Understanding the Delivery Delay of Addressed Issues in Large Software Projects

Candidate: Daniel Alencar da Costa

Advisors: Uirá Kulesza and Ahmed E. Hassan

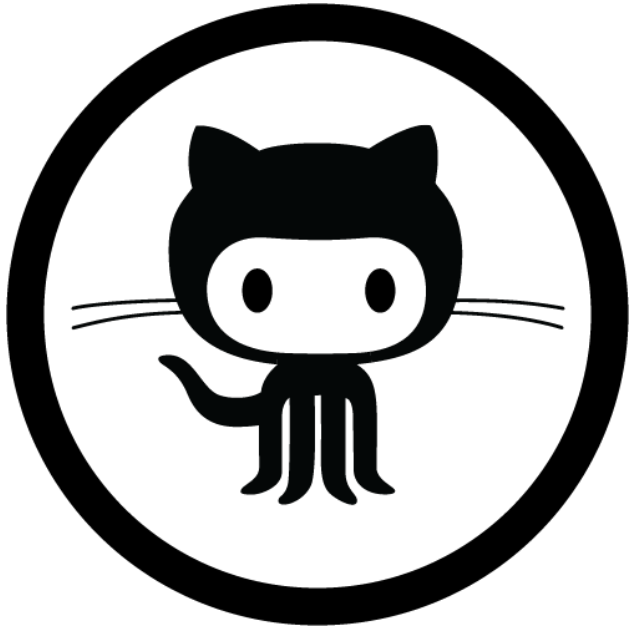
Motivation

Attracting and retaining the interest of users is a key factor for a software project's success

Motivation

Within a globalized world, it is common to see globally spread teams

Such teams use Issue Tracking Systems control the progress of their tasks



A software issue can either describe a bug, an enhancement, or a new feature

Bug fix

New feature

Enhancement

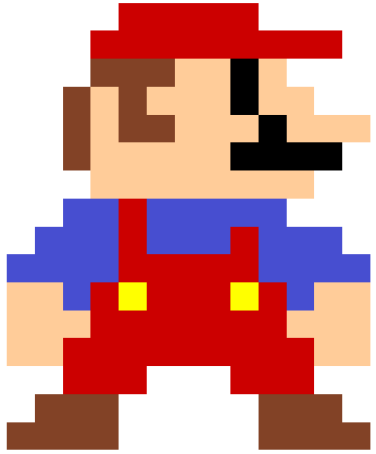
The state of an issue changes from ‘reported’ to ‘fixed’ as it progresses until delivery

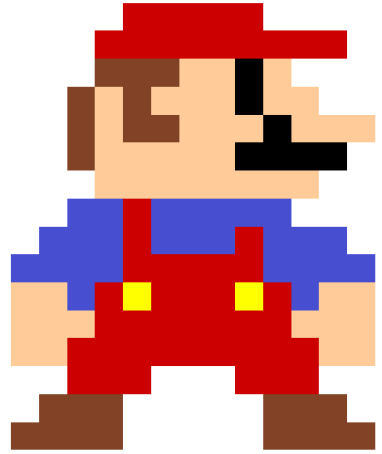
Problem

Although an issue has been addressed, users may still wait some time to experience it

User

Reported → Fixed





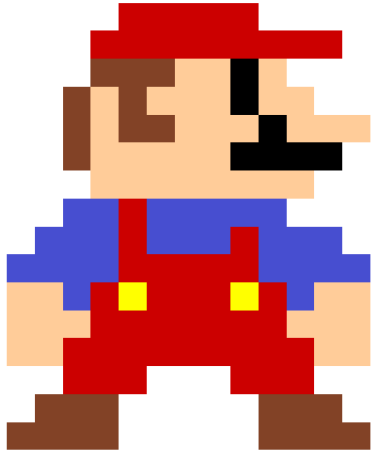
Still…

Problem

Although an issue has been addressed, users may still wait some time to experience it

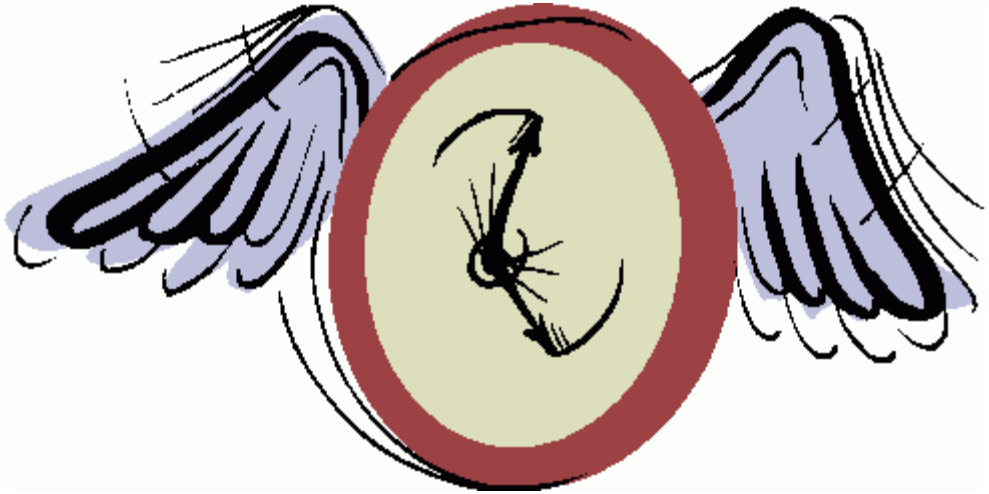
User

Reported → Fixed



Delivered!

