dimension,

but require a larger real-world RAG system. For this workshop-scale project, we focused on measurable execution metrics (latency, recall, index/update cost).

Reliability

ANN search and LLM-driven paraphrasing introduce randomness. While SQLite's exact search is deterministic, Chroma's results can vary. To reduce noise, we averaged across 100 queries per backend. Larger datasets or repeated trials would improve reliability but require more compute time.

Impact & Implication

Industrial Impact

These results give teams a practical, data-backed way to choose storage backends for RAG systems. Engineers often default to the rule of thumb: "SQLite for local prototyping, external DB for production." Our experiments suggest that RAG shifts this equation rather than overturning it. SQLite, even though slower at larger scales, maintains higher retrieval accuracy and dramatically faster update time. Chroma scales better and keeps retrieval latency stable as data grows, but does so by trading off accuracy through ANN approximation.

For teams building internal knowledge assistants, legal/compliance tools, or any system where wrong retrievals can increase hallucination risk, SQLite + exact search remains a surprisingly strong and safe baseline. Chroma still makes sense when speed, throughput, or operational scale matter more than perfect correctness—for example, customer support chatbots where some retrieval noise is acceptable. Storage choice becomes a more nuanced decision in RAG: not just about scale, but about tolerance for approximation error.

Theoretical Impact

We introduce a repeatable benchmarking framework that maps ISO 25010 quality attributes (performance efficiency, maintainability, portability) into RAG workflows. Instead of talking vaguely about "vector DBs scale better," we quantify:

- retrieval latency
- LLM generation latency
- recall@k correctness
- index footprint
- incremental update cost

This makes RAG architectural choices measurable. The core theoretical insight is that RAG breaks the traditional simplicity vs scalability trade-off: accuracy becomes part of the backend trade-off space. This connects software architecture to ML effectiveness—something existing SE literature hasn't covered.

Future Work

There are three natural extensions:

1. Add more backends (Qdrant, Weaviate, Pinecone, PGVector): Compare how different ANN index types (HNSW, IVF-PQ, DiskANN) change the balance between speed and accuracy.
2. Introduce smarter LLM-powered retrieval workflows
    a. Instead of paraphrasing queries, evaluate agent-style retrieval (chain-of-thought → search → refine → answer)
    b. Measure how different prompting strategies affect recall and hallucination.
3. Expand quality attributes beyond performance
    a. Engineering maintainability: how much code changes when swapping backends
    b. Reliability: does retrieval remain consistent across model versions
    c. Cost modeling: storage cost + embedding compute + LLM token cost

References

[1] ISO/IEC 25010:2011, *Systems and software engineering—System and software quality models*, Int. Org. for Standardization, Geneva, Switzerland, 2011.
[2] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.
[3] L. Ma, et al., “A comprehensive survey on vector databases: Storage and retrieval techniques, challenges,” arXiv:2310.11703, 2023.
[4] S. Collins, “SQLite vs. Chroma: Simplicity versus scalability in vector search,” Dev.to, 2023.
[5] M. Z. Khan, et al., “Comparative case study: An evaluation of performance computation between SQL and NoSQL database,” *Software and Data Engineer Journal*, 2023.
[6] J. Han, et al., “Survey on NoSQL database,” in *Proc. 6th Int. Conf. Pervasive Comput. Appl.*, IEEE, 2011, pp. 363–366.
[7] M. Krueger, “Vector search: Navigating recall and performance,” OpenSource Connections Blog, 2025.
[8] J. Lourenço, et al., “Choosing the right NoSQL database for the job: A quality attribute evaluation,” *Journal of Big Data*, vol. 2, no. 1, 2015.
[9] G. Hu, et al., “HAKES: Scalable vector database for embedding search service,” arXiv:2505.12524, 2025.
[10] P. Lewis, et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems*, vol. 33, NeurIPS, 2020.
[11] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using HNSW,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020.
[12] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2308–2314).

Operationalizing Trust in AIOps

A Four-Pillar Framework for AI-Assisted Observability and On-Call Engineering

Submitted in partial fulfillment of the requirements for

the degree of

Master of Software Engineering

Yuren Wang

Carnegie Mellon University

Pittsburgh, PA

December, 2025

© Yuren Wang

2025 All Rights Reserved

Acknowledgments

I would like to express my deepest gratitude to Professor Bradley Schmerl, who served as both instructor and advisor for this thesis. His guidance shaped this work from the initial problem framing through multiple iterations of the framework and study design, and his detailed reviews and thoughtful feedback consistently pushed me to clarify my arguments and strengthen the empirical grounding of my claims.

I am sincerely grateful to the participating engineers and managers who generously shared their time and on-call experiences. Their insights and stories formed the foundation of this study, and this work would not have been possible without their contributions.

I would also like to thank my colleagues in the MSE program for their peer reviews, discussions, and collaborative feedback throughout the semester. Their engagement—both in formal critiques and countless working sessions—played a meaningful role in refining this thesis.

I owe especially profound thanks to my parents, whose belief in the value of education shaped both my decision to pursue graduate study at Carnegie Mellon University and the path that led me here. Their encouragement made it possible for me to study abroad, and their lifelong support continues to inspire the work I do and the person I strive to become.

Finally, I want to express my deepest appreciation to my wife. Throughout the Master of Software Engineering program, she took on far more responsibilities at home than she ever should have had to, simply so I could devote the necessary time to my studies. Her understanding during the many evenings and weekends consumed by coursework and writing—and her steady support throughout this entire journey—made it possible for me to complete this thesis.

Abstract

Modern cloud systems generate overwhelming amounts of telemetry, making incident diagnosis increasingly difficult and motivating interest in AI-assisted observability. Despite this potential, engineers often hesitate to rely on AI because they distrust data quality, model reasoning, and automated actions during high-stakes incidents. This thesis proposes a four-pillar framework—data quality governance, operational explainability, human-in-the-loop oversight, and feedback loops—to create organizational conditions for trustworthy use of AI in on-call engineering. I evaluate the framework through a mixed-method, within-subjects study in which 10 experienced on-call engineers complete baseline trust assessments, reason through how the framework would apply to realistic incident scenarios, and then repeat the same assessments alongside a follow-up interview. Across all measured dimensions, post-framework trust ratings increase, with the largest gains in understanding AI decisions and willingness to use AI assistance during live incidents. Participants report that explainability and human oversight most directly increase their confidence, while data quality practices and feedback mechanisms are viewed as essential but slower to influence trust. Because the evaluation uses hypothetical scenarios rather than a deployed production system, the results reflect stated confidence rather than observed behavior. Even so, the findings suggest that organizations can meaningfully increase engineers' willingness to adopt AI-assisted practices by embedding transparency, control, and structured learning around these tools.

I. Introduction

Modern cloud-based systems generate overwhelming volumes of logs, metrics, traces, and alerts, making it increasingly challenging for engineers to maintain reliable operations. To cope with this complexity, industry is turning to AIOps – Artificial Intelligence for IT Operations – which applies machine learning and automation to IT monitoring and incident management tasks [1]. The goal of AIOps is to augment or automate activities like anomaly detection, root-cause analysis (RCA), and incident response, shifting teams from reactive “fire-fighting” toward proactive prevention and faster remediation. With AI support, organizations aim to anticipate issues before they impact users, automatically pinpoint causes, and even self-heal systems, thereby improving uptime and reducing on-call strain. This vision is compelling given the scale and speed of modern deployments – for example, distributed microservices and dynamic cloud infrastructure produce complex failure modes that human operators struggle to analyze in real-time [2]. In principle, AI-driven observability and on-call tools could drastically cut downtime and operational costs by handling routine issues and highlighting critical insights beyond human capacity.

1.1 Problem Statement

Despite the hype, industry surveys and adoption data reveal that AI-driven observability is still in its infancy, with a significant gap between interest and trust. As of 2025, only a small minority of enterprises (~10%) have integrated AI into their observability practices, though nearly half plan to evaluate such solutions in the next budgeting cycle [3]. This cautious adoption is largely due to a lack of confidence in AI’s outputs and decisions. Reports find that engineering teams worry about data quality, security, and the opacity of AI recommendations. In one survey of IT professionals, only 38% said they trust the quality of data and training behind AI systems [4]. Similarly, a BARC study of enterprises with data observability programs showed 42% “do not trust the outputs” of their AI/ML models [5]. Even software developers echo this skepticism – for instance, the 2025 Stack Overflow survey found that while 84% of developers use AI coding assistants, only 3% “highly trust” the accuracy of the AI’s output [6]. This trust deficit is a critical barrier: engineering leaders are hesitant to hand over incident management decisions to AI if they themselves cannot explain its reasoning or ensure its reliability. In high-stakes operational environments (e.g. finance, healthcare), blindly trusting an algorithm could lead to outages or safety incidents, so teams default to manual control. The result is under-utilization of potentially valuable AI tools – many companies pilot AIOps features but stop short of full deployment due to uncertainty and lack of proven best practices for trustworthy adoption.

1.2 Claim

This thesis argues that engineering organizations can increase their confidence in AI-assisted observability and incident response by adopting a structured organizational framework centered on four pillars: (1) robust data quality governance, (2) operational AI explainability, (3) calibrated human-in-the-loop oversight, and (4) feedback loops for continuous learning. By instituting these practices, teams will be more willing to rely on AI suggestions and even grant limited autonomy to AI in on-call scenarios, ultimately improving incident management outcomes. The claim is qualified – the framework is expected to build trust and safe adoption of AI given that baseline model performance is reasonable and the organizational culture supports data-driven decision-making. It does not claim that AI will completely replace human operators or that trust will improve if these prerequisites are not met.

1.3 Research Objectives

To investigate this claim, the research addresses the following key questions:

- RQ1: Does the introduction of the proposed framework measurably increase engineers' reported trust in and intended usage of AI-driven observability/on-call tools?
- RQ2: Does the framework lead engineers to report higher acceptance of autonomous AI actions (e.g., automated diagnostics or remediations) without increasing their perceived risk to system safety or reliability?
- RQ3: Which components of the framework (data quality measures, explainability features, human oversight mechanisms, feedback processes) are most influential in building or restoring trust?

These questions are answered through a mixed-method evaluation. In brief, our approach involves collecting empirical data on current AI tool adoption and trust levels (pre-framework surveys), exposing participants to the proposed four-pillar framework via realistic scenarios, and then collecting post-framework surveys along with qualitative interviews about how their trust and intended behaviors would change. By triangulating these evidence sources, we aim to connect the evidence (empirical data on trust and stated usage) to the claim. In other words, if we observe improved trust/uptake corresponding to specific framework elements, it will support the warrant that these organizational practices are effective mediators between AI capabilities and human acceptance. In preview, our approach leverages known links from prior research – for example, the idea that explainability fosters user trust is well-established [7]. Overall, in this study we find that the four-pillar governance and transparency framework addresses engineers' core concerns—data validity, understanding the AI's logic, and retaining control—and is associated with a safer path to realizing AIOps benefits.

II. Background

This section provides the conceptual and technical foundation for our study, defining key terms and outlining the current practices in software reliability engineering. It also highlights the challenges in today's industrial setting that make our problem both important and difficult.

2.1 Observability and On-Call Engineering

Software observability refers to the ability to infer the internal state of a system from its external outputs. In practice, observability is implemented via comprehensive logging, metrics collection, and distributed tracing of transactions across services. Modern observability stacks aggregate telemetry data to give engineers visibility into system health and behavior. Observability is a cornerstone of DevOps and Site Reliability Engineering (SRE) practices, which emphasize monitoring everything in production and acting quickly on anomalies. An on-call engineer is a team member rostered to respond to incidents 24/7 – when alerts trigger (e.g. high error rates, latency spikes), the on-call engineer must investigate and mitigate the issue. On-call work is high pressure: every minute of downtime can impact revenue and user trust, so organizations measure metrics like Mean Time to Detect (MTTD) and Mean Time to Recovery/Repair (MTTR) to assess their incident response efficiency. Traditional incident response involves diagnosing the problem by manually examining dashboards and log files, executing runbook scripts or patches, and later conducting a post-mortem analysis to learn from the failure. Best practices

such as blameless post-mortems and iterative improvement of monitoring are ingrained in SRE culture to continually enhance reliability.

However, several trends have made this process increasingly challenging. According to Notaro et al. [1], modern IT systems have grown too large and complex for manual supervision, resulting in alert fatigue and missed signals. In globally distributed teams, knowledge is siloed – the person on call might not be familiar with a particular subsystem's quirks. Moreover, many organizations have a patchwork of monitoring tools that are not well-integrated, forcing engineers to swivel-chair between consoles. This motivates the use of AI to automatically correlate data and highlight the most relevant information during an incident.

2.2 Artificial Intelligence in IT Operations (AIOps)

Artificial Intelligence in IT Operations (AIOps) applies machine learning to observability data to detect anomalies, correlate alerts, forecast issues, and sometimes trigger remediation. Recently, LLM-based 'incident copilots' have been proposed to summarize logs and suggest fixes. These tools promise faster, more accurate diagnosis, but they also introduce new challenges around data quality, model transparency, and safe automation.

To manage telemetry at scale, industry is also converging on standards and frameworks. One notable example is OpenTelemetry, an open-source standard (under CNCF) that unifies the instrumentation of traces, metrics, and logs [8]. Widespread adoption of OpenTelemetry and similar schemas improves data quality and consistency – a crucial prerequisite for effective AI analysis. Data quality governance in this context means ensuring monitoring data are accurate, timely, and relevant.

2.3 Processes and Standards

Traditional IT service management frameworks (e.g., ITIL, ISO/IEC 20000-1) define structured incident processes, while modern SRE practice layers in automation through CI/CD, rollbacks, and runbooks. However, high-impact actions (like failover or paging) typically still require human approval, and organizations emphasize auditability and error budgets to manage risk. These existing governance practices provide a basis for controlling automation, but say little about how to govern AI-driven decisions.

2.4 Contextual Challenges

Deploying AI in on-call operations faces several challenges unique to the context:

Data Silos and Legacy Systems: Enterprises often have heterogeneous systems (on-premise legacy servers, multiple cloud platforms) emitting telemetry in different formats. Integrating these for AI analysis is non-trivial. A survey by Pal et al. found that integration with legacy systems and inconsistent data formats were major pain points slowing AIOps adoption [9]. This underscores the need for robust data pipelines and cleaning (Pillar 1 of our framework).

Opaque "Black-box" Models: Many AI models, especially deep learning and large language models, act as black boxes that don't explain their reasoning. In operations, this opacity is problematic – an engineer receiving an alert "predicted service outage in 30 minutes" from an AI will hesitate to act without understanding why the model believes so. Techniques like model

interpretability, rule-based explanations, or at least transparent linking to anomalous signals can help translate AI analysis into a form engineers trust (Pillar 2).

Human-in-the-loop Requirements: Operations is ultimately socio-technical – it involves human judgment, especially under uncertainty. Trustworthy AIOps must support a human-in-the-loop approach (Pillar 3) where the AI assists but does not bypass human confirmation for impactful decisions. An example best practice is “approval gates”: the AI can suggest a remediation (restarting a service), but the on-call engineer must approve it, except perhaps in pre-approved safe cases.

Regulatory and Security Constraints: In certain industries (finance, healthcare, critical infrastructure), there are regulations that demand rigorous change control and accountability for any action on production systems. Introducing AI decision-makers raises questions: Who is accountable if an AI causes an outage? Can the AI’s decisions be audited after the fact? This security angle means organizations must implement safeguards around AI recommendations. Our framework’s feedback and review mechanisms (Pillar 4) tie into this by logging AI actions and outcomes for post-incident review, which helps in both auditing and improving the AI.

In summary, the background shows that while the technology ecosystem provides ever more data and some AI capabilities, organizational practices have not caught up to fully leverage AI in a trustworthy manner. Established standards give us reliable processes but little guidance on AI; new AI tools offer power but lack guarantees. This tension sets the stage for our research: bridging the gap with a framework that incorporates data governance, explainability, oversight, and learning – principles aligned with emerging guidelines for trustworthy AI.

III. Related Work

Research and industry efforts at the intersection of AI and software operations provide important context for our thesis. In this section, we review the state-of-the-art solutions and studies, highlight open gaps – particularly around trust and adoption – and clarify how our work is positioned relative to existing knowledge.

3.1 AI for Observability and Incident Response – Potential and Pitfalls

Over the past few years, numerous approaches have been explored to automate parts of the incident management lifecycle. Classical machine learning techniques have shown promise in specific tasks: for example, Sambamurthy [10] reports an AI-based observability system that achieved 92% accuracy in root-cause analysis (RCA) of incidents, compared to about 76% accuracy using traditional (non-AI) methods. Such improvements in diagnostic accuracy can translate to faster resolutions. Similarly, prototype “virtual SRE” agents have demonstrated the ability to reduce mean time to recovery; one study introduced a Cognitive DevOps Assistant that cut MTTR by 44% and reduced false alarms by nearly half [11]. These results illustrate the potential efficiency gains and performance improvements that AI-driven tools can bring to DevOps and SRE workflows. In short, the state of the art suggests that when AI is applied in a targeted way (anomaly detection, correlation, etc.), it can outperform manual analysis in speed and accuracy.

On the other hand, a substantial body of research has identified critical limitations and risks that come with these AI solutions. A common theme is the trust and transparency problem. Sami et al. conduct a deep survey on Explainable AIOps, noting that many AIOps deployments rely on "black-box models" and that this "introduces critical challenges in trust, auditability, and regulatory compliance", ultimately "undermining confidence in automated decisions" [7]. In other words, if an AI can flag an anomaly but cannot explain why or how it got there, engineers are unlikely to trust it for high-stakes decisions. Another issue is the reliability of AI in the wild: large language models, while powerful, are prone to hallucinations (confidently generating incorrect information) and can miss important context if prompts are incomplete. Lin et al. present IRCopilot, an LLM-driven incident response system, and document cases where the model's suggestions were erroneous due to lost context or misinterpreting log data [12]. Their evaluation showed that without strict guardrails, the LLM might recommend wrong fixes – a clear danger in an on-call scenario. Zhang et al. provide a comprehensive survey of AIOps in the era of LLMs and echo these concerns, listing hallucination, high latency and cost, and inability of current LLMs to incorporate certain structured data (like real-time metrics) as key obstacles to operational use [13].

Beyond model issues, security is a growing concern in this space. Pasquini et al. demonstrated an attack where fake telemetry signals were introduced to trick an AIOps system, causing it to take incorrect automated actions [14]. This kind of vulnerability reveals that naive automation can be dangerous – malicious actors might exploit the AI's trust in data unless organizations implement verification steps and fail-safes.

3.2 Trust, Explainability, and Human Factors

Given the pitfalls above, researchers have been investigating how to foster human trust in AI systems for operations. Explainable AI (XAI) techniques are at the forefront of this effort. For instance, Fernando et al. propose an adaptive explainability framework for high-stakes environments, arguing that AI systems should adjust the depth and style of explanations based on the urgency of the situation and the user's stress level [15]. They introduce the notion of "swift trust" in incident response – essentially, how a human operator can rapidly develop a calibrated trust in an AI assistant during an ongoing crisis. The takeaway is that in on-call scenarios, an AI needs to quickly justify its recommendations in a clear, context-aware manner (e.g., showing the chain of events leading to a database failure if it suggests a failover). Other studies emphasize governance and calibration of trust. Khati et al. note that both over-trust and under-trust in AI tools are detrimental [16]. If engineers blindly follow AI suggestions (over-trust), errors can propagate unchecked; if they ignore or second-guess every suggestion (under-trust), the AI's utility drops and it may be abandoned. The solution suggested involves training users on the AI's capabilities/limits and designing the AI to indicate its confidence and allow user feedback – approaches that our framework incorporates via human-in-the-loop controls and feedback mechanisms.

3.3 Gaps and Positioning of This Work

In summary, prior work has made it clear that trust and transparency are the keys for wider adoption of AI in on-call engineering. We have AI techniques that can reduce incidents and workload, but people won't use them (or will use them in a very limited way) if they don't trust them. Existing research has certainly identified this problem and even proposed pieces of the solution – e.g., explainability methods, trust calibration concepts, and the need for oversight. However, there is a gap in how to implement and integrate these pieces into a coherent,

practical framework within an organization. Most academic studies focus on either technical algorithms or abstract frameworks, and most industry surveys diagnose the problem without prescribing concrete remedies beyond high-level advice. For instance, while Fernando et al. [15] provide a theoretical model for adaptive explanations, they do not demonstrate how it would be applied in a real incident management tool. Similarly, Sami et al. [7] highlight the need for explainable AI, but stop short of detailing organizational processes to achieve trust. To date, there has been little empirical evaluation of interventions to increase operator trust in AI over time.

This thesis addresses the above gap by proposing a holistic framework for trustworthy AI-assisted on-call operations. Our work is positioned at the intersection of technical tool design and organizational process: we combine data quality assurance, AI explainability features, human governance policies, and feedback mechanisms into an actionable playbook for teams. Unlike prior approaches that introduce AI tools in isolation, the framework considers how such tools should be onboarded, governed, and iteratively improved within on-call workflows. We contribute new empirical evidence through a within-subjects, pre/post, scenario-based study: by surveying and interviewing practicing engineers before and after exposure to the framework, we observe changes in their reported trust and intended reliance on AI-assisted observability tools. Taken together, our findings provide initial backing for the claim that trust can be operationalized: participants reported greater willingness to delegate low-risk remediations to AI under the framework. In this way, our work extends the state of the art by offering a practical, empirically informed solution to a well-recognized problem and supporting it with multi-faceted evaluation.

IV. Four-Pillar Framework

The intervention we evaluate is a four-pillar organizational framework for trustworthy AI-assisted observability and on-call operations. Rather than introducing a new AI model, the framework defines requirements on how AI tools must be embedded into data pipelines, user interfaces, and team processes before engineers are asked to rely on them. It has four pillars:

(1) data quality governance, which ensures that the telemetry feeding the AI is complete and monitored;

(2) operational explainability, which requires every AI alert or recommendation to surface a human-interpretable rationale;

(3) calibrated human-in-the-loop oversight, which constrains what the AI is allowed to do at different risk levels and guarantees human review and override; and

(4) feedback and continuous learning, which closes the loop by capturing operator feedback and incident outcomes so the AI and the surrounding processes can improve over time.

Each pillar is designed to address a trust barrier identified in the background and related work: doubts about data quality (Pillar 1), opaque “black-box” decisions (Pillar 2), fear of losing control to automation (Pillar 3), and concern that AI errors will accumulate or decisions will drift without correction (Pillar 4).

The framework prescribes a small set of practices under each pillar that teams can implement around any AIOps or incident-assistant tool.

- **Data quality governance:** Teams define minimum telemetry coverage and schema standards (e.g., OpenTelemetry), monitor for gaps or corrupt signals, and treat broken observability as an operational incident in its own right.
- **Operational explainability:** AI outputs are required to expose the key signals and correlations driving each recommendation (e.g., memory saturation on pods X/Y plus recent deployment Z) and, where possible, link back to raw dashboards or logs; alerts without a clear rationale are treated as non-actionable.
- **Human-in-the-loop oversight:** The framework defines explicit automation tiers, from read-only suggestions by default, to auto-generated runbook steps that require human approval for medium-risk actions, to tightly scoped, pre-approved autonomous actions for low-risk fixes, all with audit logs and easy rollback.
- **Feedback and continuous learning:** Incidents involving AI suggestions are reviewed in post-mortems, engineers can rate or correct AI outputs inline, and those signals are fed back into model updates and configuration changes.

In the study, this four-pillar playbook is what participants are asked to reason about and hypothetically apply to real incidents; the subsequent evaluation measures whether this structured bundle of data, oversight, and learning practices changes their reported trust and willingness to use AI during on-call work.

V. Methods

5.1 Research Design & Justification

We employ a mixed-method, within-subjects pre/post study with practicing engineers, but our evaluation of the four-pillar framework is hypothetical rather than deployed in production. Participants first rate their current trust in, and willingness to use, AI-assisted observability tools based on their existing experience (pre-framework baseline). We then introduce the proposed four-pillar trust framework and walk them through a set of concrete, framework-aligned examples (e.g., mock alerts, UI snippets, and process changes), asking them to imagine using an AI tool implemented according to these practices. After this exposure, participants complete the same trust and usage ratings again (post-framework) and take part in a semi-structured interview about how, and why, the framework would change their attitudes and intended behaviors. This mixed-method, scenario-based design still provides triangulation: the pre/post survey scores offer empirical indicators of how the framework would affect trust and willingness to use AI (RQ1/RQ2), while the interviews explain the mechanisms behind those shifts and surface limitations and concerns (RQ3).

5.2 Setting and Participants

Our study recruits 10 practicing on-call engineers from 4 large internet-scale companies, all of whom regularly work with distributed microservices, containerized deployments, and modern observability stacks (logs, metrics, traces, and alerting). Participants have between 5 and 15 years of experience in DevOps/SRE or related on-call roles (median ~8 years) and are familiar with common monitoring and incident response tools. While their teams had begun exploring AI-assisted features (e.g., anomaly detection or automated correlation in existing tools), none were yet relying heavily on AI for critical incident decisions; this made them a good fit for evaluating a trust-building framework. Sessions were conducted in a controlled setting using

realistic, anonymized incident scenarios rather than live production, and participants were informed that the goal was to understand how the proposed four-pillar framework would affect their trust and intended usage of AI in their on-call workflows. This setting provides an ecologically grounded, though small-scale, view into how experienced engineers reason about adopting AI in observability and incident management.

5.3 Data Sources

To capture the impact of the framework, we collected 2 types of data, each mapped to our research questions and pillars of trust:

- **Surveys:** We developed a structured questionnaire to quantitatively gauge engineers' trust in AI-driven observability tools and their willingness to rely on or use these tools. The survey included Likert-scale items adapted from established trust-in-automation scales and custom questions targeting our context. Participants completed this survey twice within a single study session – once before the framework introduction (baseline) and again after the framework walkthrough and scenario exercises – to measure any change in their reported trust levels and intended adoption of a framework-aligned AI tool.
- **Interviews and Qualitative Feedback:** We conducted semi-structured interviews with each participant at two points in the same session, pre- and post-framework exposure. In the initial interview, we explored their current observability practices, any existing AI/automation tools, and their feelings of trust or mistrust ("Can you recall a time an AI-based alert was wrong or hard to believe? What did you do?"). In the follow-up interview, we asked how, in their view, the framework's elements would affect their workflow if implemented (for example, "If AI alerts came with the kinds of explanations we showed, would that change how you respond?"), and which pillars or features they expected to be most and least useful. Each interview lasted ~30 minutes.

5.4 Procedures

The study follows a conceptual mixed-method design conducted over approximately 3 weeks, combining simulated case analysis and expert evaluation. Instead of implementing the framework directly in production systems, we model and evaluate its hypothetical application through structured discussions, scenario walkthroughs, and comparative assessments by experienced SREs and engineers.

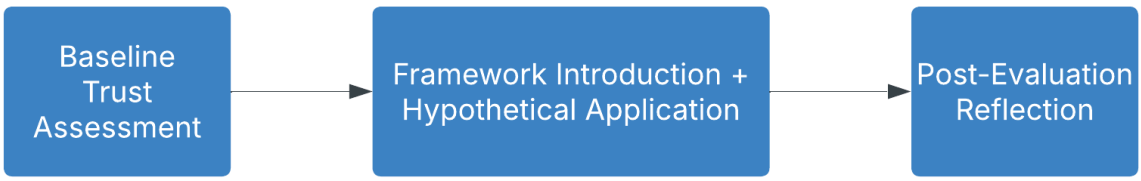


Figure 1 *illustrates the study's structure: (1) Baseline Trust Assessment, (2) Framework Introduction and Hypothetical Application, and (3) Post-Evaluation Reflection.*

5.4.a Baseline Assessment

Participants completed a pre-study survey and interview capturing their current use of AI-assisted observability tools, perceived data reliability, and levels of trust or skepticism. With an NDA and organizational consent, participants described what went wrong, how automation or AI assisted, and where confidence failed. These discussions surfaced real pain points around data gaps, unexplained alerts, and limited confidence in automation, forming a grounded baseline for later reflection.

5.4.b Framework Introduction and Hypothetical Application

Participants were introduced to the four-pillar trust framework, including conceptual explanations, examples from prior research, and simple visual diagrams. Instead of direct implementation, participants hypothetically applied the framework to the same real incidents they had discussed earlier. For each case, they were prompted to reason through questions such as:

- “If you had automated telemetry validation in place, how might detection have changed?”
- “If the AI could show its reasoning for an alert, would that have increased your confidence?”
- “If the AI had an approval gate before executing an action, would you have accepted it?”
- “If post-incident feedback were integrated, would you feel more ownership of the system’s behavior?”

Participants evaluated how each pillar might alter their workflow, trust level, or willingness to rely on AI systems, without requiring live deployment.

5.4.c Post-Evaluation Reflection

After completing the framework walkthroughs, participants filled out a post-survey and participated in a follow-up interview. The survey repeated core trust and adoption items from the baseline and added ratings for each pillar’s perceived usefulness.

In the interviews, participants shared their before-and-after perceptions, identifying which pillars felt most critical or realistic and where residual skepticism remained. They also discussed how their attitudes toward AI autonomy had shifted — for instance, whether they would now consider allowing AI to perform low-risk automated remediations.

5.5 Analysis Methods

Because this study evaluated the hypothetical application of the four-pillar trust framework rather than deploying it in production, analysis focuses on perceived change in trust and reasoning patterns across the three phases. Quantitative measures provide directional evidence of trust shifts, while qualitative analyses explain the reasoning behind those shifts. Together, they connect the data to the warrant.

5.5.a Quantitative Analysis

Survey Data: The pre- and post-surveys included parallel Likert-scale questions (1 = Strongly Disagree to 5 = Strongly Agree) about trust, perceived reliability, and willingness to adopt

AI-assisted observability tools. Each participant's before/after responses were paired to measure directional change in:

- Overall trust in AI recommendations
- Perceived data reliability and transparency
- Willingness to approve or delegate AI-suggested actions
- Confidence in feedback and learning mechanisms

Analysis relied on descriptive statistics including means and medians. These results were interpreted qualitatively (e.g., "most participants rated explainability higher post-framework"), emphasizing trends rather than statistical generalization.

Cross-Pillar Ratings: In the post-survey, participants rated the perceived usefulness of each pillar. Average pillar scores were compared to identify which mechanisms (data quality, explainability, oversight, feedback) participants believed would most improve trust and safe adoption.

5.5.b Qualitative Analysis

Structured Interview Summaries: We summarized interview content under a predefined template aligned with the four pillars. For each participant, concise notes captured:

- Key examples of incidents discussed
- Hypothetical changes if the framework were applied
- Reported shifts in trust or perceived reliability
- Specific mentions of effective or weak pillars

These summaries were aggregated across participants to count and compare recurring observations. For instance, if a majority of participants mentioned explainability as the most trust-enhancing factor, that pattern was recorded as a consensus trend, while dissenting views were noted qualitatively as outliers or rebuttals.

Comparative Reasoning Patterns: Each participant's "before vs. after" reflections on their real incidents were compared to identify reasoning shifts, focusing on three aspects:

- Confidence Change: whether participants indicated they would act faster or trust the AI more under the hypothetical framework
- Delegation Threshold: whether they would permit greater AI autonomy
- Safety Assurance: whether governance or oversight features would increase their comfort with automated responses

These comparisons were summarized in short analytical memos, which were then synthesized to highlight alignments across participants.

Integrative Interpretation: Results from surveys and summarized interviews were integrated into a matrix of evidence:

- Data: quantitative trust and pillar ratings, qualitative analysis
- Warrant: prior literature suggesting explainability, oversight, and governance improve trust
- Claim: adoption of the four-pillar framework increases trust and safe acceptance of AI assistance

If the majority of participants' reported trust and confidence increased in the post-evaluation phase and they attributed those changes to framework elements, then the data supported the warrant. Any skepticism or disagreement served as a contextual rebuttal, indicating where the framework's assumptions may not generalize.

VI. Results

6.1 Pre- vs Post-Study Ratings

10 participants completed surveys before and after experiencing the four-pillar trust framework. The survey included six Likert-scale items (1 = Strongly Disagree, 5 = Strongly Agree) measuring different aspects of trust in AI-assisted observability tools. Across all six dimensions, ratings increased from the pre-study to post-study survey. Every participant's score on each item either stayed the same or increased after seeing the framework (no decreases were observed). We report mean pre- vs. post-study ratings on six trust dimensions (higher values indicate greater trust or agreement). All dimensions show an increase in post-study ratings.

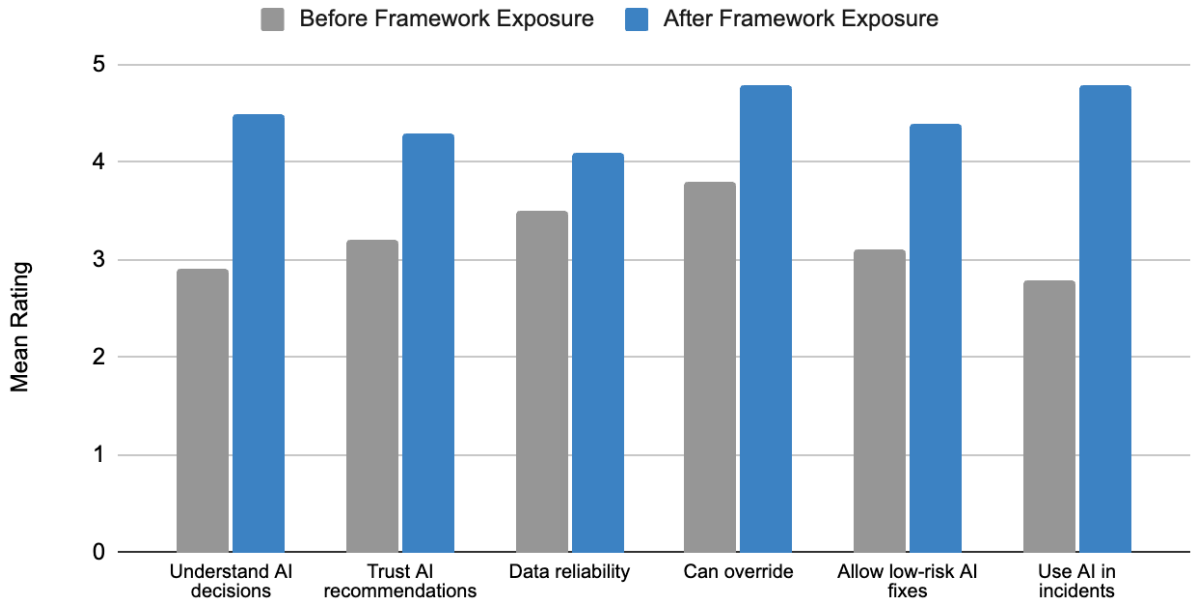


Figure 2: *Changes in Trust-Related Ratings Before and After the Framework.*

Key changes in ratings from pre- to post-study include:

- Understanding of AI decisions: Increased markedly from a mean of 2.9 to 4.5 out of 5, indicating much greater self-reported understanding of why the AI makes its suggestions.
- Trust in AI recommendations: Mean rating increased from 3.2 to 4.3.
- Perceived data reliability: Increased from 3.5 to 4.1.
- Ability to review/override AI actions: Rose from a mean of 3.8 before to 4.8 after (all participants strongly agreed post-study that they could review or override the AI's actions).

- Willingness to let AI handle low-risk fixes: Increased from 3.1 to 4.4 on average, showing greater comfort with letting the AI automate routine, low-risk tasks.
- Willingness to use AI in live incidents: Increased from a mean of 2.8 to 4.8, with almost all participants strongly agreeing post-study that they would use AI assistance during real incidents.

In summary, participants reported higher trust in the AI tool on every measured dimension after evaluating the trust framework. Notably, post-study mean responses were near the maximum rating (5.0) on two critical items: the ability to override AI actions and the willingness to use AI during incidents. This suggests that, with the framework in place, participants felt much more confident in both controlling the AI and relying on it in high-stakes situations.

VII. Discussion

7.1 Quantitative Shifts in Trust and Confidence – A Closer Look

Understanding of AI decisions and willingness to use AI in live incidents saw the largest improvements – rising from mean scores of 2.9 and 2.8 respectively (pre-framework) to 4.8 (post-framework). Participants initially neither agreed nor disagreed that they understood AI alerts; after the framework's explainability features, they almost strongly agreed they understood the AI's logic. Similarly, willingness to use AI during real incidents jumped from a hesitant neutral stance to close to full agreement, suggesting the framework greatly increased comfort in leveraging AI under high-stakes conditions.

Other trust measures also improved. Trust in AI recommendations rose from a neutral baseline (mean = 3.2) to a positive level (mean 4.3) – 8 of 10 engineers moved from neutral or low trust to agreeing that they trust the AI's suggestions. Perceived ability to review or override AI actions increased notably (from mean 3.8 to 4.8); in fact, after learning about the human-in-the-loop oversight pillar, every participant strongly agreed they could review or veto the AI's actions if needed. This unanimous post-study confidence in override control highlights the importance of the oversight mechanisms in assuaging fears of unchecked automation. Willingness to let the AI handle low-risk fixes (like auto-restarting a non-critical service) also climbed from a cautious average of 3.1 to about 4.4. Most participants became comfortable delegating routine "easy win" tasks to the AI after the framework, whereas before they were skeptical or only mildly agreeable to that idea.

The dimension that showed a relatively smallest change was perceived data reliability. The average agreement that "the monitoring data the AI uses is reliable" moved from 3.5 before the framework to around 4.1 after the framework. This suggests that merely introducing a data quality pillar conceptually did not immediately boost confidence in the data itself; participants may feel actual improvements in data governance need to be demonstrated before their rating moves.

Overall, all trust metrics improved, with "Understanding AI decisions" and "Using AI in live incidents" improving the most, and data reliability the least. These results indicate that the framework's emphasis on explainability and oversight had an especially strong effect on user confidence, while trust in data quality may require actual data improvements or proof of reliability.

7.2 Qualitative Insights: Why Trust Improved

Qualitatively, participants' comments support these quantitative shifts. Before exposure to the framework, several engineers described skepticism and slow responses to AI alerts. For example, Engineer A admitted that if an AI-driven alert lacked explanation, they would "hesitate to act without understanding why the model believes so," often double-checking information (such as dashboards or system logs) manually. Such hesitation aligns with the low pre-study scores on understanding and trust. Another participant noted that past false alarms or poor data quality had made them cautious – Engineer B said they had encountered noisy AI recommendations based on incomplete telemetry, leading them to distrust the data feeding the AI. These baseline attitudes resulted in neutral or low trust scores initially.

After the framework walkthrough, however, participants felt many of their concerns were addressed. Engineer A explained that knowing the AI's reasoning "changes the game – I'd be much quicker to trust its alert if I see why it flagged something." This sentiment was common among all the participants who showed the biggest jumps in the understanding metric. Likewise, the guarantee of human oversight gave participants confidence to use the AI more boldly. Engineer C, who was initially uncomfortable letting the AI act on its own, noted that with oversight in place they are "much more comfortable letting the AI handle a routine fix, because I know I can always intervene or roll it back if needed". This reflects the post-study consensus (4.8 mean) that they can review/override AI actions at will.

In summary, the quantitative gains in trust metrics are corroborated by participants' qualitative feedback: the framework directly addressed key pain points (opacity of AI decisions, fear of losing control, etc.), thereby measurably increasing their trust and willingness to rely on the AI tool.

7.3 Perceived Usefulness of Each Trust Pillar

After the framework introduction, participants evaluated the usefulness of each of the four pillars – data quality governance, explainability, human oversight, and feedback loops – in building their trust. All pillars were generally viewed favorably, but not all were seen as equally useful or immediate. When asked to single out which pillar most improved their confidence, responses skewed toward explainability.

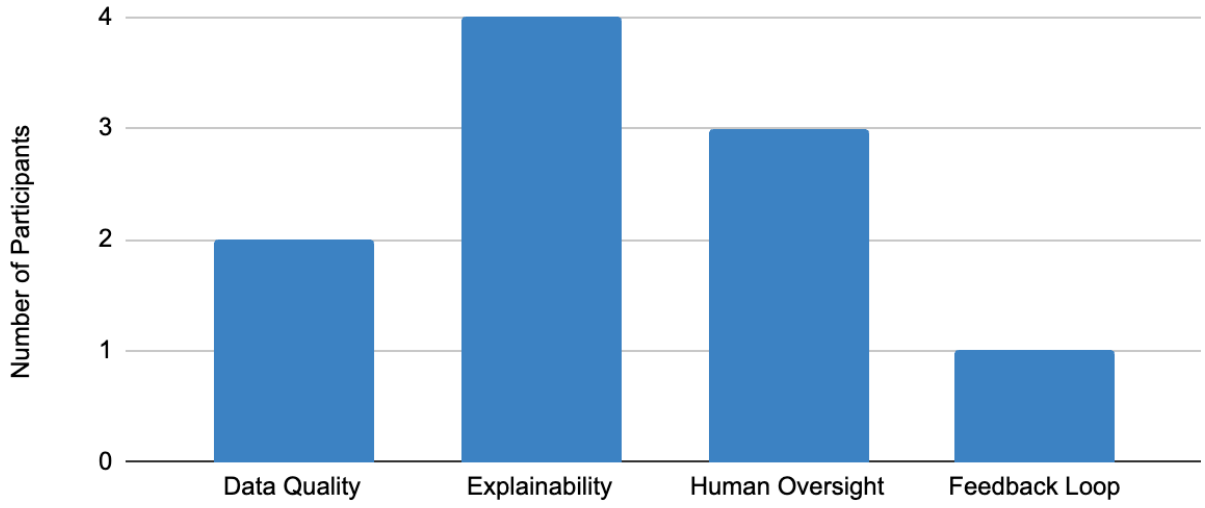


Figure 3: *Participants' Choices of the Most Confidence-Boosting Pillar.*

The chart above shows how many of the 10 participants identified each pillar as the most useful for improving their trust in the AI tool. Explainability was the top choice for a majority (4 out of 10) of participants – the most frequently cited pillar – whereas human oversight and data quality governance were named by fewer participants (3 and 2, respectively). Notably, only one of the participants chose the feedback loop pillar as the most confidence-improving.

This distribution suggests that making the AI's reasoning transparent (pillar 2) was perceived as the most critical factor in trust-building, more so than improvements to input data quality or the promise of model learning over time.

It's important to note that although only 3 people picked "Human-in-the-loop oversight" as their top pillar, others still rated it highly – in fact, all participants strongly agreed that having manual approval gates is useful. The dominance of explainability in the "pick-one" choice reflects that, when forced to choose, participants felt understanding why the AI makes decisions is the biggest game-changer for trust. Engineer A, for instance, commented that without clear explanations, "it doesn't matter how good the data is – I won't trust a recommendation if I can't see some rationale behind it." This aligns with the weight given to explainability. Meanwhile, 2 engineering managers (Engineer D, Engineer E) selected data quality governance as most important, emphasizing that "if the data feeding the AI isn't trustworthy, nothing built on it will be trustworthy". This participant had earlier expressed concern about noisy metrics; for them, the commitment to robust data validation and filtering was the key confidence booster.

The feedback loop pillar (continuous learning from operations) received the lowest enthusiasm. Participants generally agreed it was a good idea but saw it as more long-term. In interviews, a couple of engineers noted that while feedback mechanisms (like learning from mistakes or operator input) are valuable, they "don't immediately increase my trust in the moment". For example, Engineer B likened the feedback pillar to "training wheels that eventually make the AI better for future teams" – beneficial for sustained improvement but not something that directly assures them during an ongoing incident. This perspective helps explain why only 1 participant chose feedback loops as the top factor, even if they didn't rate it negatively. In fact, participants did acknowledge all four pillars as useful to some extent.

In summary, the framework pillars most credited for building trust were those that increase transparency and ensure control. Participants overwhelmingly felt that seeing why the AI is making a recommendation (Pillar 2) and having the ability to approve or override the AI's actions (Pillar 3) directly addressed their trust concerns. Strong data quality practices (Pillar 1) were also considered important. The feedback pillar (Pillar 4), while conceptually positive (helping the AI improve over time via user feedback or learning from outcomes), did not tangibly move the needle on short-term trust for these users. As we discuss next, these nuances have implications for how organizations might prioritize interventions to build trust in AI-assisted operations.

7.4 Framework Efficacy and Support for the Claim

The study results largely support the thesis claim that a four-pillar framework can increase engineers' confidence in AI-assisted observability and on-call tools. Every participant showed a positive shift in trust after being exposed to the framework, with particularly strong gains in their understanding of the AI's decisions and willingness to rely on the AI during incidents. These improvements directly align with the framework's intentions: by instituting data quality governance, AI explainability, human-in-the-loop oversight, and feedback loops, the organization addressed the very factors that prior literature and our participants identified as trust barriers (poor data, opaque logic, lack of control).

Crucially, the two pillars that participants found most useful – explainability and human oversight – correspond to well-known trust enablers in automation. The fact that all participants strongly agreed they could oversee/override the AI and that most felt the AI's reasoning was now understandable demonstrates that the framework delivered meaningful changes in perception. These changes led directly to behavioral intent shifts: by the post-study interview, several engineers said they would respond faster to AI alerts or consider allowing the AI to handle certain tasks that they would not have before.

That said, the evidence is nuanced and does not imply that all trust issues vanished. Importantly, the pillar addressing data quality did not produce an immediate leap in confidence. While participants appreciated that data governance was part of the framework, some remained cautiously "on the fence" about data reliability until proven. This indicates partial support: the framework as a whole builds trust, but certain components may need more time or tangible outcomes to fully earn engineers' confidence. Similarly, only one participant singled out the feedback loop pillar as a game-changer in the short term – again not contradicting the claim but suggesting that its trust benefits might appear more gradually.

7.5 Alternative Explanations and Rebuttals

We must consider whether factors other than the framework could explain the observed trust gains. One alternative interpretation is a Hawthorne effect [17] or expectation bias: participants knew the study was about improving trust in AI, so they might have been inclined to report higher trust simply because they expected an intervention to help or wanted to show a positive outcome. For instance, engineers might have felt pressure (even if unintended) to appear more trusting post-intervention, especially since the researcher walked them through the framework. We tried to mitigate this by emphasizing honest feedback, and indeed participants still voiced some skepticism (e.g. about data quality), suggesting they did not just tell us what we wanted to hear. Nonetheless, we cannot fully rule out that some of the increased ratings were influenced by participants' awareness of the study's goals or their desire to be seen as open-minded to new practices.

Another possible rebuttal is that time and familiarity alone improved trust, independent of the framework itself. All participants spent time discussing AI tools and trust issues, and simply by reflecting on their experiences and seeing any structured approach, they might have grown more comfortable with AI assistance. In other words, perhaps any well-organized discussion or training could have boosted confidence, not necessarily the specific four pillars proposed. This is a valid point—prior studies show that education and exposure often build trust by reducing uncertainty. However, participants' feedback was specific about which framework elements influenced them; they frequently cited the explanation feature or approval process as reasons for increased trust. This suggests that the framework's content, not just familiarity, drove the change. Still, a longer trial with a control group (e.g. another team that only receives generic AI training without the four pillars) would be needed to conclusively rule out these effects.

One could also argue that our sample of participants may have been predisposed to embracing AI, making the trust increase somewhat inevitable once key issues were addressed. All participants were from organizations already interested in AIOps and had several years of experience—they were neither complete skeptics nor novices. A skeptic might say: "These engineers wanted AI to work; once you gave them a framework that checked some boxes (data, explainability, etc.), they jumped on board." This is a fair point and relates to external validity. The framework may have been "preaching to the choir" to some extent—helping those already inclined to trust AI once conditions were right. It was never meant to convert adamant detractors, but to guide organizations willing to adopt AI in a trustworthy way. However, the claim could be challenged in cases where baseline attitudes are more negative – that remains to be tested in future work.

Finally, some might question whether all four pillars are necessary. Since explainability and oversight showed the most visible impact, one could argue that these two alone are sufficient, while data quality and feedback loops are secondary. Our data do show a ranking of pillar influence, but we caution against assuming any pillar is dispensable. Participants valued all four—for instance, even if data quality didn't immediately shift perceptions, everyone agreed it was a critical foundation, and feedback loops, though slower to show results, were seen as key to sustaining trust over time. At the same time, this study does not definitively establish that all four pillars are strictly required. Future work could more rigorously test the necessity of each pillar—for example, by selectively adding or removing pillars in controlled deployments—or refine and operationalize the pillars in ways that make their value more noticeable and necessary to practitioners.

7.6 Qualifiers and Boundary Conditions

It is important to clarify the scope and qualifiers of our conclusions, consistent with the thesis's qualified claim. First, the framework assumes a reasonable baseline of AI competence. It can help users trust an AI tool, but if the system performs poorly or erratically, no amount of explainability or oversight will fully restore trust. In our study, the AI systems were at least somewhat useful—the framework's role was to make their reasoning clearer and minimize risk from errors. It is not a magic wand to make a bad AI seem good; its benefits depend on basic model quality.

Second, the framework presumes an organizational culture open to data-driven improvement. Our participants worked in supportive environments with management buy-in. In highly change-averse or punitive cultures, adoption might falter without leadership endorsement for data quality, explainability, or oversight processes.

Moreover, our discussion presumes that trust as measured by self-reported Likert agreement indeed correlates with actual usage behavior. A qualifier here is that expressed willingness to use AI (post-survey) must translate into real willingness when an incident happens at 3 AM. We believe the strong positive shift is meaningful, but until those engineers are in the field with a framework-enabled tool, we can't be certain they will act in accordance with their survey responses. Thus, the framework demonstrably builds stated confidence and willingness, but validating behavioral trust will require longer-term observation. This does not contradict the thesis claim but nuances it – the claim is about increasing willingness and confidence, which we have shown, but the realization of that trust in practice is an area for further examination.

VIII. Conclusion

This study provides initial evidence that the four-pillar framework offers a practical and promising path to strengthen trust in AI for observability and on-call contexts. Addressing data quality, explainability, oversight, and feedback can meaningfully shift engineers from cautious acceptance to confident use of AI tools. By embedding these principles, organizations and researchers can move closer to achieving AI-augmented operations that enhance reliability without compromising human judgment or accountability.

8.1 Implications for Industry Adoption

The positive results of this pilot study have several implications for organizations looking to adopt AI in their observability and on-call practices. First, it suggests that companies should not deploy AI tools in a vacuum – they need to invest in the surrounding infrastructure of trust. The framework tested here provides a playbook with the four pillars. Implementing these pillars requires collaboration across data, UX, ML, and management teams, but the payoff is higher trust, engagement, and real operational benefits. In short, organizations must "operationalize trust" alongside deploying AI.

Second, the finding that explainability and oversight were most valued implies that tool vendors and platform teams might prioritize those features when introducing AI into incident management. Examples include an "explain" button showing why alerts were raised, or approval interfaces that let engineers supervise automated fixes. This human-in-the-loop design builds comfort and prevents early trust erosion. A phased rollout—starting with recommendations, then supervised automation, and later limited autonomy—can help teams gain confidence gradually.

Third, the lack of immediate enthusiasm for the feedback loop pillar suggests companies should temper expectations about continuous learning features. While in theory a system that learns from every incident and operator comment is great, in practice engineers might not see the benefit right away and thus may not engage with the feedback mechanism. Closing this loop visibly fosters a virtuous cycle: feedback improves performance, which builds more trust and engagement. Organizations should monitor trust over time through surveys or sentiment tracking, treating trust metrics as seriously as operational KPIs like MTTR.

8.2 Future Research Directions

While this study provides encouraging evidence for the framework's effectiveness, it also raises important questions for future research. A clear next step is to conduct larger-scale evaluations

with more participants across diverse teams—and ideally with a control group—to validate which effects generalize. It would be valuable to test whether explainability remains the dominant factor in other contexts, or if domains like aerospace or healthcare require additional pillars such as security or compliance. Future work could also quantify each pillar's contribution through statistical modeling, identifying which factors most strongly predict trust gains.

Another direction is to move from stated intent to observed behavior. A longitudinal deployment study, where teams use an AI tool incorporating the framework over several months, could reveal how trust translates into real actions—such as faster responses to AI alerts or greater willingness to approve automated fixes. Measuring operational outcomes like reduced outages or shorter resolution times would strengthen the evidence linking trust to performance. Such studies could also track resilience: whether trust holds steady or declines after AI errors, and whether retraining or reorientation sessions are needed to restore confidence.

8.3 Closing Thoughts

Ultimately, operationalizing trust in AIOps is as much a socio-technical challenge—spanning people, process, and governance—as it is a modeling problem. By making those dimensions explicit and testable, this thesis aims to provide a starting point for teams who want to build AI-augmented operations they can genuinely trust.

References

[1] P. Notaro, J. Cardoso, and M. Gerndt, “A systematic mapping study in AIOps,” arXiv preprint arXiv:2012.09108, Dec. 15, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2012.09108

[2] T. Nolle, “Improve observability with AI: 5 real-world success stories,” Search Cloud Computing, Sept. 2, 2025. [Online]. Available: https://www.techtarget.com/searchcloudcomputing/tip/Improve-observability-with-AI-Real-world-success-stories

[3] K. Lange, “IT and technology spending & budgets for 2025: Trends & forecasts,” Splunk Blogs, Nov. 25, 2024. [Online]. Available: https://www.splunk.com/en_us/blog/learn/it-tech-spending.html

[4] APMdigest, “IT Pros Want AI and AIOps but Are Concerned About Data Quality,” APMdigest, Jun. 25, 2024. [Online]. Available: https://www.apmdigest.com/it-pros-want-ai-and-aiops-but-are-concerned-about-data-quality

[5] Ataccama, “New BARC analyst report reveals what’s missing in enterprise AI trust strategies,” Ataccama News, June 17, 2025. [Online]. Available: https://www.ataccama.com/news/new-barc-analyst-report-reveals-whats-missing-in-enterprise-ai-trust-strategies

[6] M. Stanek, “AI Is Everywhere, But Trust in It Is Nowhere,” Practical Engineering Management, Aug. 11, 2025. [Online]. Available: https://www.practicalengineering.management/p/ai-is-everywhere-but-trust-in-it-is-nowhere

[7] M. A. Sami, A. Rehman, Z. Ahmad, and N. Bano, “Explainable AIOps: A deep survey on trustworthy and transparent AI in cloud-scale DevOps automation,” Spectrum of Engineering Sciences, vol. 3, no. 7, p. 488, July 12, 2025. DOI: 10.5281/zenodo.15867960. [Online]. Available: https://doi.org/10.5281/zenodo.15867960

[8] A. Thakur and M. B. Chandak, “A review on OpenTelemetry and HTTP implementation,” International Journal of Health Sciences, vol. 6, no. S2, pp. 15013–15023, Jun. 14, 2022. DOI: 10.53730/ijhs.v6nS2.8972. [Online]. Available: https://doi.org/10.53730/ijhs.v6nS2.8972

[9] D. Pal, A. Mehta, and R. Jain, “AIOps: Integrating AI and Machine Learning into IT Operations,” Int. J. of Latest Research in Projects, 2024. [Online]. Available: https://www.researchgate.net/publication/389136333

[10] P. Sambamurthy, “Advancing Systems Observability Through Artificial Intelligence: A Comprehensive Analysis,” IRJMETS, vol. 6, no. 7, 2024. [Online]. Available: https://doi.org/10.56726/IRJMETS60598

[11] X. Daga, V., Daga, P., & Dubey, S., “Cognitive Agents in DevOps: Transforming Observability, Incident Response, and Policy-Aware Automation,” Int. J. of Recent Research and Review, Special Issue 2, 2025. [Online]. Available: https://www.ijrrr.com/specialissues2-2025/ijrrr-Special-Issue-2-2025-paper39.pdf

[12] X. Lin, J. Zhang, G. Deng, T. Liu, X. Liu, C. Yang, T. Zhang, Q. Guo, and R. Chen, “IRCopilot: Automated Incident Response with Large Language Models,” arXiv preprint arXiv:2505.20945v2, Aug. 7 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2505.20945

[13] L. Zhang, T. Jia, M. Jia, Y. Wu, A. Liu, Y. Yang, Z. Wu, X. Hu, P. S. Yu, and Y. Li, “A Survey of AIOps in the Era of Large Language Models,” arXiv preprint arXiv:2507.12472v1, Jun. 23, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2507.12472

[14] D. Pasquini, E. M. Kornaropoulos, G. Ateniese, O. Akgul, A. Theocharis, and P. Efstathopoulos, “When AIOps Become ‘AI Oops’: Subverting LLM-driven IT Operations via Telemetry Manipulation,” arXiv preprint arXiv:2508.06394v2, Aug. 27, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2508.06394

[15] N. Fernando, B. Nakisa, A. Ahmad, and M. N. Rastgoo, “Adaptive XAI in High Stakes Environments: Modeling Swift Trust with Multimodal Feedback in Human-AI Teams,” arXiv preprint arXiv:2507.21158, Jul. 2025. [Online]. Available: https://arxiv.org/pdf/2507.21158

[16] D. Khati, Y. Liu, D. N. Palacio, Y. Zhang, and D. Poshyvanyk, “Mapping the Trust Terrain: LLMs in Software Engineering — Insights and Perspectives,” arXiv preprint arXiv:2503.13793v2, Mar. 18, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2503.13793

[17] R. McCarney, J. Warner, S. Iliffe, R. van Haselen, M. Griffin, and P. Fisher, “The Hawthorne Effect: a randomised, controlled trial,” BMC Medical Research Methodology, vol. 7, no. 30, 2007. [Online]. Available: https://doi.org/10.1186/1471-2288-7-30

Appendix

Appendix 1 - Survey Questionnaire

Each item uses a 5-point Likert scale (1 = Strongly Disagree, … 5 = Strongly Agree) unless noted.

Questions asked in BOTH pre-study and post-study

1. I trust the recommendations produced by AI-assisted observability / on-call tools.
2. I believe the monitoring / telemetry data these AI tools use is reliable enough to act on.
3. When an AI tool raises an alert or suggestion, I understand why it's doing that.
4. I feel I can review or override AI-suggested actions before they affect production.
5. I would be comfortable letting AI automatically handle low-risk fixes (e.g., restarting a non-critical service).
6. I am willing to use AI assistance during live incident response.

Questions asked ONLY at pre-study (baseline context)

1. In the past month, I have acted on an AI-generated alert or recommendation.
2. When AI/automation was wrong in the past, it reduced my willingness to use it again.
3. (Open-ended) Briefly describe one situation where AI/automation helped or hurt during an incident.

Questions asked ONLY at post-study (after framework exposure)

1. The four-pillar framework (data quality, explainability, human approval, feedback/learning) would increase my confidence in using AI during incidents.
2. After seeing the framework, I would support giving AI more autonomy for well-defined, low-risk actions.
3. (Pick one) Which pillar most improved your confidence?
    - Data quality governance
    - Explainability / show reasoning
    - Human approval / oversight gates
    - Feedback & continuous learning

Appendix 2 - Interview Guide

Pre-Study Interview (Before Framework)

Goal: establish baseline perceptions of trust, experience, and pain points.

1. Tell me about a recent incident or alert that was difficult to handle — what made it challenging?
2. Did any AI or automation features help (or hurt) during that incident? How did you decide whether to trust them?
3. What kinds of data or signals do you rely on most when debugging? Where do you usually notice data quality problems?
4. When tools surface anomalies or root causes, what kind of explanation or visualization actually helps you believe the result?
5. What changes—either in tools, data, or process—would make your incident response work easier or more trustworthy?

Post-Study Interview (After Framework Exposure)

Goal: capture reflections on the four-pillar framework and any change in trust or willingness to adopt.

1. Thinking of the incidents you described earlier, how might the framework have changed your investigation or decision process?
2. Which parts of the framework felt practical to implement in your environment, and which felt unrealistic or heavy?
3. Did anything in the framework make you look at existing AI tools differently—for example, how you interpret their outputs or review their data?
4. What support or policy changes would your team or company need before applying this framework in production?
5. If you could simplify or improve one aspect of the framework to make it more usable, what would you change?

Appendix 3 - Survey Responses

Pre-study Survey

#	Role / team context:	In the past month, I have acted on an AI-generated alert or recommendation.	When an AI tool raises an alert or suggestion, I understand why it's doing that.	I trust the recommendations produced by AI-assisted observability / on-call tools.	I believe the monitoring / telemetry data these AI tools use is reliable enough to act on.	I feel I can review or override AI-suggested actions before they affect production.	I would be comfortable letting AI automatically handle low-risk fixes (e.g., restarting a non-critical service).	I am willing to use AI assistance during live incident response.	When AI/automation was wrong in the past, it reduced my willingness to use it again.
1	Engineering Manager	2	3	3	4	3	3	3	3
2	Software Engineer	4	3	4	4	4	3	4	3
3	Engineering Manager	4	2	3	3	3	4	3	3
4	Software Engineer	3	2	3	2	4	3	3	4
5	Software Engineer	2	2	3	3	4	2	2	4
6	Software Engineer	5	4	3	3	4	4	3	3
7	Engineering Manager	1	4	4	4	5	4	4	2
8	Software Engineer	3	3	3	4	4	3	3	4
9	Software Engineer	2	3	3	4	3	2	1	4
10	Software Engineer	4	3	3	4	4	3	2	3
	Mean	**3**	**2.9**	**3.2**	**3.5**	**3.8**	**3.1**	**2.8**	**3.3**

Post-study Survey

#	Role / team context:	When an AI tool raises an alert or suggestion, I understand why it's doing that.	I trust the recommendations produced by AI-assisted observability / on-call tools.	I believe the monitoring / telemetry data these AI tools use is reliable enough to act on.	I feel I can review or override AI-suggested actions before they affect production.	I would be comfortable letting AI automatically handle low-risk fixes (e.g., restarting a non-critical service).	I am willing to use AI assistance during live incident response.	The four-pillar framework (data quality, explainability, human approval, feedback/learning) would increase my confidence in using AI during incidents.	After seeing the framework, I would support giving AI more autonomy for well-defined, low-risk actions.	(Pick one) Which pillar most improved your confidence? * Data quality governance * Explainability / show reasoning * Human approval / oversight gates * Feedback & continuous learning
1	Engineering Manager	4	4	4	5	4	5	5	5	Human approval / oversight gates
2	Software Engineer	5	5	4	5	5	5	5	5	Explainability / show reasoning
3	Engineering Manager	5	4	4	5	4	5	4	4	Data quality governance
4	Software Engineer	5	5	3	5	5	5	5	5	Explainability / show reasoning
5	Software Engineer	5	4	3	5	4	5	5	5	Explainability / show reasoning
6	Software Engineer	4	4	5	4	4	5	5	5	Feedback & continuous learning
7	Engineering Manager	4	5	5	5	5	5	4	5	Data quality governance
8	Software Engineer	5	4	4	5	5	5	5	5	Human approval / oversight gates
9	Software Engineer	4	4	5	4	4	4	5	5	Human approval / oversight gates
10	Software Engineer	4	4	4	5	4	4	5	5	Explainability / show reasoning
	Mean	**4.5**	**4.3**	**4.1**	**4.8**	**4.4**	**4.8**	**4.8**	**4.9**	

Evaluating the Effect of Process Change on the Performance of an Agile Software Maintenance Team Using Discrete Event Simulation

Joe Yoon
School of Computer Science
Carnegie Mellon University

Abstract

Software Maintenance is a costly and difficult task for every organization that relies on mission-critical software to conduct its business. This activity is distinct from other types of software engineering and has historically not received as much attention from the academic community. The nature of software maintenance work requires effective service management and queue management techniques rather than typical project management. A literature review reveals that well-known Agile methodologies like Kanban and Scrum can be used to successfully manage software maintenance, but that some level of adaptation and a combination of different approaches are often required. This study uses a Discrete Event Simulation-based experiment to explore these issues in the context of a software maintenance team at Strivent, a medical device manufacturing company. Using two different simulation models for comparison, the results of the simulation show that a "Scrumban"-based process that combines the work-batching paradigm of Scrum with the continuous flow-maximizing approach of Kanban can increase overall team productivity, provided that the Scrum process produces a sufficient improvement in task efficiency that compensates for the loss in productivity caused by introducing additional process overhead.

Acknowledgements

I would like to express my sincere gratitude to my thesis advisors, Dr. Eduardo Miranda and Dr. Bradley Schmerl, for the guidance, expertise and encouragement that they both generously provided throughout this thesis project. I would also like to thank Dr. Miranda, in particular, for introducing me to Discrete Event Simulation. I also want to thank the members of my peer review group, Steve Choi and Chen Yuan, for their helpful suggestions and feedback.

Introduction

The maintenance of mission-critical software is a complex and difficult task that is inevitable for most organizations (April & Abran, 2008). In his seminal article on the nature of software, Lehman defined what he called the Laws of Program Evolution, which state that from the time any software system is initially put into use, it will require continuous change or become increasingly less useful, and will lead to continuous deterioration in structure and increasing complexity, unless this progression is intentionally reversed (Lehman, 1980).

This type of software engineering activity is commonly referred to as Software Maintenance and generally defined as the phase of the software lifecycle that begins after its initial release. As end users increasingly rely on the system to support their business activities, a continuous flow of requests become generated that fall into the following categories: corrective updates to fix faults, adaptive updates to keep the system aligned with environmental changes, perfective changes to implement new functionality desired by users, and emergency updates required to keep a system operational (IEEE, 1998).

The activity of software maintenance is distinct from new development, in part, due to the constraints imposed by the live business and technical environments in which it is performed, the unpredictable and highly varied nature and frequency of incoming requests, and the pressure caused by the high cost of delay in the forms of lost revenue, user dissatisfaction, or even reputational damage for an organization. It represents a significant challenge for most organizations and is widely considered to be the longest and most expensive phase in the software lifecycle. Some studies have estimated that it represents 40-80% of all effort spent during the overall software lifecycle (Marques-Neto, Aparecido, & Valente, 2013), and that the cost of this activity for a system can be expected to continue increasing at a rate of up to 10% per year (Grubb & Takang, 2003).

In spite of the fact that software maintenance is critical for organizations and by far the most prominent activity in software engineering, it has historically not received the same degree of academic research as other areas of software engineering (Heeager & Rose, 2015). Many of the methodologies and tools used by software maintenance teams were designed for developing new software and adopted by default, but significant differences in the nature of these two types of work can cause these approaches to lead to inefficient and ineffective results, such as unacceptably long response times for urgent requests, or frequent interruptions of planned work (Rao, 2017). Several process models have been proposed over the years, but none have been universally accepted or would even be feasible for every context (Vasanthapriyan & Arachchi, 2020).

Although many of the activities undertaken during software maintenance are closely related to those required for new software development, such as analysis, design, coding, and testing, the unique environmental and technical issues faced make software maintenance a distinct field within software engineering and require specialized techniques and approaches. As opposed to new software development, there is a continuous stream of work items without the defined time frame, budget, or fixed deliverables typical of project-oriented environments, leading many software maintenance teams to find that the application of service management and queue management techniques are more effective than those from traditional project management.

The evolution of software maintenance processes has largely mirrored the evolution of software development processes in general (Grubb & Takang, 2003), and teams have unsurprisingly turned to Agile methodologies to help manage the flow of work, although commonly adopted Agile processes generally need some adaptation to work in the context of software maintenance (Washizaki, 2024). Studies that have been conducted on this topic have largely found that different Agile methods can successfully be applied to software maintenance to raise team productivity, increase the resulting quality of the software product, and reduce overall costs for organizations, but that teams often face significant issues when attempting to implement them in a standard way for software maintenance work (Ibrahim, Yahaya, Mansor, & Deraman, 2019).

While these studies attempt to validate the effectiveness of particular Agile methodologies and practices, the majority offer only anecdotal evidence from specific cases, their findings vary considerably, and any significant level of consensus still appears to be noticeably absent from the body of literature. The broader question remains: how can a software maintenance team identify the optimal Agile-based methods to employ in their particular environment?

The purpose of this paper is to attempt to address that problem for a team of software engineers at Strivent, a medical device manufacturing company, who perform software maintenance on a custom system built in-house for managing equipment financing. This team employs a process that is loosely based on Kanban concepts but is interested in learning if they can address certain team challenges and also improve team performance by applying additional Agile techniques, in particular by introducing the work-batching, iterative paradigm of Scrum. That inquiry forms the specific research question posed in this study: For a team that is using Kanban, can team performance be measurably improved by the addition of Scrum's time-boxed iterations?

The following section of this paper provides additional context regarding the Strivent software maintenance team and then presents the results of a literature review that was

conducted to better understand the benefits and challenges faced by software maintenance teams when applying Agile methodologies in their process. The literature review also sought to examine the specific ways that Agile methodologies have been adapted for use by software maintenance teams.

The remainder of the paper then describes the research method that was applied in this study, a simulation-based experiment using Discrete Event Simulation (DES). The simulation compares the output from two different process models to show that the performance of a software maintenance team can be measurably improved, as reflected in an increase in completed work, but that this outcome is dependent on sufficiently reducing task duration through effective Scrum practices.

Background and Related Work

The Software Maintenance Team at Strivent

The system that the software maintenance team at Strivent supports was initially put into production almost 10 years ago and has received continuous updates ever since. There is a small amount of system documentation that is available from the initial project build, but much of it is now outdated, having not been updated regularly, and is no longer a reliable resource for current system maintainers.

A steady stream of requests is submitted by the business users who rely daily on the system to conduct their business processes, and the team is responsible for responding to all of them, regardless of the size or type of request. These requests are received through a helpdesk system, translated into user story format by business analysts, and then accumulated into a backlog. The items in the backlog are continuously re-evaluated for priority by representatives from the business in a grouped fashion, meaning that each functional business area is asked to prioritize their requests separately, resulting in a single backlog of items that can be grouped by these functional areas.

These requests can range in level of effort from very small (1-2 hours) to very large (2-3 months). They also vary considerably in the type of work required, including troubleshooting and root cause analysis, bug-fixing, small and large new feature development, data and reporting-related tasks, feasibility analysis, system documentation, and end-user training. The prioritized requests are then continuously picked up by developers as they have additional capacity, with each completed item being deployed in the next available release. Responding to these requests requires not only general software engineering technical skills, but also excellent analytical skills, the ability to analyze and assess complex business processes, and the communication and social skills needed to work with a diverse group of business users (Heeager & Rose, 2015).

In terms of the development process, the team uses concepts and terms that come from both Scrum (User Stories, Backlog) and Kanban (Cards, Work-in-Progress, visualization board) but overall their process most closely resembles a Kanban-based methodology. Work to be done is partially groomed into user stories as they are being identified as next highest in priority and placed into the "Ready" state on the Kanban board. These items are continuously pulled into the work stream by developers when they complete other work, or face impediments and work-in-progress is forced to go into a "Hold" state for any reason. The work-in-progress capacity of the team is generally equal to the number of developers but not closely monitored or enforced as a strict Work-in-Progress limit.

The team experiences two primary challenges created by this continuous flow-style of working. The first is the lack of a more meaningful measure of the team's productivity than the number of work items completed in a given time period. Work is being continuously completed and released, but it is difficult for the team to measure overall productivity, hold itself accountable to any baseline, or measure the impact of changes to the team's processes. Improvements in these areas would benefit both the team itself in increased productivity and job satisfaction, and the business that it supports in increased satisfaction with both the software and the process and experience of working with the maintenance team.

The second major area of challenge for the team is related to the nature of having to respond quickly to a highly unpredictable flow of work, in terms of type of work, level of effort, and particularly the perceived level of urgency to the business. These combined factors contribute to the difficulty the team faces in making accurate task-level estimations, projecting when future work will be started or completed, and needing to frequently switch contexts between different types of work.

Literature Review

With these specific team challenges in mind, a literature review was conducted that sought to understand how software maintenance methodologies have evolved over time, and how the application of specific Agile practices has been found to address these challenges. Several academic research databases were used to perform this review, including Carnegie Mellon University Libraries, IEEE Xplore, the ACM Digital Library, and Google Scholar. The keywords "Agile" and "Software Maintenance" were used to search for sources, and in order to understand the history and evolution of the topic, publish dates were not filtered.

As noted earlier, the evolution of software maintenance processes reflects the evolution of software development processes in general (Grubb & Takang, 2003), and the studies reviewed reflected this progression. The earliest models, introduced in the 1980's and 1990's, were heavyweight, rigid, standards-based approaches that defined a multitude of

roles, activities, and phases. As the software industry began to embrace the principles outlined in the Agile Manifesto in 2001, more lightweight methodologies such as Scrum and XP became the most prevalent (Kajko-Mattsson & Nyfjord, 2009). And most recently, new hybrid Agile models have emerged, sometimes referred to as "Scrumban", that combine the work-batching model of Scrum with the continuous-flow paradigm of Lean and Kanban methodologies (Rao, 2017). It is this last family of methodologies that appears to offer the most compelling process model to address the Strivent team's challenges, adapting and combining elements from different Agile processes to properly manage both planned and unplanned work, while also offering more effective techniques for the measurement and tracking of team productivity. The remainder of this section presents a summary of the reviewed works' findings, organized by the common key strategies they recommend.

Strategy 1: Establish categorization of requests

Many of the studies reviewed highlighted the importance of task categorization to allow teams to triage and prioritize more effectively (Heeager & Rose, 2015; Ladas, 2009; Pino, Ruiz, Garcia, & Piattini, 2012; Polo, Piattini, Ruiz, & Calero, 1999; Rao, 2017). Requests should be categorized upon submission in any manner that would affect the way the item is processed and managed by the team, such as by priority (high/med/low) as well as by urgency (plannable vs unplannable), work type (bug-fix vs enhancement), business area (sales vs marketing), or technology (mobile vs database) (Rao, 2017). When possible, request intake systems should be configured to automatically perform this categorization, and otherwise, should enforce manual categorization.

Once this categorization is well-defined, different policies, workflows, and service level expectations can be created for the different types of work items, establishing varying classes of service (Pino, Ruiz, Garcia, & Piattini, 2012; Polo, Piattini, Ruiz, & Calero, 1999; Rao, 2017). These categories would also allow teams to more effectively track metrics like the types of work performed, team effort and productivity, and service level adherence, all at a more granular level.

Strategy 2: Leverage the benefits of Scrum for "plannable" work

The benefits of applying an iterative Agile process like Scrum are emphasized in many of the reviewed studies (Albarqi & Qureshi, 2018; Almashhadani, Mishra, & Yazici, 2024; Heeager & Rose, 2015; Khan, 2014; Pino, Ruiz, Garcia, & Piattini, 2012; Rao, 2017; Vasanthapriyan & Arachchi, 2020). Sprint-based iterations boost focus, motivation and productivity by establishing unifying goals, allowing for task-level synergies, and reducing the issue of context switching that is common for software maintenance teams. They also create greater team and individual accountability and a heightened sense of urgency.

The use of structured user stories, backlog grooming, and proper assessment of readiness during Sprint Planning, all contribute to more accurate estimations, increased predictability and consistency, and reduced rework. Daily stand-ups provide accountability and transparency, allow the team to quickly address impediments and provide timely feedback to each other, and encourage knowledge sharing.

Team retrospectives enable the team to continuously tune their process, increasing team ownership and accelerating process improvement through repeated cycles of inspection and adaptation. The sprint iterations also allow tracking of team productivity with burndown charts and velocity metrics, helping to establish performance patterns over time.

Strategy 3: Plan ahead for the "unplannable" work

One of the primary challenges of the team at Strivent is the need to respond quickly to urgent, unanticipated requests. Several different strategies have been proposed for managing this type of work without allowing it to derail other work in progress, but they were all based on anticipating the level of these events and budgeting enough team capacity to handle them appropriately. One study recommended having dedicated, concurrently running sprints for both planned and unplanned work (Pino, Ruiz, Garcia, & Piattini, 2012). Other studies recommended establishing a stream or lane within the team's single sprint that is dedicated to such work (Banijamali, et al., 2016; Ladas, 2009; Rao, 2017). Still other studies recommended managing plannable work using Scrum sprints and managing unplannable work separately using the Kanban-based, "pull-system" approach (Albarqi & Qureshi, 2018).

The details of the process would vary considerably across these different options, but the common goal is the isolation of disruptive, unplanned, urgent requests through a well-defined process for handling them. Enabled by the task categorization discussed in the first strategy, different types of work can be handled simultaneously using different processes. This also helps reduce context switching, improve the stability and predictability of plannable work, and allow for quicker response and course adjustments when unplanned work suddenly surges.

Strategy 4: Optimize the continuous flow of work using Kanban

Many of the reviewed studies describe how adopting a Kanban-based process can improve a team's throughput. Kanban and Lean are methodologies that were originally developed in the Japanese manufacturing industry but are now widely used in many other fields including software engineering. The core concepts include the elimination of waste (any non-value-adding activities), the optimization of the continuous flow of work, the visualization of workflow using boards and cards, and the creation of a pull-based (just-in-

time) system through strictly enforced work-in-progress limits (Ahmad, Dennehy, Conboy, & Oivo, 2018; Ahmad, Kuvaja, Oivo, & Markkula, 2016; Albarqi & Qureshi, 2018; Alqudah & Razali, 2018; Banijamali, et al., 2016; Heeager & Rose, 2015; Khan, 2014; Ladas, 2009; Stoica, Ghilic-Micu, Mircea, & Uscatu, 2016; Vasanthapriyan & Arachchi, 2020).

Many software maintenance teams including the one at Strivent use a Kanban-based process to manage their workload. To make the process even more effective for such teams, some studies recommend designing the Kanban board to make planned and unplanned work look different visually, or even to keep them in separate "lanes", in order to increase team awareness, enforce different policies like work-in-progress limits, or enable some categories of work items to take priority over others (Albarqi & Qureshi, 2018; Ladas, 2009; Rao, 2017).

To optimize the flow of work through the team's processes, the studies recommend focusing on flow-based metrics like lead time, cycle time, WIP-age, total work age, throughput, queue size, and flow efficiency (Ahmad, Dennehy, Conboy, & Oivo, 2018; Alqudah & Razali, 2018; Khan, 2014; Rao, 2017).

Strategy 5: Build effective team performance metrics

The previously defined strategies lay the foundation for more effective and meaningful measurements, which is currently a significant challenge for the Strivent team. Defining different processes for tasks based on categorization and then tracking team performance by those categorizations allows for more stable metrics (Khan, 2014; Pino, Ruiz, Garcia, & Piattini, 2012; Rao, 2017). Improved stability and predictability lead to more accurate forecasting and planning based on empirical team data and a higher likelihood of meeting team commitments. When team metrics become more stable and more accurately reflect the performance of the team, a realistic baseline can be established over time, which provides team accountability and can also be used to evaluate the effects of new process changes (Ahmad, Dennehy, Conboy, & Oivo, 2018).

These five key strategies found in the reviewed studies directly address the challenges experienced by the software maintenance team at Strivent and demonstrate that other teams with similar struggles have successfully applied the techniques described to more effectively manage their team's work. They also show that for a team that must balance both planned and unplanned work, the optimal process will likely resemble the "Scrumban" model, incorporating the proper categorization of incoming requests, the definition of different classes of service for specific types of requests, and both work-batching and continuous flow paradigms.

Although many of the studies validate their findings with action-research projects, case studies, or implementation-based experiments, this type of research is inherently limited by the small sample size and the risk that other unknown factors may have influenced the results of the study. Many of them also rely on instruments like surveys or researcher observations that can be prone to introduce bias like the Hawthorne effect and generate conclusions that are difficult to validate. In the case of the Strivent team, it was also not feasible to conduct an implementation-based experiment during the timeframe of this study.

To validate the findings of the literature review and attempt to answer this study's primary research question, a simulation-based experiment was designed with the goal of producing empirical evidence of the productivity gains achieved through the application of the techniques described in the key strategies outlined above. The remainder of this paper describes this experiment that was conducted and presents the results.

Technical Details

Discrete Event Simulation (DES) is a computer-based simulation technique that allows complex systems to be modeled with representations of entities (work items, components or people), processes, resources, queues, and events. It is often used to measure aspects of a system's performance, to improve the operation of the system in some desired way, or to design a system that is to be implemented in the future. It can also be used to better understand how a system functions, or how the system's performance can be expected to change if some aspect of it is modified (Kelton, Zupick, & Ivey, 2024). Software Process Simulation Modeling (SPSM) is a technique that has been used to model software development processes using simulation tools like DES. This type of simulation has been successfully shown to aid in development process evaluation, process optimization, and project risk evaluation and management (Lunesu, Tonelli, Marchesi, & Marchesi, 2021).

Discrete Event Simulation is a good choice for research studies related to software processes because the models can be designed to reflect all the pertinent details of a software process (work items, developers, tasks, queues, rules, assignments, events) and then the tool can quickly simulate the process running over a long period of time. It also allows researchers to quickly test changes to processes without having to implement those changes using a real team of developers, which would likely be highly disruptive and may not even be feasible. It also eliminates many of the issues encountered when researchers attempt to implement a new or changed process in a live environment and then try to analyze the results using surveys or observations, because there are no hidden factors that

might be affecting the simulation results. Finally, the use of a simulation tool allows a researcher to make more generalized claims, limited only by the design of the model and the simulation inputs, rather than by the limited context of a small sample size.

For this study, a Discrete Event Simulation experiment was designed to evaluate how switching from a Kanban-based process to a hybrid, Scrumban model that adds time-boxed iterations would affect a software maintenance team's performance. Using Rockwell Automation's Arena Software, a baseline model was created (Figure 1) that simulates a team that manages work using the continuous-flow approach of Kanban.

Figure 1. Kanban Simulation Model (see Appendix for full size diagram)

A second model was then created that simulates Scrumban (Figure 2), a process in which Scrum-based sprints are used to batch work that is pulled from a product backlog during sprint planning meetings, while retaining the ability to expedite urgent requests by bypassing the product backlog and sprint planning process.

Figure 2. Scrumban Simulation Model (see Appendix for full size diagram)

Both simulations use the same pattern of request creation (Figure 3), which was modeled using one year of historic data from the Strivent maintenance team. Based on this data, 18% of requests represent Urgent requests that must be handled immediately upon submission, with the remaining 82% being Planned requests that are lower in priority. The requests are then assigned a Story Point-based estimation of required effort that ranges from 1 to 13 points, following Fibonacci series-based values (1, 2, 3, 5, 8, 13).

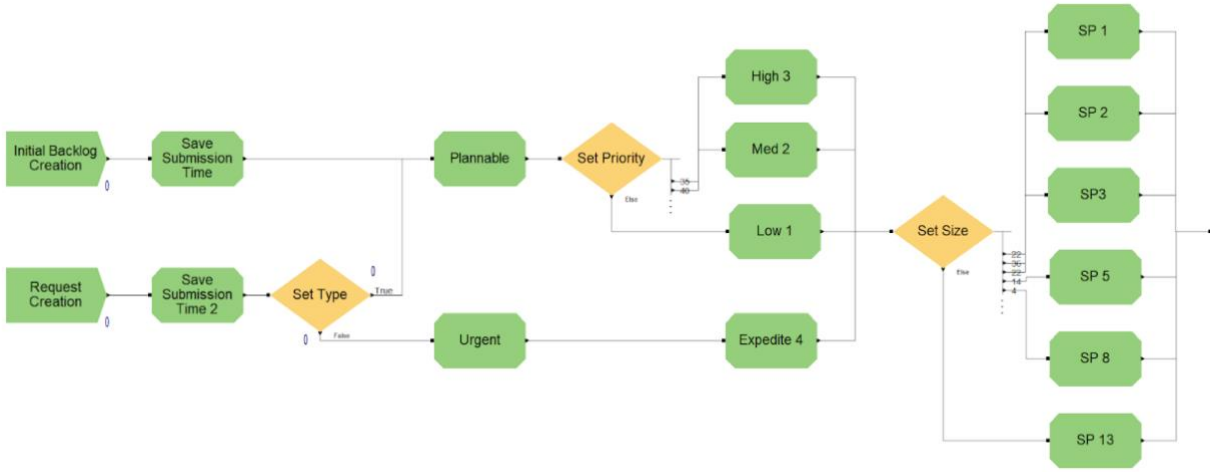


Figure 3. Request Creation (same in both models)

These Story Point values are then translated into the total number of development hours required to complete the request using random triangle-type distributions that define the minimum, most-likely, and maximum values for each Story Point level (Table 1). These distributions were modeled using the team's historic data and reflect varying amounts of effort caused by factors such as incomplete requirements, lack of system comprehension, and any other impediments faced by developers.

Story Points	Request %	Development Length (days)
1	22	TRIA(0.5, 1.6, 8.5)
2	36	TRIA(0.5, 2, 12.5)
3	22	TRIA(1.5, 6, 26.5)
5	14	TRIA(2.5, 11, 22.5)
8	4	TRIA(17.5, 18, 32.5)
13	2	TRIA(14.5, 36, 65.5)

Table 1. Random Distribution for Request Size

The total development hours are then used to calculate the time required for each individual development task (Planning, Design, Coding, Unit Testing, QA, System Testing, Documentation, Demonstration, and Deployment) using an estimate of the percentage of time typically allocated to such tasks (Table 2). The models generate an initial backlog of

fifteen requests at the start of the simulation, and then continuously generate one additional request per day.

Task	% of Total Development Time
Planning	9
Design	13
Coding	36
Unit Testing	7
QA Testing	13
System Testing	9
Documentation	4
Demo/Review	3
Deployment	6
Total	100

Table 2. Task Duration Assignment

Both models simulate a team of three developers, but the total available capacity of these developers differs. In the Kanban model, all developers are available for work assignments for the full eight-hour duration of their workdays. In the Scrumban model, this capacity is reduced by the amount of time they would spend in the typical Scrum process schedule of meetings (Figure 4), which consists of a 3-hour Sprint Planning meeting, a 1.5-hour Sprint Review meeting, and a 1.25-hour Sprint retrospective, each conducted once per 2-week sprint. The schedule also accounts for a 15-minute Stand-Up meeting held at the start of each workday. The total duration of these combined sprint meetings reduces the developer capacity for development work by approximately 10% (8.5 hours out of each 2-week period).

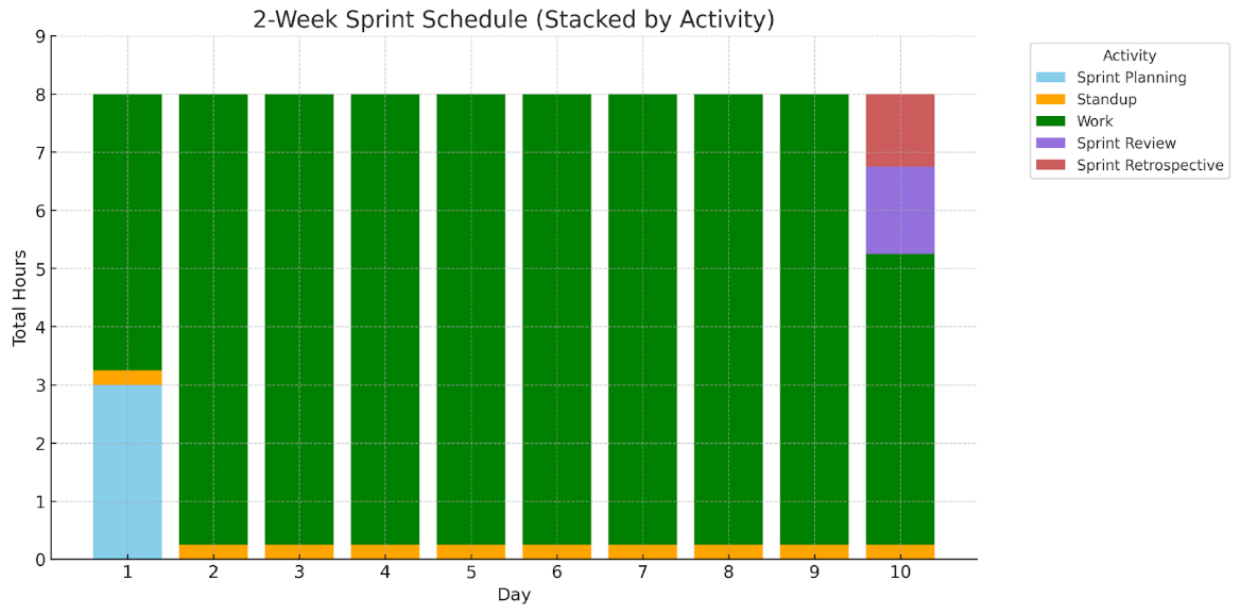


Figure 4. Sprint Schedule

The other key difference between the two models lies in how incoming requests are handled by the team. In the Kanban model, the requests are queued in a single, continuously reprioritized backlog, and are "pulled" by developers in order of priority, regardless of the size of the request. In the Scrumban model, incoming requests are first evaluated with respect to their estimated development effort, and any requests that exceed the team's agreed-upon threshold (40 hours) are first split into smaller units of work that could be handled by multiple developers in parallel (Figure 5).

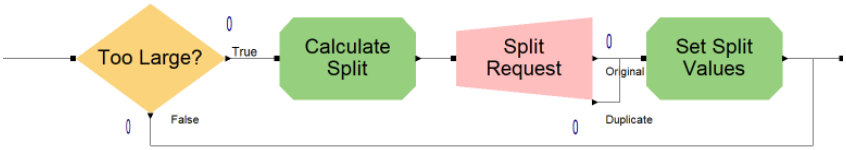


Figure 5. Request Splitting

Urgent requests are still immediately pulled by the next available developer into the work stream, but all other requests are queued in a product backlog for additional grooming and planning. These requests are then pulled into a sprint backlog every two weeks at the Sprint Planning Meeting (Figure 6).

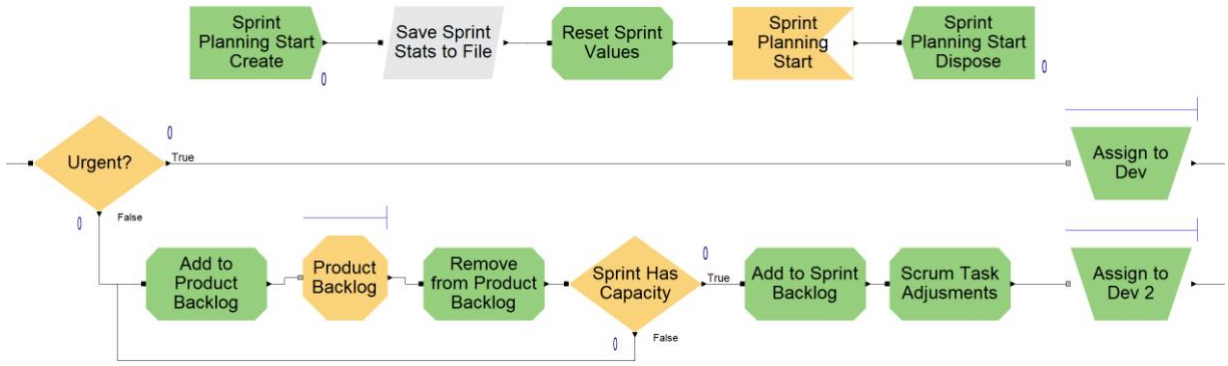


Figure 6. Sprint Planning

For the requests that are handled within a Sprint in the Scrumban model, two additional mechanisms are incorporated in the simulation that are intended to reflect the potential effects of the Scrum process on the team's work.

- Productivity Boost: The daily Stand-Up meetings are designed to increase focus and accountability, improve teamwork, particularly in how impediments are resolved, and create a greater sense of urgency to complete the work that the team has committed to completing in the current sprint. The impact of these influences on the team's performance is represented by a configurable "Productivity Boost" factor that reduces the time required to complete the Design, Code, and Unit Testing tasks for a given request.
- Batching Efficiencies: The second potential effect of the Scrum process on the team's productivity is related to the practice of "batching" work items during a sprint for certain development tasks (Figure 7). Instead of performing these tasks individually for each request, the team completes these activities collectively on a group of work items, which creates efficiencies such as a reduction in setup time for testing and elimination of certain repetitive tasks. The impact of these potential advantages is reflected in the Scrumban model by a configurable "Batching Efficiencies" factor that reduces the time required to perform the QA, System Testing, Documentation, and Deployment tasks.

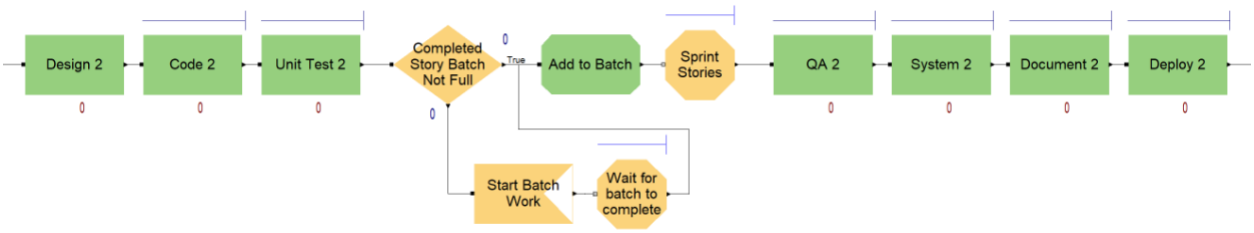


Figure 7. Batching Sprint Items

To evaluate and compare the performance of the team using these two models, the following measurements were selected and incorporated into the simulation:

- Average Cycle Time (time between development start and development end)
- Average Lead Time (time between request submission and development end)
- Average Request Size, in Hours
- Average Request Size, in Story Points
- Total Completed Requests, by Count
- Total Completed Requests, by Story Points

A list of simulation scenarios was designed for the experiment to compare the results of the baseline Kanban process with different versions of the Scrumban process, each with varying levels of "Productivity Boost" and "Batching Efficiencies" reflected in the team's productivity. Both models were run for a simulated 6-month period, consisting only of weekdays and accounting for holidays and paid time off (PTO), which resulted in 98 workdays. Each time the model was run, the simulation was repeated 10 times to allow the defined distributions to introduce a level of randomness in the generated requests, and then the measurements from all 10 replications were averaged to produce the final results, which are presented in the following section.

Results and Analysis

Table 3 lists the scenarios that were run in the experiment along with the resulting measurements from each simulation. The first three columns show the two different simulation categories: Kanban and Scrumban. In the Kanban-based process, the developers are at 100% capacity and there is no limit to request size. This simulation provides the baseline against which the Scrumban simulations are compared. In the Scrumban-based process, the developers are limited to 90% capacity (due to the required sprint meeting schedule) and requests are limited in size to 40 hours, with stories beyond

that threshold being split up into smaller units of work. Table 3 lists the 36 different variations of the Scrumban-based process that were run, each configured to apply a different value for the "Productivity Boost" and "Batching Efficiency" factors. These values ranged from 100% (representing no change in productivity or efficiency) to 75% (representing a 25% reduction in task duration) in 5% increments. The last four columns in the table (Avg Dev Hours, Avg Story Points, Avg Cycle Time, Avg Lead Time) show the results of each simulation.

Simulation Category			Configuration		Measured Output			
Model Name	Developer Capacity	Request Size Limit	Productivity Boost	Batching Efficiency	Avg Dev Hours	Avg Story Points	Avg Cycle Time	Avg Lead Time
Kanban	100%	N/A	N/A	N/A	32.584	2.756	5.300	21.329
Scrumban	90% (Scrum Meetings 10%)	40 hours	1	1	21.539	2.007	8.875	24.574
			1	0.95	21.186	2.014	8.799	24.128
			1	0.9	20.965	2.013	8.768	23.808
			1	0.85	20.574	2.017	8.778	23.733
			1	0.8	20.377	2.029	8.775	23.633
			1	0.75	19.999	2.025	8.734	23.6
			0.95	1	21.037	2.012	8.692	23.846
			0.95	0.95	20.727	2.021	8.877	24.427
			0.95	0.9	20.349	2.017	8.846	23.967
			0.95	0.85	20.069	2.02	8.745	23.38
			0.95	0.8	19.828	2.027	8.423	23.152
			0.95	0.75	19.51	2.024	8.828	23.177
			0.9	1	20.482	2.026	8.543	23.637
			0.9	0.95	20.21	2.038	8.481	23.755
			0.9	0.9	19.888	2.026	8.511	22.85
			0.9	0.85	19.594	2.03	8.676	23.139
			0.9	0.8	19.213	2.032	8.526	23.042
			0.9	0.75	18.946	2.041	8.495	22.842
			0.85	1	19.939	2.027	8.286	23.124
			0.85	0.95	19.706	2.031	8.269	23.044
			0.85	0.9	19.29	2.025	8.355	23.06
			0.85	0.85	19.043	2.04	8.222	22.771
			0.85	0.8	18.726	2.047	8.328	22.79
			0.85	0.75	18.423	2.052	8.293	22.321
			0.8	1	19.477	2.033	8.381	22.868
			0.8	0.95	19.09	2.034	8.109	22.539
			0.8	0.9	18.871	2.054	8.048	22.376
			0.8	0.85	18.467	2.048	8.299	22.379
			0.8	0.8	18.118	2.046	8.182	22.17
			0.8	0.75	17.747	2.049	8.245	22.136
			0.75	0.95	18.549	2.047	8.103	22.179
			0.75	0.9	18.231	2.04	8.13	21.93
			0.75	0.85	17.867	2.053	8.125	21.824
			0.75	0.8	17.545	2.047	8.012	21.666
			0.75	0.75	17.159	2.047	7.9	20.991

Table 3. Average Request Size and Response Times

The Average Request Size, measured in both development hours and story points, decreased significantly due to the splitting of larger requests into smaller work items. The Average Cycle Time of requests in the Kanban process was 5.3 days, which increased to a range of 7.9-8.9 days in the Scrumban process. The Average Lead Time of requests in the Kanban process was 21.3 days, which increased slightly to a range of 21- 24.6 days. Both of these results primarily reflect the batching of some of the development tasks like QA, System Testing, Documentation, and Deployment, which introduced additional queueing time and resulted in longer overall development time.

Simulation Category			Configuration		Measured Output			
Model Name	Developer Capacity	Request Size Limit	Productivity Boost	Batching Efficiency	Completed Count	Completed Story Points	Story Points Diff	Story Points Diff %
Kanban	100%	N/A	N/A	N/A	71	193	N/A	N/A
Scrumban	90% (Scrum Meetings 10%)	40 hours	1	1	93	172	-21.2	-10.97
			1	0.95	95	176	-17.2	-8.90
			1	0.9	96	177	-16.2	-8.39
			1	0.8	97	180	-13.2	-6.83
			1	0.75	99	184	-9.2	-4.76
			0.95	1	96	176	-17.2	-8.90
			0.95	0.95	97	179	-14.2	-7.35
			0.95	0.9	98	181	-12.2	-6.31
			0.95	0.85	99	182	-11.2	-5.80
			0.95	0.8	101	188	-5.2	-2.69
			0.95	0.75	101	187	-6.2	-3.21
			0.9	1	99	183	-10.2	-5.28
			0.9	0.95	100	187	-6.2	-3.21
			0.9	0.9	100	186	-7.2	-3.73
			0.9	0.85	101	188	-5.2	-2.69
			0.9	0.8	103	192	-1.2	-0.62
			0.9	0.75	105	197	3.8	1.97
			0.85	1	101	188	-5.2	-2.69
			0.85	0.95	103	191	-2.2	-1.14
			0.85	0.9	105	195	1.8	0.93
			0.85	0.85	106	198	4.8	2.48
			0.85	0.8	107	201	7.8	4.04
			0.85	0.75	109	206	12.8	6.63
			0.8	1	104	194	0.8	0.41
			0.8	0.95	106	197	3.8	1.97
			0.8	0.9	108	202	8.8	4.55
			0.8	0.85	109	205	11.8	6.11
			0.8	0.8	111	208	14.8	7.66
			0.8	0.75	111	208	14.8	7.66
			0.75	1	108	202	8.8	4.55
			0.75	0.95	109	205	11.8	6.11
			0.75	0.9	112	209	15.8	8.18
			0.75	0.85	113	213	19.8	10.25
			0.75	0.8	114	214	20.8	10.77
			0.75	0.75	117	219	25.8	13.35

Table 4. Total Completed Requests

Table 4 is organized similarly to the previous table of results, showing the two different simulation categories (Kanban and Scrumban) and the baseline Kanban process followed by the 36 Scrumban variations each using different values for the "Productivity Boost" and "Batching Efficiency" factors. The output columns in this table, however, show the Completed Count (total number of requests completed) and Completed Story Points measurements from each simulation.

The Completed Count value increased from 71 in the Kanban model to a range of 93-117 in the Scrumban scenarios, which was also a natural result of splitting the larger stories before the team accepted them in sprint planning. The total number of story points completed, however, provides a more meaningful measure of the amount of work that was completed during the simulation period. In the Scrumban simulations, when larger requests were split into smaller work items, the total combined number of story points of the resulting requests were kept equal to the number of story points of the original request, ensuring that the same overall amount of work was still being represented.

The Scrumban scenarios that are highlighted in red show that fewer overall story points were completed than the baseline Kanban model, reflecting the decrease in developer capacity that resulted from the time developers spent in the scrum meetings. The remaining Scrumban scenarios highlighted in green show that the combined impact of the "Productivity Boost" and "Batching Efficiency" values being applied reduced task duration enough to compensate for the reduction in developer capacity, ultimately resulting in higher overall productivity.

The results also show that at the lowest levels of "Productivity Boost" (100% and 95%), even the highest amounts of "Batching Efficiencies" applied (75%) were insufficient to make up for the loss in developer capacity. In contrast, the highest levels of "Productivity Boost" (80% and 75%) were able to make up for that loss, even without any amount of "Batching Efficiencies" applied (100%). These outcomes illustrated that the "Productivity Boost" factor has a much greater impact than the "Batching Efficiencies", reflecting the fact that the Design, Coding, and Unit Testing tasks collectively account for over 50% of total development effort and therefore provided the greatest area of potential for making an impact on total request duration.

Another detail to note from the results of the Scrumban scenarios is that when "Productivity Boost" and "Batching Efficiency" were both set to 90% (representing a 10% reduction in task duration), the effect was not enough to fully compensate for the 10% decrease in developer capacity. To better understand this aspect of the results, a second series of simulations was conducted (Table 5), in which "Productivity Boost" and "Batching

Efficiency" values were configured to be equal to the Developer Capacity, while that capacity was gradually reduced from 90% down to 65% in 5% increments.

The simulation results in Table 5 show that as Developer Capacity was decreased, while Productivity Boost and Batching Efficiencies were set to an equal level, the resulting loss in productivity continued to drop at an increasing rate.

Category	Configuration			Measured Output			
Model Name	Developer Capacity	Productivity Boost	Batching Efficiency	Completed Count	Completed Story Points	Story Points Diff	Story Points Diff %
Kanban	100%	N/A	N/A	71	193	N/A	N/A
Scrumban	90%	90%	90%	100	186	-7.2	-3.73
	85%	85%	85%	100	186	-7.2	-3.73
	80%	80%	80%	100	185	-8.2	-4.24
	75%	75%	75%	97	179	-14.2	-7.35
	70%	70%	70%	95.7	177	-16.2	-8.39
	65%	65%	65%	93	172.6	-20.6	-10.66

Table 5. Total Completed Requests for Varying Developer Capacity Levels

Conclusion

The results from the simulation provide the answer to the study's research question, as well as some other interesting findings related to the Scrumban process. The first, quite obvious result is that Scrum introduces a significant amount of process overhead in the form of meetings that are critical for successful implementation but can substantially reduce overall capacity. The simulation scenarios incorporated a schedule consisting of the most commonly adopted Scrum meetings, which amounted to 10% of the developer capacity in each two week sprint period, but the Scrum process can also require developers to assist in other activities that are not reflected in the model, such as assisting with backlog grooming and QA testing, preparing for sprint reviews, and participating in code reviews, reducing developer capacity even further.

The values from Table 4 then illustrate the next logical conclusion, which is that the effect of that reduction in developer capacity on team productivity can be mitigated and even surpassed by reducing the time required to complete the requests, which is ultimately what provides the measurable improvement in overall team performance that is needed to satisfy the study's research question. These two opposing forces that are illustrated by the Scrumban-based simulation model, the reduction of developer capacity due to time spent on non-development tasks and ceremonies, and the increase in productivity that can

potentially be achieved through well-managed Scrum practices, lead to the two main findings of this simulation-based study.

The first is that the effectiveness of the daily standup meetings is a critical factor for the successful implementation of a Scrum-based process. The intended purpose of these daily standups is to increase developer focus and accountability, maintain a sense of urgency in meeting the sprint goals, and allow team members to coordinate tasks, share knowledge and work together to quickly resolve any issues that may be impeding progress on sprint tasks. The degree to which the daily standups achieve these intended results was represented in the Scrumban model by the varying "Productivity Boost" factor, and the simulation results showed that overall team productivity declines (due to loss of developer capacity caused by the process overhead meetings) unless this effect can sufficiently reduce task duration during the sprint.

The second finding from the results of the simulation study relates to the duration of the ceremonies and other non-development tasks that are introduced in the Scrum process. Table 5 lists a series of simulations with progressively decreasing Developer Capacity, designed to simulate the effect of increasingly longer Scrum meetings. The resulting difference in Completed Story Points for each simulation illustrates that as developer capacity decreases, the effect of that capacity loss increasingly exceeded the effect of the "Productivity Boost" and "Batching Efficiency" factors that were applied. These results reveal that keeping the Scrum ceremonies and other meetings to the minimum duration possible, while still ensuring that they are effectively fulfilling their intended purpose, is another critical success factor for implementing Scrum.

The creation of the simulation models also produced some unexpected outcomes related to the size of requests. In both the Kanban and Scrumban models, a large percentage of the incoming requests have a total development duration that is greater than 10 days. Table 1 shows that only the smallest category of requests has a maximum value less than 10 days and that the most likely value for the 3 larger categories all exceed 10 days. These larger requests would not fit into a two-week sprint period, so the Scrumban simulations incorporated logic to split them into smaller units of work, which would also permit them to be worked on in parallel. Although not reflected in the model or the results of the simulation, this activity would likely have an additional, significant impact on the Scrumban team's performance in terms of the reduced variability, improved estimation accuracy and reduced scope creep that would likely result naturally from smaller, more manageable units of works.

Another unexpected outcome was related to the measurements used to evaluate team performance. When a team is following a Kanban-based process, team performance is

typically assessed using flow-based metrics, which include Cycle Time, Lead Time, and Throughput. When a team implements a Scrum-based process, however, those metrics become less meaningful. Work on each request becomes more spread out throughout the sprint, or in some cases even before the sprint begins during backlog grooming sessions, which can make the "start time" of the request difficult to define. Items are then queued for some tasks that do not get completed until the end of the sprint period, which leads to longer Cycle Time and Lead Time values. Completed work items may also not be considered truly "done" until the sprint review or the deployment is complete, which makes the "end time" of the request less meaningful. Teams that follow a Scrum-based process, on the other hand, typically rely on sprint-based measurements to track team performance, such as average velocity and sprint burndown charts. A Scrumban team that is implementing both Scrum-based sprints for planned work and a Kanban-based pull system for unplanned work would benefit from using both types of measurements to track team performance for these different categories of requests, but for the purposes of this simulation study, neither of these types were effective in comparing the results of the two models.

Finally, it should be acknowledged that the validity of the results of this simulation study is limited to the specific design of the process models and the simulation inputs that were selected for the models. The details of the Kanban model and the model inputs were designed to represent the Strivent team's current process as closely as possible, while the Scrumban model was designed by envisioning how that process would be changed by adding the Scrum-based iterations while preserving the ability to quickly respond to urgent, unplanned work items.

The models did not capture every detail of the development process, however, and some of these aspects could have affected the measured outputs of the simulation. When larger requests were split into smaller units of work, the total net development time of those resulting units would likely be lower or higher than that of the original request, but that difference was not accounted for. The simulation also did not incorporate different levels of developer skills or specialized skills, so the simulation assumed that any developer on the team was able to handle any type of request. The random distribution of development time for each request was, however, designed to reflect this variability to some degree. The models also did not include any logic for rework or test failures, both of which would have occurred while using either process, but the Scrum process would likely reduce these rates through better scope management and the rejection of work items that do not meet the team's agreed-upon "definition of ready".

Summary and Future Work

This study examined the activity commonly referred to as Software Maintenance, defining this type of work in terms of what makes it distinct from other types of development and the unique challenges faced, and then looking to understand the implications of using Agile-based methodologies to manage such work. The study also described the particular challenges faced by the software maintenance team at Strivent, which became the basis for the research question posed in this study: how would the adoption of the hybrid Agile methodology known as Scrumban affect this team's overall productivity?

A Discrete Event Simulation-based experiment was then designed with the goal of producing empirical evidence of the productivity gains achievable through the application of these techniques. A baseline model was created that represents the Strivent team's current Kanban-based process, which was then compared to a second model that was based on a Scrumban process that incorporated the strategies identified in the literature study.

The simulation illustrated two opposing forces in the Scrumban process, the reduction of developer capacity due to time spent on non-development tasks and ceremonies, and the increase in productivity that can potentially be achieved through well-managed Scrum practices. The results showed that the Scrumban process could produce a measurable increase in the team's productivity, as reflected in the total completed number of story points, but that this outcome was dependent on reducing task duration through effective daily standups and controlling the duration of scrum meetings.

To extend the work that was begun in this study, an implementation experiment by the Strivent team would be necessary to validate these simulation-based findings. Additional research could also examine how specific practices result in the reduction of task duration, which was represented by the "Productivity Boost" factor in the Scrumban simulations.

Bibliography

Ahmad, M., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software, 137*, 96-113.

Ahmad, M., Kuvaja, P., Oivo, M., & Markkula, J. (2016). Transition of software maintenance teams from Scrum to Kanban. *2016 49th Hawaii International Conference on System Sciences* (pp. 5427-5436). IEEE.

Albarqi, A., & Qureshi, R. (2018). The proposed L-Scrumban methodology to improve the efficiency of agile software development. *International Journal of Information Engineering and Electronic Business, 11*(3), 23-35.

Almashhadani, M., Mishra, A., & Yazici, A. (2024). Software maintenance practices using agile methods towards cloud environment: A systematic mapping. *Journal of Software: Evolution and Process, 36*(11), e2698. doi:10.1002/smr.2698

Alqudah, M., & Razali, R. (2018). An empirical study of Scrumban formation based on the selection of scrum and Kanban practices. *International Journal on Advanced Science Engineering Information Technology, 8*(6), 2315-2322.

Anderson, D., Concas, G., Lunesu, M., & Marchesi, M. (2011). Studying lean-kanban approach using software process simulation. *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011* (pp. 12-26). Madrid, Spain: Springer Berlin Heidelberg. doi:10.1007/978-3-642-20677-1_2

Anderson, D., Concas, G., Lunesu, M., Marchesi, M., & Zhang, H. (2012). A comparative study of Scrum and Kanban approaches on a real case study using simulation. *Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 201* (pp. 123-137). Malmö, Sweden: Springer Berlin Heidelberg. doi:10.1007/978-3-642-30350-0_9

April, A., & Abran, A. (2008). *Software maintenance management: evaluation and continuous improvement.* Hoboken, NJ: John Wiley & Sons, Inc.

Banijamali, A., Dawadi, R., Ahmad, M., Simila, J., Oivo, M., & Liukkunen, K. (2016). Empirical Investigation of Scrumban in Global Software Development. *Model-Driven Engineering and Software Development: 4th International Conference, MODELSWARD 2016* (pp. 229-248). Rome, Italy: Springer International Publishing.

Barghouti, N., & Rosenblum, D. (1994). A case study in modeling a human-intensive, corporate software process. *Proceedings of the Third International Conference on*

the Software Process. *Applying the Software Process* (pp. 99-110). Reston, VA: IEEE. doi:10.1109/SPCON.1994.344418

Grubb, P., & Takang, A. A. (2003). *Software Maintenance: Concepts and Practice*. Singapore: World Scientific Publishing Co. doi:10.1142/5318

Heeager, L. T., & Rose, J. (2015). Optimising agile development practices for the maintenance operation: nine heuristics. *Empirical Software Engineering : An International Journal, 20*, 1762-1784. doi:10.1007/s10664-014-9335-7

Ibrahim, K., Yahaya, J., Mansor, Z., & Deraman, A. (2019). The emergence of agile maintenance: A preliminary study. *Proceedings of the 2019 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 146-151). IEEE. doi:10.1109/ICEEI47359.2019.8988815

IEEE. (1998). *IEEE Standard for Software Maintenance, IEEE Std 1219-1998*. New York, NY: Institute of Electrical and Electronics Engineers.

Kajko-Mattsson, M., & Nyfjord, J. (2009). A model of agile evolution and maintenance process. *2009 42nd Hawaii International Conference on System Sciences* (pp. 1-10). IEEE.

Kelton, W., Zupick, N., & Ivey, N. (2024). *Simulation with Arena (7th ed.)*. McGraw-Hill Education.

Khan, Z. A. (2014). *Scrumban - Adaptive Agile Development Process - Using scrumban to improve software development processKhan*.

Ladas, C. (2009). *Scrumban-essays on kanban systems for lean software development*. Modus Cooperandi Pr.

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE* (pp. 1060 - 1076). IEEE. doi:10.1109/PROC.1980.11805

Lunesu, M., Tonelli, R., Marchesi, L., & Marchesi, M. (2021). Assessing the risk of software development in agile methodologies using simulation. *IEEE Access, 9*, 134240 - 134258. doi:10.1109/ACCESS.2021.3115941

Marques-Neto, H. T., Aparecido, G. J., & Valente, M. T. (2013). *A quantitative approach for evaluating software maintenance services*. doi:10.1145/2480362.2480565

Pino, F. J., Ruiz, F., Garcia, F., & Piattini, M. (2012). A software maintenance methodology for small organizations: Agile_MANTEMA. *Journal of Software: Evolution and Process,* (8), 851-876. doi:10.1002/smr.541

Polo, M., Piattini, M., Ruiz, F., & Calero, C. (1999). MANTEMA: A software maintenance methodology based on the ISO/IEC 12207 standard. *Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99).'Best Software Practices for the Internet Age'* , (pp. 76-81).

Rao, N. (2017). *Scrumban Software Maintenance: 5 Steps to Stop Starting and Start Finishing.* Createspace Independent Publishing. Retrieved from https://www.amazon.com/Scrumban-Software-Maintenance-Mr-Nagesh/dp/1546519319

Rus, I., Neu, H., & Münch, J. (2014). A systematic methodology for developing discrete event simulation models of software development processes. *Proceedings of the 4th International Workshop on Software Process Simulation and Modeling (ProSim 2003)*. doi:10.48550/arXiv.1403.3559

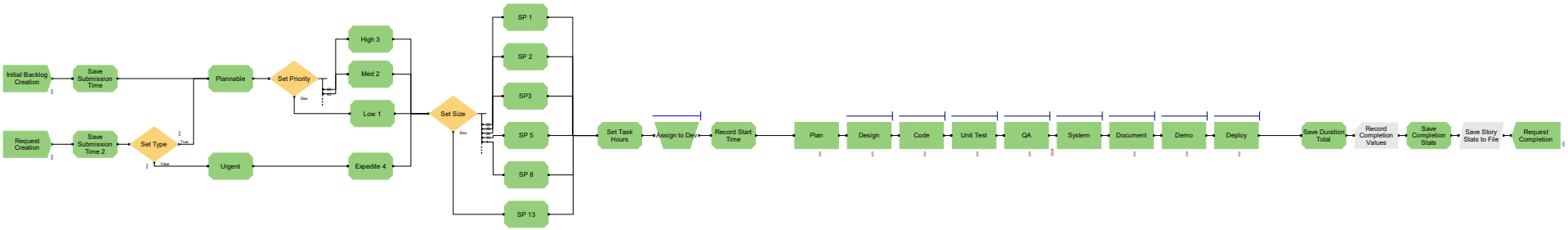
Singh, R. (1996). International Standard ISO/IEC 12207 software life cycle processes. *Software Process Improvement, 2*(1), 35-50.

Stoica, M., Ghilic-Micu, B., Mircea, M., & Uscatu, C. (2016). Analyzing agile development-from waterfall style to scrumban. *Informatica Economica, 20*(4), 5-14.

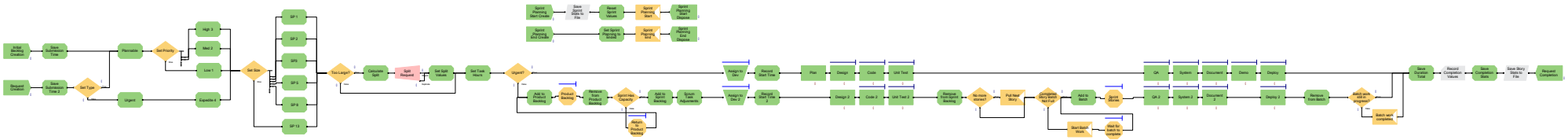
Vasanthapriyan, S., & Arachchi, K. (2020). Effectiveness of Scrum and Kanban on Agile-Based Software Maintenance Projects. *Software Engineering for Agile Application Development*, 282-300. doi:10.4018/978-1-7998-2531-9.ch013

Washizaki, H. (Ed.). (2024). *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0.* IEEE Computer Society. Retrieved from https://www.swebok.org

Appendix



Kanban Simulation Model



Scrumban Simulation Model

Enhancing Shared Mental Models in Distributed Software Engineering Teams

Chen Yuan cyuan2@andrew.cmu.edu

Abstract

Distributed software engineering has posed new challenges to team coordination, shared understanding, and performance, which are often inadequately addressed by communication tools alone. This thesis examines the application of the cognitive science framework of Shared Mental Models (SMMs) to evaluate and improve collaboration in distributed engineering teams. Drawing on Johnson's five-factor SMM model, which encompasses team knowledge, team skills, team attitude, team dynamics, and team environment, a structured survey was designed and deployed within a U.S.-based distributed software team. Leveraging quantitative measurements, such as mean agreement scores and within-group consensus (rWG), as well as qualitative analysis of open-ended responses, the thesis identifies key areas of misalignment and improvements, particularly in team environments and coordination practices. Based on these findings, the thesis proposes lightweight and targeted interventions, such as utilizing asynchronous planning templates, team role mapping, and structured check-ins. Through the integration of theory, measurement, and interventions, this thesis aims to provide a practical and replicable approach to enhancing team cognition and performance in a distributed environment, thereby laying the foundation for scalable and cognitively sensitive engineering practices.

Introduction

The COVID-19 pandemic has significantly changed how software engineering teams operate, accelerating the adoption of remote and distributed work as a long-term norm. In 2021, more than 44% of employees in the information sector were working remotely, almost four times the rate seen in 2019 (Pabilonia & Redmond, 2024). Despite distributed software development having clear benefits in terms of a broader talent pool and greater flexibility, it also comes with typical coordination issues that can lead to lower productivity compared to teams working in co-located environments (Liska, 2022; Herbsleb & Mockus, 2023).

Many productivity tools and platforms have been developed and adopted by large companies to facilitate distributed team communication and documentation, such as

Slack, Zoom, and Confluence. While tools like Slack and Zoom have made team interactions more seamless, they fall short in fostering accurate alignment. These platforms enhance communication, yet they do not guarantee that team members interpret goals, responsibilities, or workflows in the same way, a fundamental barrier in distributed collaboration. The misalignment may cause delays, repeated efforts, and failures to meet goals, eventually leading to a decline in the effectiveness of the teamwork.

This thesis argues that team cognition, specifically Shared Mental Models (SMMs), is a critical model to understand and improve the performance of software teams, and can also help distributed teams resolve misalignments in their unique environment. SMM represents the shared knowledge structure that enables team members to anticipate each other's needs, coordinate effectively, and adapt to changing situations (Cannon-Bowers et al., 1993; Cooke et al., 2000). When these models are weak or misaligned, coordination will become inefficient, regardless of the tools or technologies used.

To gain a better understanding of how we can leverage SMM to help build distributed teams, I took three steps to apply SMM:

1. **Measurement**: I created and implemented a structured survey based on the five SMM factors–team knowledge, team skills, team attitudes, team dynamics, and team environment–to evaluate cognitive alignment in a real-world distributed software development team.
2. **Analysis**: Quantitative and qualitative studies were conducted after the survey responses were collected. Analysis helps identify the strongest and weakest SMM factors, allowing for the design of targeted interventions accordingly.
3. **Intervention**. Based on the areas of improvement from the analysis, I then proposed experiments with lightweight and targeted interventions designed to reinforce the weak points in shared cognition.

The work is based on an empirical study conducted within a distributed software engineering team at a large technology company. The team of engineers spans multiple U.S. time zones, with different in-office and remote work arrangements. I created a survey with questions designed according to various factors of SMM to assess the current level of shared mental models within the team. The findings informed a series of targeted, lightweight interventions designed to enhance alignment and cooperation. By focusing on one team in depth, this thesis

demonstrates how systematic assessment and customized intervention can yield immediate, scalable strategies for improving the effectiveness of distributed teams.

Focusing on cognitive origins of teamwork rather than technological fixes, the work here attempts to provide software engineer teams and technology leaders practical recommendations for how to increase coordination, trust, and productivity in the distributed context.

Key Terms & Concepts

Term	Description
Team Cognition	The process by which team members collectively encode, store, and retrieve knowledge to coordinate action (Salas et al., 2005). It includes communication, shared understanding, and coordinated decision-making.
Shared Mental Models (SMMs)	Internalized and overlapping cognitive structures among team members regarding tasks, roles, tools, and workflows. SMMs facilitate implicit coordination and have been demonstrated to enhance performance (Cannon-Bowers et al., 1993; Cooke et al., 2000).
SMM Factors	The five categories of shared mental models used in this thesis are **team knowledge, team skills, team attitudes, team dynamics,** and **team environment** (Johnson, 2007).
Distributed Software Development (DSD)	A software engineering practice in which team members are located across different geographic locations or time zones and collaborate through digital tools (Tait et al., 2013).
rWG (Within-Group Agreement)	A statistical measure used to assess consensus or agreement among raters, typically when multiple individuals rate items on a Likert scale. It's helpful in team or organizational research to evaluate whether individual responses can be meaningfully aggregated to the group level.

Background and Related Work

Distributed Teams and Their Challenges

Distributed software development (DSD) refers to the engineering collaboration across geographically dispersed locations, often spanning multiple time zones (and even countries) and work environments (Tail et al., 2013). Although this structure offers advantages of access to a broader talent base, improved work-life balance, and flexibility, each still comes with coordination challenges. Compared to co-located teams, distributed teams often face increased communication overhead, fewer informal interactions, and splintered tool ecosystems (Herbsleb & Mokus, 2003; Storey et al., 2017).

Several factors contribute to these challenges:

- **Reduced informal communication**: Spontaneous interactions, such as coffee chats and overheard conversations, are largely absent from remote team members, weakening situational awareness and alignment (Olson, 2000).
- **Lack of observability**: Team members cannot easily infer others' workload, blockers, or emotional state without explicit updates (Espinosa et al., 2007).
- **Asynchronous coordination**: Time zone misalignments lead to delayed feedback loops and difficulty maintaining shared context (Sarker et al., 2011). Even though technologies have been widely adopted to improve the remote communication experience, prompt response can never be guaranteed.
- **Tool fragmentation**: Knowledge is spread across various platforms, including Slack, Confluence, and Jira, making it challenging to form a unified understanding (Storey et al., 2017).
- **Reduced psychological safety**: Team members may feel less recognized or supported when working remotely, leading to hesitance in asking questions or raising concerns, which is critical to refine shared understanding from their perspectives. (Edmondson, 1999).

These factors directly harm the cognitive alignment necessary for effective collaboration and team performance within a distributed environment.

Team Cognition and Its Role In Distributed Collaboration

As Sales et al. (2004) pointed out, teamwork is achieved when members interact independently and work together toward shared and valued goals. It is more than work accomplished by a group of individuals, and can be viewed as the result of collective cognitive, behavioral, and attitudinal activity. Sales then defined Team Cognition as an emergent property of teams that results from the interplay of individual cognition and team-level processes such as communication and coordination. It refers to how knowledge is structured, shared, and used within a team to enable coordinated action. For example, in a distributed software team, an iOS developer might begin implementing a new feature immediately after the backend engineer updates the API without requiring a formal handoff. This type of seamless coordination is possible because both developers share a mental model of the project timeline, task dependencies, and each other's responsibilities, allowing them to act proactively and in sync.

Numerous studies have established a strong correlation between team cognition and team productivity and performance, highlighting that well-developed team cognition processes within teams significantly enhance collaborative outcomes (Cooke et al., 2000; Salas et al., 2005). Team cognition supports team performance by ensuring that members share a common understanding of goals and tasks, roles and responsibilities, tools and technologies, and the capabilities and interdependencies of team members (Mohammed, Ferzandi, & Hamilton, 2010). It helps improve team coordination and anticipation, reducing the need for constant clarification or micromanagement. Team cognition ensures more efficient decision-making and problem-solving, while also decreasing miscommunication and errors, leading to higher adaptability and resilience.

In distributed teams, the contextual cues are sparse, and misunderstandings are widespread. Team cognition becomes especially vital. Research has shown that teams with high levels of cognitive congruence, characterized by a shared awareness of goals, roles, workflows, and expertise, perform significantly better on complex tasks (Mohammed, Ferzandi, & Hamilton, 2010; Cooke et al., 2000).

There are several influential Team Cognition frameworks, such as Shared Mental Models, Transactive Memory Systems, Team Mental Models, and Team Situation Awareness. I choose Shared Mental Models because SMMs provide concrete dimensions that can be measured and targeted directly, and are more lightweight and scalable.

Shared Mental Models (SMMs)

According to Cooke et al. (2000), Shared Mental Models are internalized knowledge structures about key elements of the team's environment, task, and members. When these structures are aligned across the team, individuals can coordinate implicitly, leading to higher performance. The knowledge associated with SMM can be classified into two types: task knowledge and team knowledge. Task knowledge refers to the specific knowledge related to a particular task and is typically highly domain-specific, whereas team knowledge is related to general team processes and characteristics.

SMMs can then be further categorized based on two different criteria:

Four Dimensions of SMM (Cannon-Bowers et al., 1993)

Dimensions reflect the types of knowledge that can be shared:

- **Task Model** - the understanding of goals, strategies, and task procedures.
- **Team Model** - the awareness of team member skills, preferences, and roles.
- **Equipment Model** - the shared understanding of tools, systems, or interfaces.
- **Team Interaction Mode**l - norms and expectations for communication and coordination.

Five Factors of SMM (Johnson, 2007)

Johnson (2007) conducted a meta-analysis of various research related to team knowledge and summarized the following key factors of team knowledge:

- **Team knowledge** - knowledge about members and tasks that they need to perform as a team, including teammates' knowledge and task knowledge.
- **Team skills** - abilities associated with successful job performance, including communication skills, interpersonal skills, leadership skills, and skills to deal with conflicts and team cohesion.
- **Team attitudes** - shared beliefs and shared values that influence team members' choices or decisions under particular circumstances.
- **Team dynamics** - a combination of dynamic processes of team coordination and team cohesion.
- **Team environment** - external conditions affecting the foundation of the shared mental model, including technology organization, synchrony, and geographical dispersion.

The four dimensions describe the content of shared knowledge, while the factors describe the function and context of shared knowledge (why it matters in practice). In this thesis, the five-factor model is employed to structure measurement and interventions, as it aligns more closely with software engineering workflows and observable team challenges.

Measuring Shared Mental Models

Before we attempt to improve the shared mental models for a distributed team, we need to determine a method for understanding its current situation, specifically by measuring the shared mental models. Mental models are inherently internal and invisible; therefore, measuring SMMs involves eliciting, comparing, and analyzing how team members conceptualize key aspects of their work. Measuring SMMs will help teams detect gaps in shared understanding before they lead to coordination breakdowns. With the five-factor model, specific dimensions where misalignment occurs can be identified to design better interventions and evaluate their effectiveness over time. In this way, measurement makes intangible cognition tangible and actionable.

Researchers have developed both qualitative and quantitative methods to capture and measure shared cognition, and these approaches can be grouped into three categories:

Elicitation-Based Methods

These methods aim to externalize team members' mental models directly. Some standard techniques are:

- **Concept Mapping**: A visual tool used to represent knowledge and relationships between concepts. For example, individuals draw diagrams showing relationships among tasks, tools, or responsibilities, and then maps are used to compare for structural similarity.
- **Card Sorting/Similarity Ratings:** Members sort items (roles, tasks, tools) based on perceived relationship or rate similarity between pairs. These are aggregated to form cognitive distance or similarity metrics.
- **Structured Interviews:** Interviews with open-ended questions to reveal team members' assumptions, terminology, and understanding, which are coded for convergence/divergence.

Analysis-Based Methods

These techniques infer sharedness by analyzing data generated by the team:

- **ACSMM** (Analysis-Constructed Shared Mental Models): Developed by Johnson (2007), this method reconstructs shared mental models from team artifacts (e.g., surveys, documents) by computing convergence patterns across responses.
- **Semantic Analysis**: Text responses are compared using natural language processing (e.g., Jaccard similarity, cosine similarity) to quantify overlap in conceptual representatives.

Survey-Based Methods

Surveys offer a structured and scalable way to quantify team cognition across key dimensions:

- Items are constructed around specific SMM content areas (e.g., understanding of goals, team responsibilities, communication practices).
- Responses are typically collected via Likert scales, allowing quantitative analysis of agreement and overlap.
- Survey responses can be compared across team members using interrater reliability measures (e.g., rWG, ICC).

Literature Review

Numerous studies have been conducted to understand how shared mental models can impact team performance, particularly in complex and distributed task environments.

One of the earliest and most comprehensive frameworks for measuring team cognition originates from Cooke et al. (2000). These authors argue that team knowledge is not merely the aggregation of individual cognition; instead, it emerges from common task representations, communication, and coordinated action. The authors suggested ways of measuring team cognition in both qualitative (e.g., concept maps, performance observations) and quantitative (e.g., similarity measures, survey data analysis) methods. One of the key takeaways from their work is that good teams are not about sharing all the knowledge; they are about sharing the proper knowledge in appropriate ways that facilitate anticipation and implicit coordination. Their work also highlighted the importance of converging across mental models. This insight is foundational to this thesis's use of interrater

agreement and semantic similarity metrics to assess sharedness in distributed teams.

Johnson (2007) conducted a meta-analysis based on decades of research on team knowledge and cognition, proposing five core factors that consistently impact team performance: team knowledge, team skills, team attitudes, team dynamics, and team environment. Unlike Cannon-Bowers et al. (1993)'s model, which focused solely on task and teammate, Johnson's framework expanded the lens of SMMs to include broader behavioral and contextual elements. His study was particularly relevant to distributed teams, where environmental factors and interpersonal dynamics are highlighted. The five-factor model proposed by Johnson directly inspired the structure of the survey instrument in this thesis, providing a more actionable approach to analyzing and intervening on specific cognitive weaknesses in distributed teams.

Herbsleb and Mockus (2003) offer one of the first large-scale empirical studies comparing co-located and distributed software teams. Comparing change request data and survey data, they find that modifications made by distributed teams incurred about 2.5 times the time of changes made by co-located teams. Crucially, the performance disparity was not due to task complexity, but rather to added communication overhead and reduced informal awareness. Their result reinforces the fact that communication infrastructure and common knowledge, rather than geography in isolation, account for software engineering team performance disparities. Their work laid much of the groundwork for subsequent studies that explored team cognition in the distributed context.

Ali (2021) surveyed software engineers at scale during the COVID-19 pandemic and found that 83% of developers experienced workplace burnout, with the root causes being inefficient processes, ill-defined goals, and excessive workload. All of these are connected to misaligned mental models of expectations, ownership, and task management. The work demonstrates how distributed contexts exacerbate the problems and highlights psychological safety and common ground as essential buffer factors for productivity and well-being.

Limitations

While this thesis provides a structured approach to assessing and improving Shared Mental Models (SMMs) in distributed software engineering teams, several limitations should be noted:

1. Small Sample Size
The study involved only 10 participants from a single distributed engineering team. While sufficient for exploratory analysis and preliminary rWG calculations, the limited sample size constrains statistical power and generalizability. In particular, confirmatory factor analysis (CFA) could not be reliably conducted, and findings may not extend to teams of different sizes, geographies, or cultures.

2. Single-Team Context
The empirical investigation was conducted within one team embedded in a large technology organization. The team's structure, culture, and tooling environment may have shaped both their SMMs and the effectiveness of the interventions. Broader validation across multiple teams and companies is necessary to strengthen external validity.

3. Self-Reported Data
The survey relied on self-reported perceptions, which are inherently subject to bias, such as social desirability, acquiescence, or mood effects. Although triangulated with open-ended responses, the absence of behavioral or performance-based data (e.g., delivery speed, defect rates) limits objective correlation with actual outcomes.

4. Short-Term Evaluation
This study captured a snapshot of cognitive alignment at a single time point. While interventions were designed and proposed based on observed gaps, longitudinal evaluation of post-intervention outcomes was beyond the scope of this project. As such, the lasting impact of SMM interventions on team performance remains untested.

5. Simplification of Cognitive Constructs
The use of a five-factor SMM model provides a practical measurement structure, but it may oversimplify the dynamic and emergent nature of team cognition. Real-world team cognition involves overlapping constructs such as Transactive Memory Systems and Situational Awareness, which were not measured in this study.

Technical Detail

Shared Mental Models Measurement

Shared mental models can be measured using elicitation-based methods, analysis-based methods, and survey-based methods. Although elicitation methods can provide more details, they are usually time-intensive and challenging to scale in fast-moving software development teams. On the other hand, analysis-based approaches are more effective in distributed settings where traceable artifacts and asynchronous documentation are prevalent. Survey-based methods are also suitable for distributed teams due to

- High scalability, so that the survey is easy to deploy across team members, regardless of locations and time zones;
- Easy to design by following predefined SMM factors.
- Better comparability by applying statistical analysis across time and between teams.

Therefore, in this thesis, a survey was conducted based on the five-factor model created by Johnson (2007), and different analyses will be performed based on question types.

Survey Design and Implementation

To systematically evaluate the condition of shared cognition among members of a distributed software engineering team, I created a survey based on the five-factor model of Shared Mental Models (Johnson, 2007). These five factors, **team knowledge, team skills, team attitudes, team dynamics, and team environment**, are essential facets of cognitive alignment that affect the performance of the teams, especially in the context of distribution.

Both quantitative and qualitative data were obtained using the survey. Perceived convergence and alignment among the team members were assessed using Likert-scale items. At the same time, open questions enabled participants to express their views in their own words, allowing for the exploration of thematic divergence. These two supplementary methods enabled a multi-layered assessment of the standard mental model of the team. Both the detailed survey and collected responses can be found in the appendix section.

Survey Context and Participant Environment

The survey target was a software engineer team consisting of developers across different locations in the US. Below is a breakdown of the team composition:

Location	Size (# of Engs)	Office Attendance
New York City	4	In-office Tue–Thu
Remote	3	Fully remote
Mountain View / SF	3	In-office optional, no manager onsite

Table 1: Survey Participants Distribution

The majority of the team (20+ engineers) is located around NYC and are required to work from the office 3 times a week. To avoid bias, only five engineers from NYC were picked to participate in this survey. When people work from California and are close to a local office, enforcing this requirement can be challenging because there is no engineer manager in the same area. As for people living in other places, they are fully remote and do not have access to nearby offices. The team consists of engineers from different platforms, and in their daily lives, they need to collaborate closely with the product team (product managers, designers, and data scientists), as well as external engineering teams.

The sample size of this experiment was relatively small, so the result could be inaccurate due to limited statistical power. To resolve this problem, some strategies will be applied to improve the credibility of this research. For example, qualitative interviews can be conducted in complement with the survey, and each participant can serve as their own baseline for the pre/post intervention comparison. Through the use of methods like pairwise comparison, structured coding, and descriptive similarity metrics, the work will be able to draw valid inferences about how shared understanding differs within a distributed group and changes after targeted intervention.

Post Survey Analysis and Intervention Design

Survey Response Analysis

To determine the level of shared cognition across the distributed engineering team, I examined the survey responses with quantitative measures of agreement and

qualitative measures of divergence. The goal of the analysis was to uncover the weakest SMM factors where members of the group had the least common knowledge, and to use the results as a basis for targeted intervention.

1. Quantitative Analysis of Likert-Scale Responses

For each of the Likert-scale questions (rated from 1: Strongly Disagree to 5: Strongly Agree), two key statistical measures were calculated:

- **Mean and Standard Deviation**: The average score reflects the general perception across the team, while the standard deviation captures variability. Items with low means (<3.5) and/or high standard deviation (>1.0) suggest potential misalignment or uncertainty in that area.

- **Within-Group Agreement (rWG):** I used the rWG statistics to measure the degree of consensus among members for each item. rWG can be calculated using the following formula:

$$rWG = 1 - \frac{S_x^2}{S_{EU}^2}$$

  - S_x^2 : Observed variance of ratings within the group
  - S_{EU}^2 : Expected variance under uniform distribution (i.e., if there were no agreement)

  To interpret the rWG analysis result, we agree on the following:

  - rWG > 0.70 → Strong agreement (suitable for aggregating to team level)
  - rWG 0.51–0.70 → Moderate agreement (acceptable with caution)
  - rWG < 0.50 → Low agreement (respondents disagree)

2. Qualitative Analysis of Open-Ended Responses

To complement the numerical findings, open-ended responses were thematically analyzed using a manual coding approach:

- Responses were reviewed and grouped into conceptual themes based on keywords and intent.
- Divergence was assessed by comparing the number and consistency of themes across respondents.

- Where appropriate, semantic similarity (e.g., Jaccard index) was applied to quantify conceptual overlap in key responses such as team goals, coordination processes, and values.

This qualitative layer was critical in identifying latent misalignments not evident from Likert scores alone. For instance, when asked “What is the team’s primary goal this quarter?”, participants referenced a wide range of initiatives (e.g., backend refactor vs. customer-facing feature delivery), suggesting a lack of shared task focus under the Team Knowledge factor.

3. Cross-Factor Comparison

After analyzing individual questions, results were aggregated by SMM factor to determine which areas exhibited the lowest sharedness. This was based on:

- The number of low-scoring Likert items (mean < 3.5)
- rWG values indicating poor within-team agreement
- Thematic inconsistency in open-ended responses

This composite view enabled the identification of specific cognitive gaps in the team and directed the development of uniquely targeted interventions. For example, if the result indicates low agreement on the Team Dynamics factor, coordination ritual, and clear handoff interventions received emphasis.

SMM Interventions

Based on the quantitative and qualitative survey data and diagnostic observations, I proposed a factor-aligned intervention based on the five SMM factors. Unlike adopting a standard solution, the intervention relies on the weaknesses inferred from the individual's actual factors. For instance, Team Knowledge shortcomings can be addressed by using role-mapping workshops or knowledge-transfer sessions. Similarly, Team Environment problems can lead to adjustments in collaboration tool settings or modifications to asynchronous communication norms. With the factor-aligned approach, all intervention comes directly to supporting the development of a better, coherent, and aligned shared mental model in the distributed team.

Results and Analysis

A total of 10 responses were collected for this survey, with 3 participants working remotely from home, three working from the California office, and four working from the NYC office. Three people did not participate in this survey.

Quantitative Analysis

Fifteen questions were asked on a Likert scale, and the mean, standard deviation (SD), and rWG (Within-Group Agreement) were calculated to evaluate both the **strength** and **coherence** of shared mental models. Please refer to the above section on how to interpret these numbers. The calculated results are as follows:

Question	Q1	Q3	Team Knowledge
Mean	3.9	3.6	3.75
SD	0.5676	0.6992	0.5401
rWG	0.855	0.78	0.8175

Question	Q6	Q7	Q8	Q9	Team Skills
Mean	4.0	4.4	4.1	4.5	4.25
SD	0.6667	0.5164	0.8756	0.7071	0.5137
rWG	0.8	0.88	0.655	0.775	0.7775

Question	Q11	Q12	Q14	Team Attitudes
Mean	3.7	4.2	3.9	3.93
SD	1.0593	0.6325	0.8756	0.7337
rWG	0.495	0.82	0.655	0.66

Question	Q15	Q17	Q18	Team Dynamics
Mean	3.6	3.8	3.9	3.77
SD	0.8433	0.9189	0.7379	0.7036
rWG	0.68	0.62	0.755	0.685

Question	Q21	Q22	Q23	Team Environment

Mean	4.5	3.9	3.1	3.83
SD	0.527	0.9944	1.1972	0.572
rWG	0.875	0.555	0.355	0.595

Table 2: Mean, Standard Deviation, and rWG for Each Item and Factor

We can quickly identify some standout questions:

- Q23 (Mean = 3.1, SD = 1.20, rWG = 0.355): Mixed views on **time zone/location issues** with lowest agreement but high variance.
- Q11 (SD = 1.06, rWG = 0.495): Varying views on **shared team values.**
- Q22, Q17, and Q8 are borderline and may benefit from clarification or intervention.

Also, by comparing the data across factors, we can see that:

SMM Factor	Question IDs	Mean	SD	rWG	Interpretation
Team Knowledge	Q1, Q3	3.75	0.63	0.82	Moderate score, strong agreement
Team Skills	Q6-Q9	4.25	0.68	0.78	Strongest area, cohesive
Team Attitudes	Q11, Q12, Q14	3.93	0.86	0.66	Moderate-high score, borderline agreement
Team Dynamics	Q15, Q17, Q18	3.77	0.83	0.69	Moderate score, mixed coordination
Team Environment	Q21-Q23	3.83	0.90	0.60	Weakest alignment, inconsistent views

Table 3: Factor Level Analysis

- **Team Skills** (Mean = 4.25) is the strongest area, indicating the team feels confident in communication and conflict resolution.
- **Team Attitudes** and **Team Dynamics** exhibit higher variability (SD > 0.7 and rWG close to 0.7), indicating mixed perceptions of shared values and coordination.
- **Team Knowledge** is slightly below average (Mean = 3.75), which may indicate opportunities for better role/task clarity.
- **Team Knowledge** (0.82) and **Team Skills** (0.78) show strong agreement, and the results are safe to aggregate to team-level scores.
- **Team Environment** (0.60) has the lowest agreement; team members are not aligned in how they perceive tooling or geographic/time zone impact.

The results from both quantitative analyses indicate that **Team Environment** is the weakest shared mental model for this team. Moreover, this finding aligns with distributed software development characteristics, where the environment is the key factor leading to performance decrease.

Qualitative Analysis

To complement the quantitative survey results, qualitative responses from open-ended questions were coded and thematically analyzed. Each response was reviewed for recurring patterns and then grouped under the five Shared Mental Model (SMM) factors. This provided more profound insight into the cognitive alignment and operational challenges experienced by the distributed team.

Under **Team Knowledge**, two themes emerged. First, respondents consistently identified **informal networks of expertise**, often listing specific colleagues they turn to for help on iOS, Android, or API work. This indicates that while knowledge is present in the team, it relies heavily on tacit awareness rather than formal documentation or role clarity. Second, the theme of **fragmented task ownership** appeared in how team members described their roles, often in narrow, platform-specific terms with limited reference to cross-functional collaboration. These findings align with the relatively moderate mean score and variability seen in the quantitative responses for Team Knowledge.

The qualitative responses reflected a mixed perception of **Team Skills** in the distributed engineering team. Most respondents praised the communication habits of the team, including regular stand-ups and asynchronous notifications. However, there were consistent complaints about **inconsistent communication styles** across time zones and **unclear leadership** when working collaboratively, particularly in cross-functional projects. A number of the responses indicated a lack of formal conflict management processes, leading to unresolved tensions or passive disengagement from decision-making. These results suggest that there is a baseline level of communication, but the team may benefit from the development of common communication norms, improved clarity of roles, and explicit conflict management practices to improve team cohesion and performance.

The factor **Team Attitudes** showed strong alignment in responses to the team’s overarching goals. Most participants independently cited "product-market fit" and "retention improvement" as the team’s key focus, reflecting a shared understanding of mission and values. This theme of **shared goal alignment** reinforces the high mean score observed in the corresponding Likert-scale responses.

In the **Team Dynamics** category, responses highlighted blending cohesion-building processes, including stand-ups, retrospectives, Slack chat, and informal check-ins. The **mixed cohesion processes** theme reflects that, although there are multiple ways of interacting, they are not as frequent or effective. It corresponds to the moderate standard deviation of the **Team Dynamics** factor, in that some members find strong bonds, yet others do not.

Finally, under **Team Environment**, a strong theme of **environmental disconnection** emerged. Several participants explicitly cited challenges related to remote work, time zone differences, and the absence of organic, in-person interactions. These issues affect communication flow and delay feedback cycles. This aligns with both the low mean and low rWG observed in the Team Environment factor, indicating not only dissatisfaction but also disagreement on the extent of the problem.

In sum, the open-ended responses provide firm contextual grounding for the quantitative findings. They reinforce the areas of cognitive strength, such as goal alignment, while surfacing qualitative evidence for potential weaknesses in task awareness, coordination, and environmental design in distributed team settings.

Intervention Design

The survey analysis reveals that while the team demonstrates a strong alignment in **Team Skills** (mean = 4.25, rWG = 0.78) and **Team Knowledge** (rWG = 0.82), there are notable weaknesses in the other three areas:

1. Team Environment

- Quantitative finding: Mean = 3.61; rWG = 0.60, the lowest agreement and satisfaction across all factors
- Qualitative insight: Open-ended responses cited **time zone** delays, remote work friction, and lack of spontaneous interaction.
- Problem: Asynchronous communication challenges and the absence of informal cues disrupt shared understanding.

Intervention:
Adopt structured async planning templates for key projects and decisions, e.g., “decision memos,” status boards with context notes, and video briefings. These promote shared context even across time zones and have been shown to reduce misalignment in distributed teams (Storey et al., 2021).

To offset the challenge of time zone mismatching as well as diminished synchronous work in the distributed team, overlapping working agreements can be instituted. The intervention instigates a predictable window, most commonly one to two hours on special days, during which all the members of the team should be logged in. It aims to achieve high-bandwidth collaboration to clear blockers, make decisions, or conduct fast alignment discussions. It has been proven that even overlapping time in short expanses has significantly improved coordination as well as shared situational awareness in globally distributed teams (Ford et al., 2021; Herbsleb & Mockus, 2003). Establishing intentional moments of synchronicity, the team can reduce the latency, provide shared context, and enhance the overall alignment of the team environment.

2. Team Attitudes

- Quantitative finding: Mean = 3.93; rWG = 0.66, indicating moderately high scores but only borderline consensus.
- Qualitative insight: While respondents showed alignment around high-level goals (e.g., product-market fit), open-ended responses revealed variability in perceived values and decision-making norms under pressure.
- Problem: Inconsistent internalization of shared values and varying expectations around accountability and team norms may weaken trust and collective ownership.

Intervention:
Introduce a “Team Values Reflection” exercise in a retrospective or offsite. In this lightweight exercise, team members are asked to discuss what they believe are team values and where they see inconsistencies (e.g., collaboration vs. speed, ownership vs. deference). The output is collated into a simple “Values-in-Action” guide to be used during planning and deciding. It fosters reflection, alignment, and reinforcing of common beliefs, particularly in high-stress environments. Team psychological safety and alignment research (Edmondson, 1999; Salas et al., 2005) indicates that if team members collaboratively develop their set of guiding principles, they gain improved trust and consistency in decisions.

The team can also adopt blameless postmortems as a standard practice after incidents or failures. Instead of assigning fault, these sessions emphasize learning, system improvement, and shared accountability, reinforcing a psychologically safe culture that encourages open dialogue.

3. Team Dynamics

- Quantitative finding: Moderate agreement (rWG = 0.69), Mean = 3.77
- Qualitative insight: Some respondents referenced rituals like retros and happy hours; others mentioned coordination challenges and inconsistent check-ins.
- Problem: Coordination mechanisms are inconsistent, reducing predictability and team rhythm.

Intervention:
Lead a lightweight coordination process, e.g., weekly coordination check-ins or pre-retro alignment check-ins. Employ structured prompts (e.g., “What are we co-dependent on?”, “Who is blocked?”) to support reflection on interdependence. These processes promote shared task awareness and planning, at the center of teamwork (Entin & Serfaty, 1999).

Conclusion

Together with the rising popularity of distributed work among software engineers, the software engineering community has growing problems in achieving appropriate coordination, alignment, and team performance. Although communication has improved with the increased use of virtual communication media like Slack and Zoom, they lack a standard foundation, a key to successful collaboration. Team cognition, in this work, through the framework of the construct of the Shared Mental Models (SMM), presents a systematic, science-based framework for the diagnosis of issues in the performance of distanced teams.

In a case study of a dispersed engineering team, we illustrate how SMM can be assessed using a standardized survey tool that corresponds to five primary factors: team knowledge, team skills, team attitudes, team dynamics, and team environment. The quantitative data, consisting of descriptive statistics and rWG values, was complemented by thematic coding of open-ended survey responses to provide a comprehensive view of cognitive alignment within the team.

These findings highlighted areas of strength in shared communication skills and common goals, yet also revealed issues of inconsistent coordination, ambiguity in the roles of members, and environmental friction due to the disparity in time zones. From these, special-purpose, lightweight intervention strategies were suggested to strengthen the looser SMM factors, such as by using role-mapping sessions, asynchronous planning software, and formalized coordination rituals.

Through the integration of empirical data with cognitive theory, this thesis presents a replicable process for the assessment and enhancement of team cognition in geographically dispersed environments. It enables software teams and technical leaders to transcend mere tooling fixes and address the underlying cognitive processes that foster productive collaboration.

Looking ahead, this work lays a strong foundation for future iterations of the SMM survey, improved validation via CFA with larger sample sizes, and longitudinal studies of how shared cognition evolves in distributed teams over time. Ultimately, the goal is to enable distributed teams not just to function, but to thrive through shared understanding, aligned expectations, and stronger cognitive coherence.

Summary and Future Work

This thesis aimed to investigate how the Shared Mental Models (SMM) of team cognition can be used to understand, measure, and improve the performance of distributed software development teams. By designing and implementing a survey based on five SMM factors, this research captured the positive aspects as well as the misalignments within a real-world distributed team. Quantities such as mean values, standard deviations, and within-group agreement (rWG) were supplemented by qualitative discussion of open-ended comments. These findings revealed persistent alignment in communication and the team's objectives. However, they also highlighted discrepancies in task knowledge, in the consistency of coordination, and in the impacts of environmental constraints such as time zones.

Based on these findings, the thesis proposed a set of targeted, lightweight interventions aimed at improving cognitive alignment and team effectiveness. This framework demonstrates how SMM theory can be translated into practical diagnostics and design strategies, enabling distributed teams to build stronger shared understanding and adaptability in complex, asynchronous work environments.

Several important directions remain for future research:

- Scaling the Survey
  The current study involved a small team of 10 participants. Future iterations should collect responses from larger and more diverse teams to improve statistical validity and enable stronger conclusions from methods like

Confirmatory Factor Analysis (CFA). CFA was attempted when analyzing the responses using Jamovi, but the results yielded low credibility due to the small sample size. A broader dataset would also support Exploratory Factor Analysis (EFA) to refine the factor structure of the survey.

- Longitudinal Evaluation:
  The study only obtained a snapshot of the cognitive alignment of the team. Future work should conduct follow-up surveys or interviews at regular intervals to monitor the changes in SMM following interventions and assess their long-term impact on coordination, trust, and performance.

- Cross-Team Comparison:
  Comparing teams across different organizations, project types, or degrees of distributedness (e.g., fully remote, hybrid) can reveal contextual SMM development patterns and effectiveness, yielding more generalizable design recommendations.

- Tool Integration:
  Future work could explore how SMM metrics might be embedded into team tools (e.g., dashboards, retrospectives, onboarding flows) to support continuous feedback loops and real-time awareness of team cognition.

- Intervention Validation:
  While this thesis proposes targeted interventions, empirical testing of their outcomes, through performance metrics, sentiment tracking, or follow-up SMM assessments, is needed to confirm their effectiveness and optimize design.

This thesis contributes a first step toward bringing the science of team cognition into distributed software practice. By linking theory, measurement, and intervention, it offers a pathway for distributed teams to strengthen alignment, not just through better communication, but through better understanding.

Acknowledgements

I would like to thank my thesis advisors sincerely, Prof. Eduardo Miranda and Prof. Bradley Schmerl, for their thoughtful guidance, critical insights, and consistent feedback throughout this research.

I would also like to thank Joe Yoon and Steve Choi for their invaluable support and feedback throughout this course. Their insights and feedback continuously helped me refine my thesis and sparked ideas that inspired new directions in my work.

I am especially grateful to my teammates at LinkedIn for participating in the survey and providing open, honest feedback. Their collaboration and candid responses made this research both meaningful and grounded in real-world challenges.

Finally, I am deeply appreciative of my family. Their patience, motivation, and emotional support sustained me through the long hours of writing, reflection, and revision.

Bibliography

Bernardo, A., Smeak, R., & Williams, A. (2020). ACCOMMODATING THE OLDER WORKER: Implications for Safety Professionals. Professional Safety, 65(10), 26–31.https://aeasseincludes.assp.org/professionalsafety/pastissues/065/10/F1Bernardoetal_1020.pdf

Cannon-Bowers, J. A., Salas, E., & Converse, S. (1993). Shared mental models in expert team decision making. In N. J. Castellan Jr. (Ed.), Individual and group decision making: Current issues (pp. 221–246). Lawrence Erlbaum Associates.

Cooke, N. J., Salas, E., Kiekel, P. A., & Bell, B. (2004). Advances in measuring team cognition. In E. Salas & S. M. Fiore (Eds.), Team cognition: Understanding the factors that drive process and performance (pp. 83–106). American Psychological Association. https://doi.org/10.1037/10690-005

Johnson, T. E., Lee, Y., Lee, M., O'Connor, D. L., Khalil, M. K., & Huang, X. (2007). Measuring Sharedness of Team-Related Knowledge: Design and Validation of a Shared Mental Model Instrument. Human Resource Development International, 10(4), 437–454. https://doi.org/10.1080/13678860701723802

Entin, E. E., & Serfaty, D. (1999). Adaptive team coordination. Human Factors, 41(2), 312–325. https://doi.org/10.1518/001872099779591196

Fiore, S. M., & Salas, E. (Eds.). (2004). Team cognition: Understanding the factors that drive process and performance. American Psychological Association. https://doi.org/10.1037/10690-000

Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering, 29(6), 481–494. https://doi.org/10.1109/TSE.2003.1205177

Mohammed, S., Klimoski, R., & Rentsch, J. R. (2000). The measurement of team mental models: We have no shared schema. Organizational Research Methods, 3(2), 123–165. https://doi.org/10.1177/109442810032001

Pabilonia, S. W., & Redmond, P. (2024). Telework, productivity, and worker well-being in the information sector. Monthly Labor Review, U.S. Bureau of Labor Statistics. https://doi.org/10.21916/mlr.2024.5

Salas, E., Sims, D. E., & Burke, C. S. (2005). Is there a “Big Five” in teamwork? Small Group Research, 36(5), 555–599. https://doi.org/10.1177/1046496405277134

Storey, M.-A., Treude, C., van Deursen, A., & Cheng, L.-T. (2021). The (r)Evolution of software engineering collaboration. Communications of the ACM, 64(3), 62–71. https://doi.org/10.1145/2593882.2593887

Johnson, T. E., & O'Connor, D. (2008). Measuring team shared understanding using the analysis-constructed shared mental model methodology. Performance Improvement Quarterly. https://doi.org/10.1002/piq.20034

Edmondson, A. (1999). Psychological Safety and Learning Behavior in Work Teams, Administrative Science Quarterly, 44(2), 350–383. https://doi.org/10.2307/2666999

Appendix

Survey Detail

Category	Question
Factor 1 - Team Knowledge	Understanding of team members’ responsibilities and the team’s collective task knowledge.
	Q1: (Rating*) How confident are you that your understanding of each team member’s responsibilities matches their actual responsibilities?
	Q2: (Open-ended) Who do you rely on for help with key areas (e.g., Android, iOS, API, DS, Design, Product)?

	Q3: (Rating) I clearly understand each team member's role and technical strengths.
	Q4: (Open-ended) What are your primary responsibilities within the team?
	Q5: (Open-ended) In your own words, what is the team's main goal for this quarter?
Factor 2 - Team Skills	Abilities critical to team success, including communication, leadership, and conflict resolution.
	Q6: (Rating) Team members communicate clearly and effectively with one another.
	Q7: (Rating) Conflicts or disagreements within the team are addressed constructively.
	Q8: (Rating) Our team has leaders (formal or informal) who help maintain clarity and direction.
	Q9: (Rating) I feel comfortable giving feedback to teammates regardless of their seniority.
	Q10: (Open-ended) What practices help your team build cohesion (e.g., rituals, retros, syncs)?
Factor 3 - Team Attitudes	Shared values, beliefs, and decision-making tendencies under pressure or uncertainty.
	Q11: (Rating) Our team shares a common belief in delivering high-quality software.
	Q12: (Rating) When problems arise, the team prioritizes collaboration over blame.
	Q13: (Open-ended) What values or principles do you think guide how your team works?
	Q14: (Rating) My team shares a sense of ownership and accountability for our outcomes.
Factor 4 - Team Dynamics	Processes such as coordination, decision-making, and cohesion are involved in day-to-day work.
	Q15: (Rating) Our team has precise and efficient workflows for tasks such as code review and deployment.

	Q16: (Open-ended) Describe how your team coordinates work when tasks are interdependent.
	Q17: (Rating) Even across time zones, my team is aligned and well-coordinated.
	Q18: (Rating) Our team regularly adapts or improves our collaboration practices (e.g., adjusting meetings, communication styles).
	Q19: (Open-ended) What is one example of strong coordination you have seen on this team recently?
Factor 5 - Team Environment	Contextual factors include time zones, tools, communication platforms, and technical infrastructure.
	Q20: (Open-ended) What tools or platforms do you use most frequently to collaborate with your team (e.g., Slack, Jira, GitHub)?
	Q21: (Rating) I have the tools and access needed to stay in sync with my teammates.
	Q22: (Rating) The team's communication tools are practical for both synchronous and asynchronous work.
	Q23: (Rating) Time zone differences create challenges for my daily collaboration.
	Q24: (Open-ended) What is the most significant environmental challenge you face in working with this distributed team?

Table 3: Survey Questionnaire

* Unless otherwise noted, rating questions will use the following Likert scale:
☐ Strongly Disagree — ☐ Disagree — ☐ Neutral — ☐ Agree — ☐ Strongly Agree

Survey Responses

You can find the survey results in the CSV file:
https://drive.google.com/file/d/16UA_DtrNG0uVbP3r4BX8AbCuJpMf9ofR/view?usp=drive_link

The original survey cannot be shared externally since it was created using a company account.

Small Language Models on Edge Devices: What Works and What Doesn't

Submitted partial fulfillment of the requirements for the degree
of

Master of Software Engineering

Lihan Zhan

Bachelor of Science, Computer Science, The City College of New York - Grove School of Engineering

Carnegie Mellon University
Pittsburgh, PA

December 2025

© Lihan Zhan

2025 All Rights Reserved

Acknowledgments

A special thanks to my colleagues Grace A. Lewis and Brandon Born for inspiring this topic through our discussions on deploying small language models on edge devices. I am also grateful to my classmate Carl Gruhn for his guidance and for helping me stay focused when my writing went off topic. Finally, I would like to thank my thesis advisor, Bradley Schmerl, for his timely feedback and for connecting me with peers whose insights expanded my view of how SLMs can be applied across different fields.

Abstract

In this thesis, we studied how small language models (SLMs) run on a small, low-powered edge device. We focused on the NVIDIA Jetson Xavier NX and measured practical limits across short, medium, and long inputs. We ran stress tests until failure and logged out-of-memory errors, timeouts, and runtime errors. We also reported end-to-end latency, peak memory during inference, and model size on disk. Finally, with the collected data, we summarized trade-offs among speed, memory, size and provided simple guidelines for model size, input length, and runtime choices on Jetson-class devices.

1. Introduction

1.1 Motivation and Importance

Large language models (LLMs), such as ChatGPT 5, typically have tens or hundreds of billions of parameters and run in the cloud. Small language models (SLMs) use the same transformer architecture but have approximately eighty million to five billion parameters, which makes them candidates for embedded deployment [11].

Many organizations would like to use SLMs on edge devices – such as the NVIDIA Jetson – to support activities, for example, automated robot instructions or voice-based assistants for soldiers in an internet-denial environment. These devices typically run with strict limits on memory, power, and local storage; often battery-powered and mounted on small robots, vehicles, or handheld units. They often share hardware resources -- both GPU and RAM -- with other time-critical tasks such as perception, control, or mapping. In many cases, they operate with weak or no network connectivity, or are not allowed to send sensitive data off the device, so developers cannot rely on a cloud LLM API as a backup. If a language model uses too much memory, draws too much power, or responds too slowly, it can shorten mission time or simply lead to mission failure. In practice, every token of computation must fit within a tight resource budget while still leaving some resources for the rest of the system.

Overall, deploying SLMs on edge devices is difficult for two main reasons: (1) latency and memory use grow quickly as prompt length increases, and (2) deploying similar models in different runtimes can fail or change behavior. Techniques such as quantization can make models smaller and faster but can also reduce quality or become unstable for some tasks and input lengths [12]. The bottom line is that engineering teams at different organizations do not have a simple way to understand how prompt length, model size, and deployment choices interact with latency, power, memory, and storage on edge devices; which are essential for making design and deployment decisions.

1.2 Problem Statement & Emerging Claim

Currently, there are a limited number of reproducible methods that help engineering teams assess whether a given SLM will meet edge-deployment requirements, e.g. end-to-end latency and resource usage on a specific device and software setup. They lack the tool that can stress test a SLM and be confident that the edge device is capable of supplying the needed resources. To address this, we aimed to create a simple stress evaluation on a fixed Jetson Xavier NX image. By “stress evaluation,” we mean a controlled stress test where we steadily increase the prompt length while measuring how the model’s latency, power usage, peak memory, and size on disk behave. We logged latency, power, energy, peak memory, and model size on disk of different models while incrementally increasing the input prompt length. By doing so, we intend to show the practical limits and common failure modes of different models.

1.3 Research Questions

To turn this problem into something concrete and answerable, we focused on how prompt length and model size interact on a fixed Jetson Xavier NX device. The goal is to help engineers who are choosing an SLM for an edge deployment in understanding three key aspects: (1) how far they can safely push prompt length before the system becomes impractical, (2) what trade-offs they face between latency, power, memory, and storage, and (3) when and why deployments fail. These three concerns lead to the following research questions.

- RQ1. Identify practical limits on edge devices for execution of small language models for latency, memory, and storage versus short, medium, and long inputs.
- RQ2. Identify the trade-offs between different metrics of the model. For example, to keep the latency under a certain threshold, do we need to sacrifice energy consumption?
- RQ3. Identify when and why deployment fails on edge devices due to for example: out-of-memory errors, export or runtime errors, timeouts. Based on findings, develop guidance for teams to select the model size, quantization level, input length, and runtime to balance latency, memory, and storage.

2 Background and Related Work

2.1 Terminology

We used a significant number of language-model related technical terms throughout this thesis, it would be beneficial to go over them for better context. Small language models (SLMs) are models trained with roughly 100M–5B parameters [11]. Prompt length is measured by the number of input tokens supplied to the model in a single prompt [8]. Generated tokens are the tokens produced by the model during the decoding process.

In this thesis, end-to-end latency indicates the time from the initial request to the final token being produced [4, 9]. Throughput is the number of generated tokens divided by the end-to-end latency. Power refers to the power draw per request in milliwatts; we log the power trace per token during a run [5]. Peak memory is the maximum device RAM observed during a run [5, 7].

We also mentioned a few deployment choices. Quantization is a technique that reduces the precision of weights to shrink model size and increase throughput [1, 2]. The runtime path is the inference runtime we use; in this work, the runtime path is PyTorch.

At the end, we defined two terms related to failure. A failure condition occurs when the system hits an out-of-memory (OOM) error, a timeout, a runtime error, or a system crash. The practical limit is the largest input size that an SLM can handle on the device without encountering a failure condition.

2.2 Standards & Best Practices

- Latency Reporting - Latency percentile and end-to-end metrics following Google SRE [9] recommendations for usability and reliability.
- Benchmark Consistency - Methodologies inspired by LMPerf Inference guide [14] (fixed hardware image, consistent config, etc)
- Quantization Practices - Procedures for low-bit inference with the ONNX Runtime guides. [1]

2.3 Contextual Issues

Edge devices face a number of contextual constraints. They typically have much lower memory capacity than conventional systems, and memory usage can grow superlinear with input length, creating practical limits for running large models. In addition, power pressure during inference can trigger power throttling, which may increase latency and reduce system stability. Finally, this study is limited in scope to a single Jetson model and a single JetPack image, meaning that all observed limits apply only to this specific configuration; broader hardware validation will be required in future work to generalize these findings.

3 Related Work

3.1 State of the Art

Research on edge inference has focused on how to balance latency, memory, and accuracy when deploying language models on constrained hardware. Chen et al. [7] proposed a benchmarking framework for LLM inference on edge devices that measures how latency and memory interact with model accuracy and confirm these two metrics are the dominant factors of deploying models on edge devices.

Surveys conducted by Wang [11] show that the current industry considers small language model size ranges from 100 M - 5 B parameters. It also describes how quantization, pruning, and runtime selection can affect models under energy and storage limits. Edge-First Language Model Inference [13] compared single-device and distributed edge setups, showing that SLMs can reach near-cloud performance for short inputs when latency and memory are thoroughly tested and constraints are defined. These results suggest that small language models are promising candidates for embedded deployment, and that engineering teams in different industries are attempting to deploy SLMs on different resource-constrained devices.

Consider, for example, a battery-powered mobile robot used for warehouse inspection that runs on a Jetson-class device. The engineering team might want to deploy an SLM on the robot so operators can ask natural-language questions about recent sensor readings or alerts, but the

device also has to stay within strict power, latency, and memory limits. Industrial standards like MLPerf Inference (Edge) [4] have established transparent benchmarking methods for computer-vision and speech workloads, but there is a lack of comparable, reproducible evaluations for SLMs on edge devices, forcing teams in this situation to rely on trial and error when deciding which model, prompt length, and configuration are safe to deploy. Meanwhile, HELM [10] provides a holistic framework for comparing models using different metrics, showing the need to consider accuracy, efficiency, and reliability together when evaluating deployment trade-offs, but it does not answer device-specific questions such as how far prompt length can be pushed on a small device like Jetson before latency, power, or memory make the system unusable.

3.2 Gaps and Open Questions

Despite the growing amount of research, important gaps remain:

- Lack of Reproducible Evaluation Methods:
- Limited Power Analysis
- Insufficient Characterization of Failure Conditions
- Limited Studies on Small Models Overall

Current benchmarks often lack detailed device configuration/setup guidance making it hard for engineering teams to quickly test the feasibility of deploying certain models on an edge device. Most of the prior work measures latency and memory without power consumption. But this is one of the most important metrics in a resource-constrained environment. Studies usually focus on the performance metrics and do not categorize failure causes properly and link them to different metrics. (OOM due to increased input size, peak of latency due to power limit, etc) The industry tends to focus on cloud-scale models, leaving a gap for lightweight language models that can run on embedded GPUs.

3.3 Position of My Work

Based on the gaps discussed in the previous section, this thesis addresses those gaps by developing a stress-based evaluation tailored to Jetson Xavier NX that can be expanded to all other Jetson models. Our work incorporates one of the most important metrics: power consumption, while trying to develop a practical rule of thumb guide for developers to deploy SLMs on Jetson. This work conducts experimental measures to generate a trade-off table between different important metrics of SLMs. It aims to translate academic/industrial benchmarks to an industrial-ready deployment guide.

4. Methods

4.1 Research Design & Justification

The goal of this thesis is to determine how far a small language model (SLM) can be pushed on an embedded GPU before it becomes impractical. We define “impractical” in this case as the point at which the device runs out of memory, prompting becomes too slow or causes power throttling, or a failure occurs due to other reasons. In order to evaluate the effect of input size, we hold the device and software image constant and increase the prompt length incrementally. For each length, we send the same type of request and generate a fixed number of output tokens, then calculate the per-token latency, power, and energy for that run. We also record the peak memory usage, and any export/runtime failures of the model. This approach allows us to link prompt length to the core deployment constraints on edge hardware. It helps us answer RQ1 by showing practical limits as inputs grow, RQ2 by quantifying the trade-offs among latency, energy, memory, and storage (for example, how keeping latency below a target can increase energy cost or tighten memory), and RQ3 by identifying when and why runs fail.

4.2 Setting and Systems

All experiments were conducted on an NVIDIA Jetson Xavier NX (8 GB) with a clean JetPack 5.1.2 image. This board is a common embedded GPU platform in robotics and other edge-AI applications, and its 8 GB RAM and limited power budget are representative of the resource-constrained devices where SLMs are likely to be deployed. We use PyTorch with CUDA from that image to evaluate a set of small language models. Model wise, we use the same six SLMs described in Section 5. To make the results reproducible, we save the image build information, Python and CUDA versions with `pip freeze`, and commit hashes for the runner scripts so that other engineers can flash the same image and repeat or expand the study to different devices and models.

4.3 Data Sources

We run the Jetson built-in tegrastats command in the background with timestamps aligned to the device clock. `Tegrastats` is available on all Jetson boards by default and reports GPU and system power, clocks, and memory at a fixed sampling interval, which makes it a practical and reproducible choice for our study. From these samples, we compute the average power per request, then we can compute energy by integrating the power trace over the request window.

The “average power per request” and “energy over the request window” that we report are the means of these ten per-token values for that specific (model, L) run. Power and energy can provide information on how much battery a request consumes, which is critical as operators need to plan on-device workloads. We take the average power of the requests because it helps us reduce bias from uneven sampling and OS noises. We also record peak RAM during the run, which allows us to spot any out-of-memory failures and how memory scales with the length of the prompt, addressing RQ1 and RQ3. As we keep increasing the prompt length, the Jetson eventually freezes because of

system throttling. We record the prompt length before sending the prompt to the model, so even if the model crashed, we can still relate it to the latency, power, and memory behavior for RQ3.

4.4 Procedures

Initial setup

Before running any experiments, we perform a one-time initial setup on the Jetson. As described in Sections 2.2 and 4.2, we flash JetPack 5.1.2 onto the Jetson Xavier NX and record the image information, install the Jetson builds of Python and PyTorch, and pin versions by saving a `pip freeze`. After this initial setup, each experiment run follows the same procedure below:

1. Start `tegrastats` with timestamps aligned to the Python process.
2. Invoke the benchmarking script to generate a fixed number of output tokens.
3. Log latency, power per token and energy, peak memory usage, and model size on disk to a CSV file.
4. Stop `tegrastats` and move to the next prompt length.

Experiment

1. In the experiments reported in this thesis, we used prompt lengths from L=128 up to L=512 in steps of 64, and for larger models we stopped earlier once runs became unstable.
2. For each prompt length L:
    - Construct prompts by taking a sliding window of length L from a fixed corpus.
    - Send the request to the model.
    - In the implementation used for these experiments, instead of sending 10 separate requests, we generate 10 output tokens within a single request at each prompt length L and treat these 10 token windows as the repeated measurements.

    For each request:
    - Record t_start just before feeding the prompt, then record t_end after the final token.
    - Parse power samples whose timestamps fall inside each token window; compute mean power usage per token; then average across tokens to get avg_power_request.
    - Compute energy_request by integrating the in-window power samples (preferred) or avg_power_request × (t_end − t_start). In the current script, we compute energy separately for each token window as (average power in mW × token latency in seconds), then take the mean of these 10 values to obtain the energy estimate for that (model, L) run.
    - Capture any failures (timeout threshold, for example: 10 s for L < 512, 20 s for L > 768, system crash once L > 512, etc.).

- Stop at hard failure (OOM/system crash) or continue to the next prompt length if stable.

Logging & Reproducibility

We saved raw tegrastats logs and environment snapshot files per run. For each prompt length, our script calculated the average latency, power and energy, and stored them in the corresponding summary row in the csv file. We kept a README.md with all used commands, commit hashes, and result directory layout. By doing so, we ensured the same evaluation methods were applied to other models/hardwares with minimal efforts in the future.

4.5 Analysis Methods

Latency: To measure the latency for each model, we generate one request per model at different prompt lengths. We then calculate the average latency across the 10 generated tokens instead of across 10 separate requests. We measure the latency with this method because it directly reflects whether the system is responsive enough for interactive edge usages that require quick outputs, and it is essential to evaluate the consistency of latency across 10 tokens to limit bias.

Power: We calculate the average power per token window and across the 10 token windows for each request to estimate the power usage for each model at different prompt lengths. Power is measured because it reveals how a given model and prompt length combination stresses the device's power and thermal limits. It can be used when engineering decisions are being made for which model should be used for different tasks.

Energy: We compute energy separately for each token window as the average power in mW multiplied by that token's latency in seconds, then take the mean of these 10 values as the energy estimate. Energy is an important metric for language models in edge environments since resources are very limited. It gives the operators essential information on the battery cost per token and how many requests can be supported within a fixed energy budget.

Memory and storage: For each model at each prompt length L, we also record the peak GPU memory versus L. This tells us how quickly each model consumes the limited 8 GB memory budget as inputs grow. In addition, we measure the model size on disk so that teams can see how many models and related artifacts can realistically fit on the device.

Failure: We create limit plots marking the first observed failure or sharp latency spike that coincides with power throttling. We also document hard failures (OOM/system crash) and link them to the corresponding prompt length.

5. Results

This section presents the collected data from our experiment. We evaluated six small language models on this device: DistilGPT2-82M, GPTNeo-125M, OPT-125M, OPT-350M, SmolLM-135M, and SmolLM-360M. For each model we tested different prompt lengths L through a fixed set of values and generated 10 output tokens per request. For each model, as the prompt length increases, we recorded the average end-to-end latency, average power during the request, and energy per token computed from the power trace.

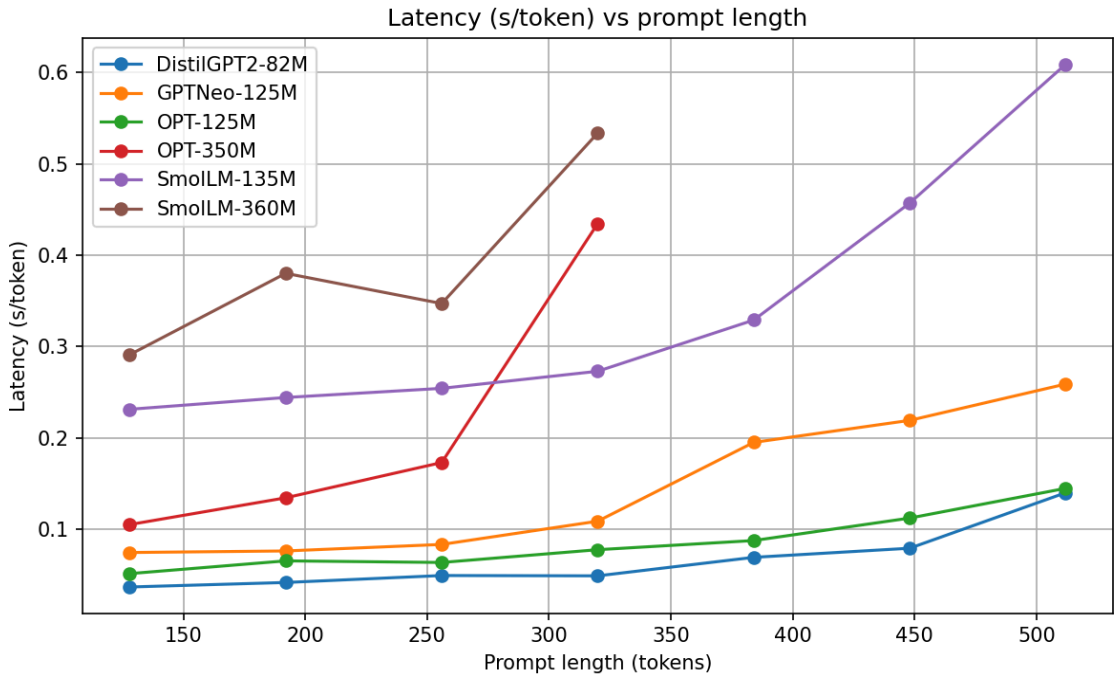


Figure 5.1 Latency VS Prompt Length

Starting with DistilGPT2-82M, the latency per token is the lowest across all models for all prompt length. Its latency grows steadily but remains lower than 0.1 s/token until above 500 prompt length. Two other models that share similar trends are OPT-125M and GPTNeo-125M. SmolLM-135M acts as the outlier amongst other models that trained with less than 150M parameters. Two models that were trained with more than 350M parameters - SmolLM-360M and OPT-350M all experienced latency spikes at around 250 prompt length and crash near 300 input tokens.

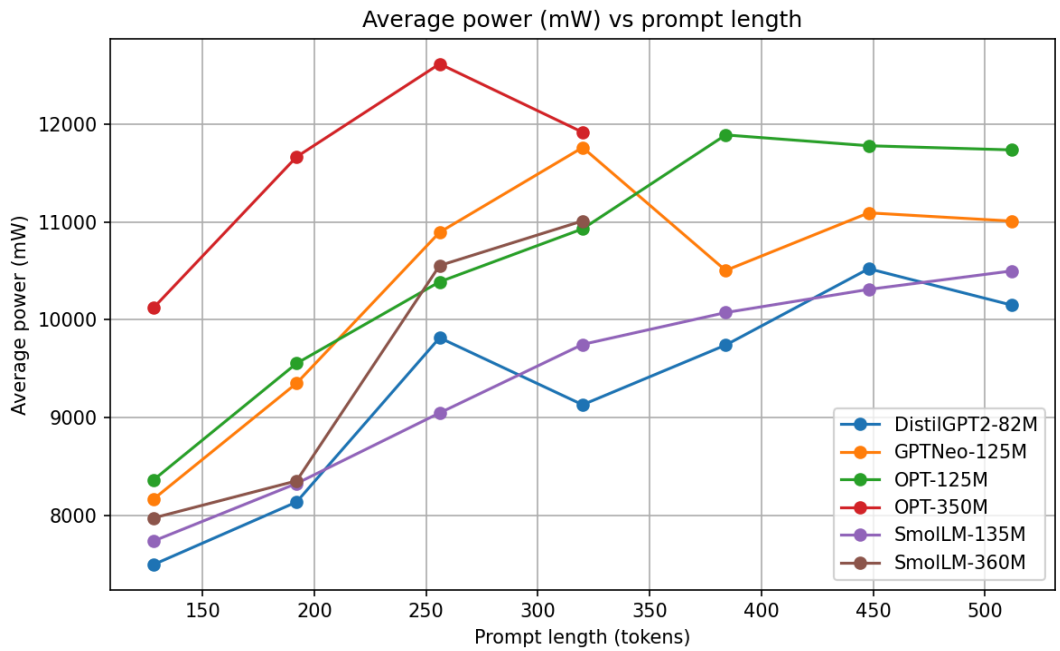


Figure 5.2 Power VS Prompt Length

The average power usage for all six models are in the range of 7 watts to 13 watts. DistilGPT2-82M starts at the low end of this range, drawing around 7-8 watts at L=128 and rising gradually to around 10 watts at L=512. OPT-125M and GPTNeo-125M shared a similar pattern, starting around 8 watts at 150 token length and increasing into the 10 - 12 watt range as the prompt length increases. SmolLM-135M shows the same general trend. OPT-350M is the outlier in this case, operating near the top of the range. It starts close to 10 watts at L=128, reaches a peak around 12.5 watts at L=256, and remains in the high 11 - 12 watt band at L=320. SmolLM-360M starts near 8 watts at L=128 and rises to about 11 watts at L=320. Overall, these curves show that the Jetson Xavier NX runs all six models within a similar power envelope; most of the differences between models come from latency rather than from large differences in average power.

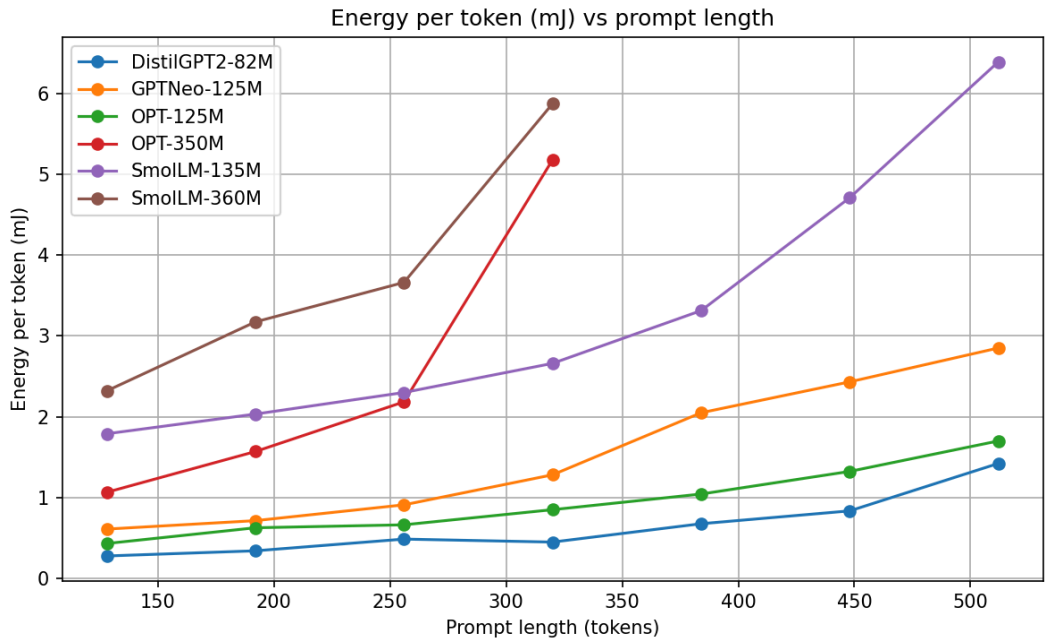


Figure 5.3 Energy VS Prompt Length

We obtained the energy per token by combining the latency and power data together. DistilGPT2-82M has the lowest energy cost among the 6 models that we experimented with. At L=128 it consumes well under one mJ(millijoules) per token, and even at L=512 it remains close to 1.5 mJ per token. OPT-125M costs a bit more energy than DistilGPT2-82M, starting a bit higher at L=128 and reaching around 1.7 - 1.8 mJ per token at L=512. GPTNeo-125M sits in a middle band: it begins near 0.6 mJ per token and increases smoothly to about 2.8 mJ per token at L=512. SmolLM-135M costs around 1.8 mJ per token at L=128, and increases more sharply at longer prompts. By L=512 it uses a little more than 6 mJ per token, making it the most energy-consuming model among those that have energy data for the full range of prompt lengths. OPT-350M and SmolLM-360M are only measured up to L=320, but both show clear signs of approaching a practical limit. OPT-350M grows from just above 1 mJ per token at L=128 to a little more than 2 mJ per token at L=256, then jumps to just over 5 mJ per token at L=320. SmolLM-360M has a similar pattern, increasing from around 2.3 mJ per token at L=128 to roughly 3.7 mJ per token at L=256 and then rising to almost 6 mJ per token at L=320. Together, these energy curves make it clear that longer prompts are cheap for the smallest models, but become increasingly expensive - and eventually unstable - for the mid-size and larger models on this device.

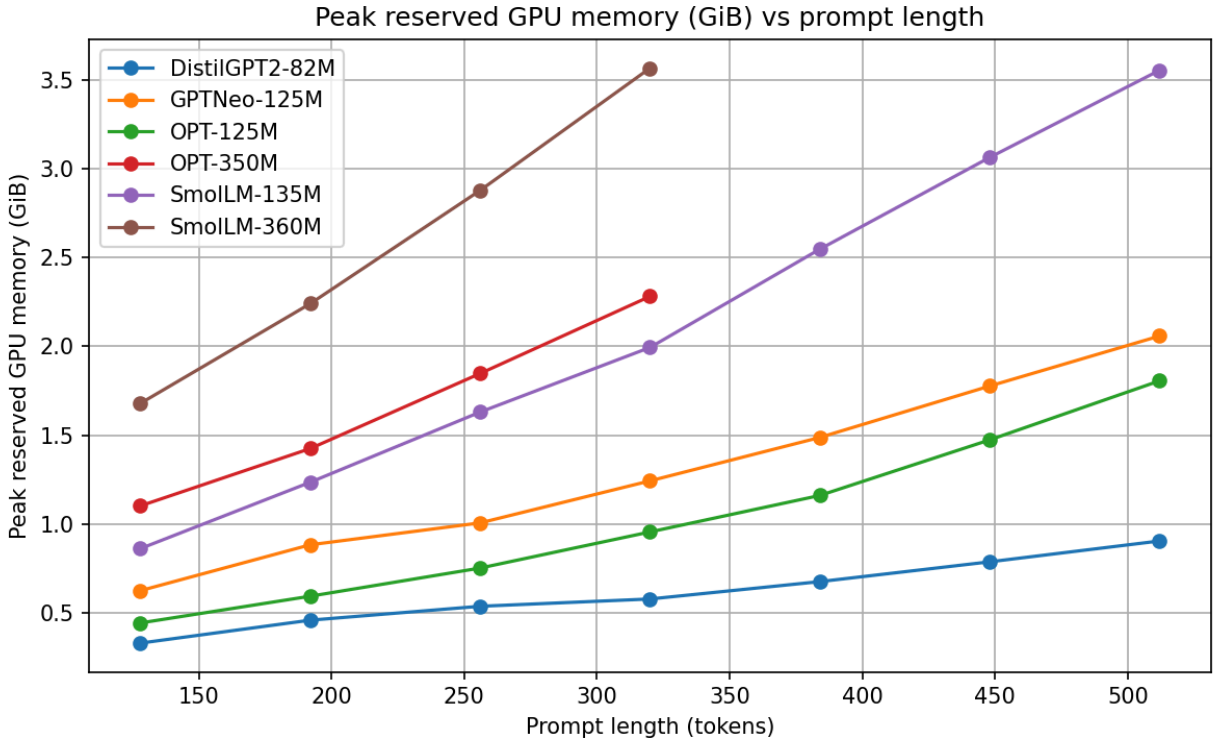


Figure 5.4 Peak Reserved GPU Memory VS Prompt Length

From Figure 5.4, we can see how peak reserved GPU memory grows as prompt length increases for all six models. DistilGPT2-82M uses the least amount of GPU memory. It starts around 0.33 GB at L=128 and stays under 1 GB even at L=512. OPT-125M and GPTNeo-125M use slightly more reserved GPU memory than DistilGPT2-82M in the middle band. They begin between about 0.5 and 0.8 GB at L=128 and increase steadily to roughly 1.8-2.0 GB by L=512. SmolLM-135M uses much higher memory when compared to models trained with less than 150M parameters. It starts near 1.1 GB at L=128 and rises in a nearly linear way to more than 3.5 GB at L=512, making it the most memory-hungry model that still completes the full sweep. OPT-350M and SmolLM-360M consume the most memory at shorter lengths and do not have stable runs beyond L=320. By L=320, OPT-350M is already above 2 GB and SmolLM-360M is above 3.5 GB of reserved GPU memory.

Model	**Model size (MB)**	**% of total disk**
DistilGPT2-82M	**340 M**	**0.29 %**
GPTNeo-125M	**505 M**	**0.42 %**

OPT-125M	**241 M**	**0.20 %**
OPT-350M	**634 M**	**0.53 %**
SmolLM-135M	**517 M**	**0.43 %**
SmolLM-360M	**1.4 G**	**1.20 %**

Table 5.5 SLM Model Sizes

Table 5.5 reports how much disk space each model checkpoint occupies on the Jetson Xavier NX. OPT-125M requires the least amount of space at 241 MB, using about 0.20% of the device's total disk space. DistilGPT2-82M is slightly larger at 340 MB (0.29%), while GPTNeo-125M and SmolLM-135M both sit around the half-gigabyte mark at 505 MB (0.42%) and 517 MB (0.43%). OPT-350M is the second largest model at 634 MB, or about 0.53% of disk. SmolLM-360M is the only model that crosses the one-gigabyte line at roughly 1.4 GB, which corresponds to about 1.20% of total disk space. Overall, not a single model has major storage requirements on Jetson.

6. Discussion

6.1 Interpretation of the Results

The results show a clear pattern that as the prompt length increases, the latency and energy per token rise for all models. The rate of increase and the ceiling where runs become impractical depend strongly on model size. For the two smallest models, DistilGPT2-82M and OPT-125M, they both have low-latency, low-energy characteristics even at L=512. In contrast, mid-size and larger models, such as OPT-350M and SmolLM-360M, become expensive or unstable at much shorter prompt lengths.

For DistilGPT2-82M, latency per token remains well under a tenth of a second until prompt lengths above L=500, and energy per token grows slowly from well below one mJ at L=128 to around one and a half mJ at L=512. OPT-125M has a very similar profile. These two models run inside the same eight-to-twelve watt power envelope as the others, but convert that power into tokens more efficiently at every prompt length. GPTNeo-125M's performance is between that of OPT-125M and DistilGPT2-82M.

SmolLM-135M behaves differently. At L=128 it already reached the latency and energy per token in the range that GPTNeo-125M only reaches at much longer prompts. Its curves initially increase gradually, then steepen as L approaches L=512. By that point, SmolLM-135M has the highest energy per token among the models that can be run at L=512. This does not mean that SmolLM-135M is unusable, but it shows that on this device longer prompts with this model

consume a large fraction of the available latency and energy budget. It's a good reminder for the operators that there is no one size fits all solution, even if models were trained with the same number of parameters, the model performances might still differ due to factors like model uniqueness or architectures.

The two largest models, OPT-350M and SmolLM-360M, show clear limitations near L=320. Both models run stably with prompt lengths up to L=256, where latency and energy per token are higher than those of the smaller models but still follow a gradual trend. At L=320, both models experienced a sharp jump in latency and energy, and attempts to run them at longer prompt lengths often result in timeouts or crashes. However, the average power remains within the same range as before. As the system crashes, the system UI would display a "System Throttling" message before freezing. This pattern suggests that the Xavier NX is hitting a throttling that stops computation without radically changing average power draw. The system performance and crash helped us verify that for these larger models, a practical prompt-length limit on this configuration lies somewhere below L=320.

Our results properly answer the three research questions that we proposed at the beginning. For RQ1, the plots of three metrics (latency, power, energy) help us track the practical ranges of prompt length L for each model on this Jetson configuration, rather than a single universal limit. For RQ2, our data reflected the trade-offs among latency, energy, and safe prompt length: keeping latency within an interactive bound often required either accepting higher energy cost per token or shrinking the prompt-length budget, especially for larger models. For RQ3, the sharp increases and system failures caused by models at prompt length L=320 for OPT-350M and SmolLM-360M demonstrated where and how deployments start to fail: not through a gradual drift but through a sudden change in behavior at a particular combination of model size, prompt length, and device.

These findings support the claim of the thesis. A reproducible stress test on a pinned Jetson image to test different prompt lengths, generate a fixed number of tokens, and record per-token latency, power, and energy, can reveal model-specific operating limits on a real embedded GPU. Our results demonstrated how we can utilize the metrics that we collected from different models as a concrete, budgeted quantity rather than an abstract parameter in a configuration file.

In addition to latency and energy, the memory results give a clearer picture of how close each model comes to the GPU limits as prompt length grows. Peak reserved GPU memory increases in a roughly linear way with L for all six models, but the starting point and slope are very different. DistilGPT2-82M remains below 1 GiB even at L=512, and OPT-125M and GPTNeo-125M stay under about 2 GiB. SmolLM-135M uses significantly more memory and reaches more than 3.5 GiB at L=512. OPT-350M and SmolLM-360M consume the most memory at shorter lengths and already reserve more than 3.5 GiB by L=320, which is the point where they become unstable. This means that while memory alone is not yet the hard failure mode on an 8 GiB Xavier NX for a single model, larger models and longer prompts quickly eat into the headroom that would be needed for other processes or additional models.

In terms of storage space needed, all of the models occupy less than about 1.5 GB on disk, which is less than two percent of the Jetson's total storage, but there is still a clear spread: OPT-125M and DistilGPT2-82M are noticeably smaller than the others, while SmolLM-360M is by far the largest. This suggests that storage is not the primary constraint when deploying one model on edge devices like Jetson, but it will matter more if teams want to keep multiple SLMs, datasets, or container images on the same device.

6.2 Rebuttals & Counterarguments

One of the main rebuttals is that averages may hide short power spikes that matter during deployment. We addressed this by aligning tegrastat samples to token windows and computing integrated energy. For example: a spike that happens during token 7 is attributed to token 7, not smeared across the whole request. We can then see which tokens cause bursts and how often they happen. Another concern is that currently the Jetson crashes after a specific length for both models (512 for SmolLM-135M and 320 for SmolLM-360M). How do we effectively capture the last data before the device froze? To resolve this issue, we incremented the prompt length at a shorter interval. Also, we make sure data is flushed even if the model hangs. Power logging runs in a separate process and writes to disk continuously with timestamps. The benchmark writes a row to CSV after every request, not at the end of a batch. By taking this approach we made sure we record the logs necessary before the model crash.

In later experiments, we extended the same design to four additional models (DistilGPT2-82M, GPTNeo-125M, OPT-125M, and OPT-350M) and saw similar concerns about hidden spikes, so the token-window alignment is still important. For the larger models such as OPT-350M and SmolLM-360M this also helped reveal sharp jumps in latency and energy around L=320 even when average power stayed in the same general range, which supports the claim that we are seeing real limits rather than just noise in the measurements. A related rebuttal is that the current results do not include accuracy, so they only show the cost side of the trade-off. Calculating accuracy for every prompt length on an embedded device is challenging because it requires extensive runs, well-defined benchmark corpus, and handling of non-determinism for SLMs; It will also greatly increase runtime and make it difficult to differentiate hardware effects from task effects. For this reason, this thesis focuses first on latency, power, and energy as core deployment constraints, and leaves a more complete accuracy study to the future work section.

6.3 Qualifiers & Threats to Validity

Scope: As mentioned earlier, all of the results are related to the same NVIDIA Jetson Xavier NX (8 GB) on JetPack 5.1.2, PyTorch runtime, SmolLM-135M/360M, the evaluated prompt lengths, and 10 generated tokens per request. In the final set of experiments, we also include DistilGPT2-82M, GPTNeo-125M, OPT-125M, and OPT-350M, still generating 10 tokens per request at each prompt length.

Internal validity. We aligned built-in system times with power sampling, and removed incomplete runs but OS noise can bias energy downward if samples are dropped or runs were not finished properly. Temperature history could also shift the throttling point from back-to-back runs leading to minor variation in when throttling occurs. Another limitation is that we only ran the benchmark once per

model at each prompt length rather than repeating runs and averaging across them, so we may miss some run-to-run variation. We also used a fixed prompt pattern to control the input, which means that domain-specific or adversarial prompts that might stress the models differently or cause performance degradation were not directly tested here.

External validity. Results may not generalize to other Jetson models (e.g., Orin), other runtimes (TensorRT/ONNX Runtime), alternative decoding settings, or longer output lengths. We would need to test more different models and hardware in order to test the generalization and adaptability of our stress test mechanism.

6.4 Impact and Implication

Industrial Impact: For edge scenarios such as robot instruction, voice assistants in internet-denied areas, on-vehicle triage, or small UAV mission copilots, mission operators have a well-defined way to treat prompt length as a budgeted resource just like battery or bandwidth. Our evaluation mechanism provides a robust way to generate a pre-mission checklist; they can now follow the path of picking a model -> set a safe maximum input length -> estimate battery per request -> kick off the mission. When certain missions require longer inputs, engineers can perform either horizontal or vertical scaling based on the prompt length curve versus specific model.

The expanded model set gives operators more concrete choices. DistilGPT2-82M and OPT-125M appear well-suited for interactive workloads that need prompt lengths up to L=512 with relatively low latency and energy per token. The GPTNeo-125M sits right in the middle with higher cost but smooth behavior. When it comes to larger models, OPT-350M and SmolLM-360M add higher-capacity options but show much sharper increases in latency and energy as prompt length approaches L=320, which signals that they should be used more carefully on this device and only within shorter prompt ranges if interactivity and battery life are important.

Across the six models we tested, SLMs with about 150M parameters or fewer stayed usable for interactive tasks up to prompt lengths of roughly L = 512 on the Jetson Xavier NX. For larger models around 350M parameters, we observed that prompts at L = 320 already introduce noticeable delays, so practitioners who need interactive response times should keep L < 256 on this device. Engineering teams can translate these limits directly into corresponding tasks. When a task requires longer inputs, the data can be used to identify where to intervene (quantize, change runtime, or partition prompts). It also helps with engineering decisions about what are the suitable tasks that can be deployed on Jetson and when would be a good time to introduce more computing power for different computing tasks.

Theoretical Impact: Our study demonstrates that power monitoring with a simple time recording method is sufficient to reveal resource constraints on embedded GPUs. This offers a quick and reproducible template for SLM evaluation at the edge and motivates modeling work based on prompt length and power availability. The additional models strengthen this point by showing the same basic pattern across several architectures and parameter scales, and by making it clear

that prompt length and model size must be considered together when reasoning about edge deployments.

Future Work: Currently, one of the most popular ways for reducing model size is applying quantization techniques for better model efficiency. In the future, we can try adapting INT8/INT4 quantization and alternative runtimes (TensorRT/ONNX Runtime) to test whether optimization shifts the L=320 boundary for 350M+ models or reduces 100M+ models' energy growth. Currently, we only analyzed the smoothness of the operation from latency and energy perspectives, but didn't really account for output quality. We should measure task-accuracy at each length to complete the quality-latency-energy trade-off, and perform the same set of experiments on Orin-class devices to assess portability of the safe ranges. A follow-on study could also design a small, edge-friendly accuracy benchmark that can be run at multiple prompt lengths on the Jetson without overwhelming the device, so that quality, latency, power, and energy can be evaluated together in a more balanced way.

7. Conclusion

This thesis asked a practical question: how far can small language models be pushed on a resource-constrained NVIDIA Jetson before performance breaks down? There are very few methods for engineers to discover those limits without reinventing redundant trial and error processes. To answer that question, we designed and implemented a reusable stress-test framework that fixes the hardware and software image, steadily increases prompt length, and documents the latency, power, energy, memory, and storage for each model.

Using this framework, we evaluated six small language models with different prompt lengths and observed how their behavior changed as input length increased. The results demonstrated a clear pattern: smaller models remained stable and predictable even at longer prompts, while larger models became expensive and unstable earlier.

These observations directly answer the research questions posed in the introduction. For RQ1, the latency, power, energy, and memory results revealed the practical prompt length ranges for each model in the tested environment. For RQ2, the comparison across models revealed a clear trend in these trade-offs: configurations that maintain lower latency tend to pay for it with higher energy cost per token or a reduced safe prompt-length budget, and this tension becomes more pronounced as model size increases. For RQ3, the sharp jumps and failure patterns show that SLMs tend to fail abruptly through throttling, timeouts, or crashes without much warning.

One potential next step is to consolidate these outputs in a single UI so that the framework becomes a quick decision hub for model and device selection, first for Jetson deployments and later for larger models on more capable hardware. Lastly, if we manage to incorporate accuracy versus prompt length in future works, we will be able to analyze the full quality, prompt length, and power trade-offs – all done with one tool that can be deployed easily in any environment.

8. References

[1] “Making LLMS Even More Accessible with Bitsandbytes, 4-Bit Quantization and Qlora.” *Hugging Face - The AI Community Building the Future.*, huggingface.co/blog/4bit-transformers-bitsandbytes. Accessed 21 Sept. 2025.

[2] “Quantize ONNX Models.” *Onnxruntime*, onnxruntime.ai/docs/performance/model-optimizations/quantization. Accessed 21 Sept. 2025.

[3] “Nvidia Triton Inference Server.” *Performance Analyzer - NVIDIA Triton Inference Server*,

docs.nvidia.com/deeplearning/triton-inference-server/archives/triton-inference-server-2310/user-guide/docs/user_guide/perf_analyzer.html.

Accessed 21 Sept. 2025.

[4] “Benchmark Mlperf Inference: Edge: Mlcommons v3.1 Results.” *MLCommons*,

9 Sept. 2025, mlcommons.org/benchmarks/inference-edge/.

[5] “Get Started with Jetson Xavier NX Developer Kit.” *NVIDIA Developer*, developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit.

Accessed 21 Sept. 2025.

[6] *Nvidia Computex 2024 Keynote.Pdf*, media.iprsoftware.com/219/files/20245/NVIDIA

Computex 2024 Keynote.pdf. Accessed 22 Sept. 2025.

[7] Chen, Hao, et al. “Inference Performance Evaluation for LLMS on Edge Devices with a

Novel Benchmarking Framework and Metric.” *arXiv.Org*, 15 Aug.

2025, arxiv.org/abs/2508.11269.

[8] Dao, Tri, et al. “Flashattention: Fast and Memory-Efficient Exact Attention with IO-Awareness.” *arXiv.Org*, 23 June 2022, arxiv.org/abs/2205.14135.

[9] Thurgood, Steven, et al. “Implementing Slos.” *Google SRE - Continuous Improvement To Get Reliability*, Google SRE,

sre.google/workbook/implementing-slos/. Accessed 28 Sept. 2025.

[10] Liang, Percy, et al. “Holistic Evaluation of Language Models.” *arXiv.Org*, 1 Oct. 2023, arxiv.org/abs/2211.09110.

[11] Wang, Fali, et al. “A Comprehensive Survey of Small Language Models in the Era

of Large Language Models: Techniques, Enhancements, Applications, Collaboration

with LLMS, and Trustworthiness." *arXiv.Org*, 28 Dec. 2024, arxiv.org/abs/2411.03350.

[12] Jiajun Xu Meta {jjxu217}@meta.com &Zhiyuan Li∗ Nexa AI {zack}@nexa4ai.com &Wei Chen∗ Nexa AI {alexchen}@nexa4ai.com &Qun Wang∗ Computer

Science Department, and Equal contribution. *On-Device Language Models: A Comprehensive Review*, arxiv.org/html/2409.00088v1. Accessed 28 Sept. 2025.

[13] Labs Cambridge, SiYoung Jang Nokia Bell, et al. *Edge-First Language Model Inference: Models, Metrics, and Tradeoffs*, arxiv.org/html/2505.16508v1. Accessed 28 Sept. 2025.

[14] MLPERF inference benchmarks. MLPerf Inference Benchmarks - MLPerf Inference Documentation. (n.d.). https://docs.mlcommons.org/inference/

Supplemental Materials

Stress Test Script

```python
#!/usr/bin/env python3


# Prompt-length benchmark for Jetson.
# Records token latency, power, energy, memory, and disk usage.


import time
import csv
import math
import argparse
from pathlib import Path
import os
import sys
import gc
import shutil


import torch
from transformers import AutoModelForCausalLM, AutoTokenizer


# local utils
here = Path(__file__).resolve().parent
for p in (here, here / "utils", here.parent / "utils"):
    if p.exists():
    sys.path.append(str(p))
```

```
from run_tegrastats import start_tegrastats, collect_power_samples



def sync_gpu():
    if torch.cuda.is_available():
    torch.cuda.synchronize()



def clear_mem():
    if torch.cuda.is_available():
    torch.cuda.empty_cache()
    gc.collect()
    time.sleep(0.05)



def average(xs, default=float("nan")):
    xs = [v for v in xs if v is not None and math.isfinite(v)]
    if not xs:
    return default
    return sum(xs) / len(xs)



def read_meminfo():
    total = avail = None
    try:
    with open("/proc/meminfo") as f:
        for line in f:
        if line.startswith("MemTotal:"):
```

```
                    total = int(line.split()[1]) * 1024
              elif line.startswith("MemAvailable:"):
                    avail = int(line.split()[1]) * 1024
       except:
       pass
       return total, avail



def get_disk_stats(path):
       try:
       d = shutil.disk_usage(str(path))
       return d.total, d.free
       except:
       return None, None



def build_prompt(tok, n):
       # Build a prompt of approximately n tokens.
       base = "The quick brown fox jumps over the lazy dog. "
       text = base * (n // 9 + 2)
       ids = tok(text, return_tensors="pt",
add_special_tokens=False)["input_ids"][0]
       return tok.decode(ids[:n], skip_special_tokens=True)



def run_one(model_name, prompt_len, n_tokens, writer, log_every, disk_root):
       dev = "cuda" if torch.cuda.is_available() else "cpu"
```

```
        tok = AutoTokenizer.from_pretrained(model_name)
        if tok.pad_token is None and tok.eos_token:
        tok.pad_token = tok.eos_token

        model = AutoModelForCausalLM.from_pretrained(
        model_name,
        torch_dtype=torch.float16 if dev == "cuda" else None
        ).to(dev)
        model.eval()

        # GPU memory info
        gpu_total = None
        if torch.cuda.is_available():
        p = torch.cuda.get_device_properties(0)
        gpu_total = getattr(p, "total_memory", None)

        sys_total, _ = read_meminfo()
        disk_total, disk_free = get_disk_stats(disk_root)

        # Prompt and input IDs
        prompt = build_prompt(tok, prompt_len)
        ids = tok(prompt, return_tensors="pt",
add_special_tokens=False).to(dev)["input_ids"]

        # tegrastats sampling
        q, stopper, thread = start_tegrastats()
        while q.empty():
        time.sleep( . )
```

```
# Warmup
_ = model(ids)
_ = model(ids)

print(f"[{model_name}] len={prompt_len}  tokens={n_tokens}")

started = time.time()

# Tracking
lat_list = []
pow_list = []
e_list = []
ga_list = []
gr_list = []
sys_av_list = []

peak_rsrv = 0

for i in range(n_tokens):
sync_gpu()
t0 = time.time()

out = model(ids, use_cache=True)
nxt = torch.argmax(out.logits[:, -1, :], dim=-1).unsqueeze(0)
ids = torch.cat([ids, nxt], dim=1)

sync_gpu()
```

```
        t1 = time.time()
        lat = t1 - t0

        # Power
        ps = collect_power_samples(q, t0, t1)
        if isinstance(ps, (tuple, list)) and ps:
                pt = ps[0]
        else:
                pt = None

        if pt is not None and pt > 0 and math.isfinite(pt):
                en = pt * lat
        else:
                en = None

        # Memory
        _, sys_av = read_meminfo()

        if torch.cuda.is_available():
                ga = torch.cuda.memory_allocated(0)
                gr = torch.cuda.memory_reserved(0)
                peak_rsrv = max(peak_rsrv, gr)
        else:
                ga = None
                gr = None

        lat_list.append(lat)
        pow_list.append(pt)
```

```
        e_list.append(en)
        ga_list.append(ga)
        gr_list.append(gr)
        sys_av_list.append(sys_av)

        if (i == 0) or ((i + 1) % log_every == 0):
            w = (pt / 1000.0) if pt else 0.0
            print(f"  tok {i+1}/{n_tokens}:  {lat:.3f}s  {w:.2f}W")

        writer.writerow({
            "row_type": "token",
            "model": model_name,
            "prompt_len": prompt_len,
            "token_id": i + 1,
            "token_text": tok.decode(nxt[0], skip_special_tokens=True),
            "latency_sec": lat,
            "power_total_mw": pt or "",
            "energy_mj": en or "",
            "gpu_alloc_B": ga or "",
            "gpu_reserved_B": gr or "",
            "sys_mem_avail_B": sys_av or "",
        })

        stopper.set()
        thread.join()

        total = time.time() - started
        tps = n_tokens / total if total >  else float("nan")
```

```
        # Summaries
        al = average(lat_list)
        ap = average(pow_list, default=0.0)
        ae = average(e_list)
        aa = average([x for x in ga_list if x is not None])
        ar = average([x for x in gr_list if x is not None])
        asys = average([x for x in sys_av_list if x is not None])

        print(f"done: {n_tokens} in {total:.2f}s  tps={tps:.2f}
avg_lat={al:.3f}s")

        writer.writerow({
        "row_type": "summary",
        "model": model_name,
        "prompt_len": prompt_len,
        "token_id": 0,
        "token_text": "",
        "latency_sec": "",
        "power_total_mw": "",
        "energy_mj": ae,
        "tokens_per_sec": tps,
        "avg_token_latency_sec": al,
        "avg_power_mw": ap,
        "gpu_total_B": gpu_total or "",
        "gpu_reserved_peak_B": peak_rsrv or "",
        "gpu_alloc_avg_B": aa or "",
        "gpu_reserved_avg_B": ar or "",
```

```
        "sys_mem_total_B": sys_total or "",
        "sys_mem_avail_avg_B": asys or "",
        "disk_total_B": disk_total or "",
        "disk_free_B": disk_free or "",
        })

        del model, tok, ids
        clear_mem()



def parse_args():
        p = argparse.ArgumentParser(description="Prompt-length benchmark")
        p.add_argument("--model", action="append")
        p.add_argument("--prompt-start", type=int, default=128)
        p.add_argument("--prompt-end", type=int, default=512)
        p.add_argument("--prompt-step", type=int, default=64)
        p.add_argument("--tokens", type=int, default=10)
        p.add_argument("--out", type=Path, default=Path("results.csv"))
        p.add_argument("--log-every", type=int, default=1)
        p.add_argument("--disk-path", type=Path, default=Path("/"))
        p.add_argument("--default-model", default="HuggingFaceTB/SmolLM-135M")
        return p.parse_args()



def main():
        args = parse_args()
        models = args.model if args.model else [args.default_model]
```

```
        args.out.parent.mkdir(parents=True, exist_ok=True)

        fields = [
        "row_type","model","prompt_len","token_id","token_text",
        "latency_sec","power_total_mw","energy_mj",
        "gpu_alloc_B","gpu_reserved_B","sys_mem_avail_B",
        "tokens_per_sec","avg_token_latency_sec","avg_power_mw",
"gpu_total_B","gpu_reserved_peak_B","gpu_alloc_avg_B","gpu_reserved_avg_B",
        "sys_mem_total_B","sys_mem_avail_avg_B","disk_total_B","disk_free_B",
        ]

        with open(args.out, "w", newline="") as f:
        writer = csv.DictWriter(f, fieldnames=fields)
        writer.writeheader()

        for m in models:
                for L in range(args.prompt_start, args.prompt_end + 1,
args.prompt_step):
                try:
                        run_one(m, L, args.tokens, writer, args.log_every,
args.disk_path)
                except RuntimeError as e:
                        print("error:", e)
                        clear_mem()
                        continue


if __name__ == "__main__":
```

```
os.environ["TOKENIZERS_PARALLELISM"] = "false"

main()
```

This Python script was the stress-test framework used to generate all results in the results section.