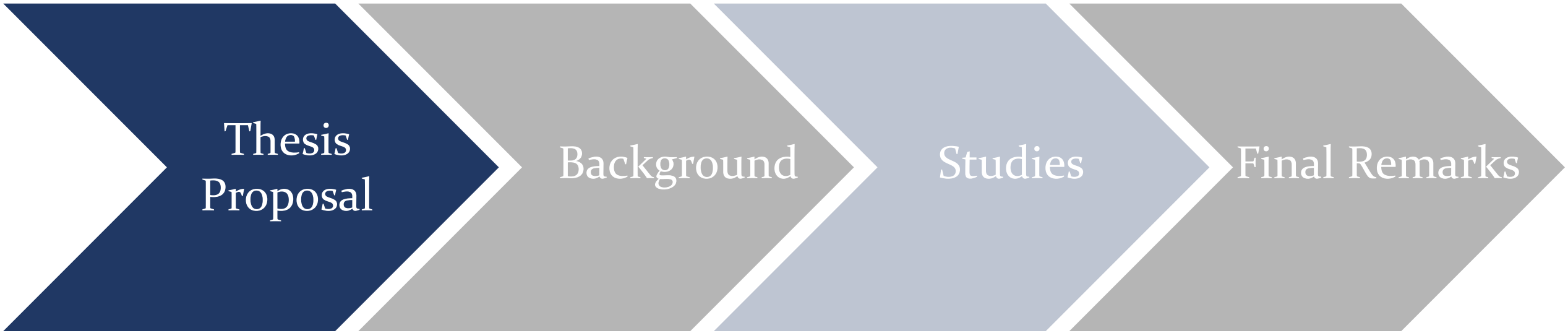
Much effort has been invested on studying the needed time to address an issue



Current limitations

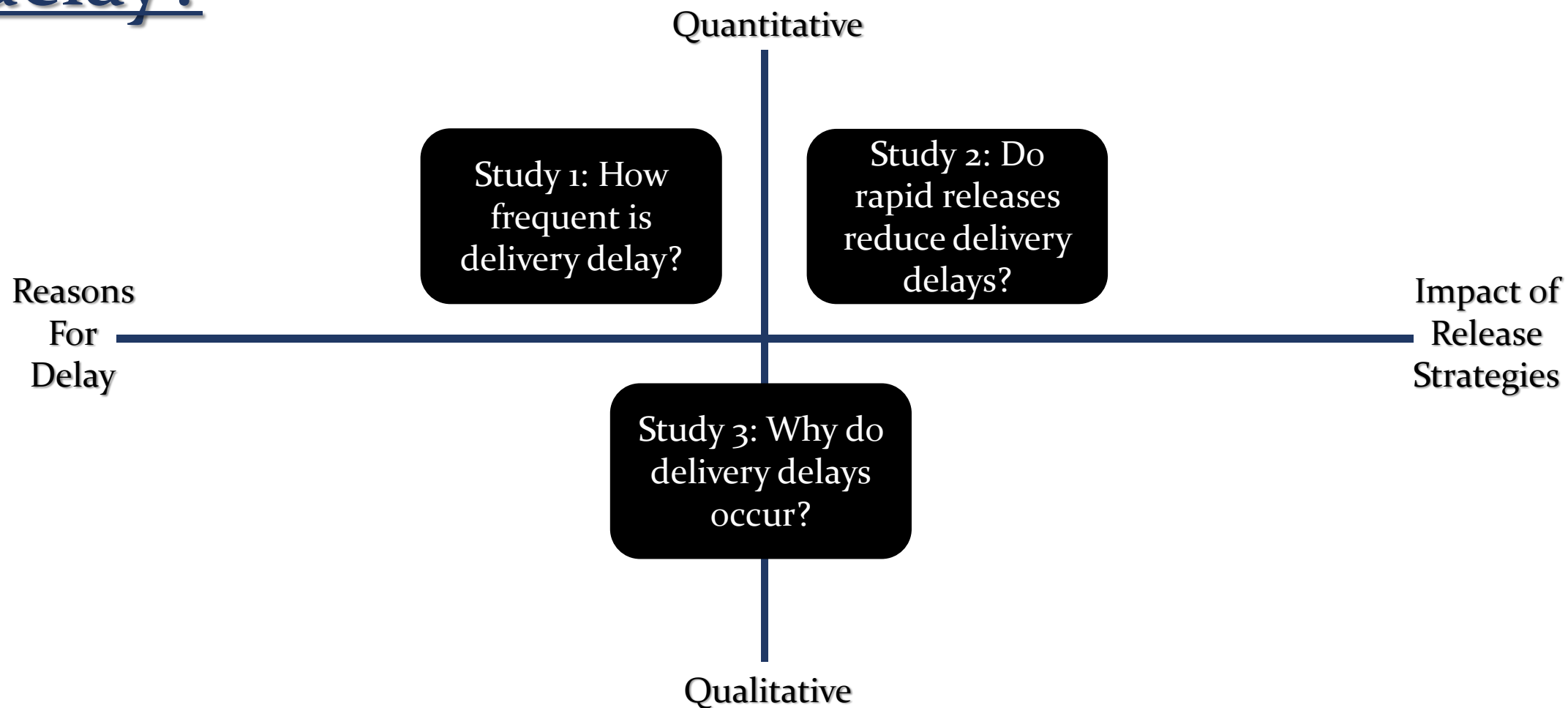
However, studying the required time to deliver addressed issues remains as an open challenge

Agenda



Thesis proposal

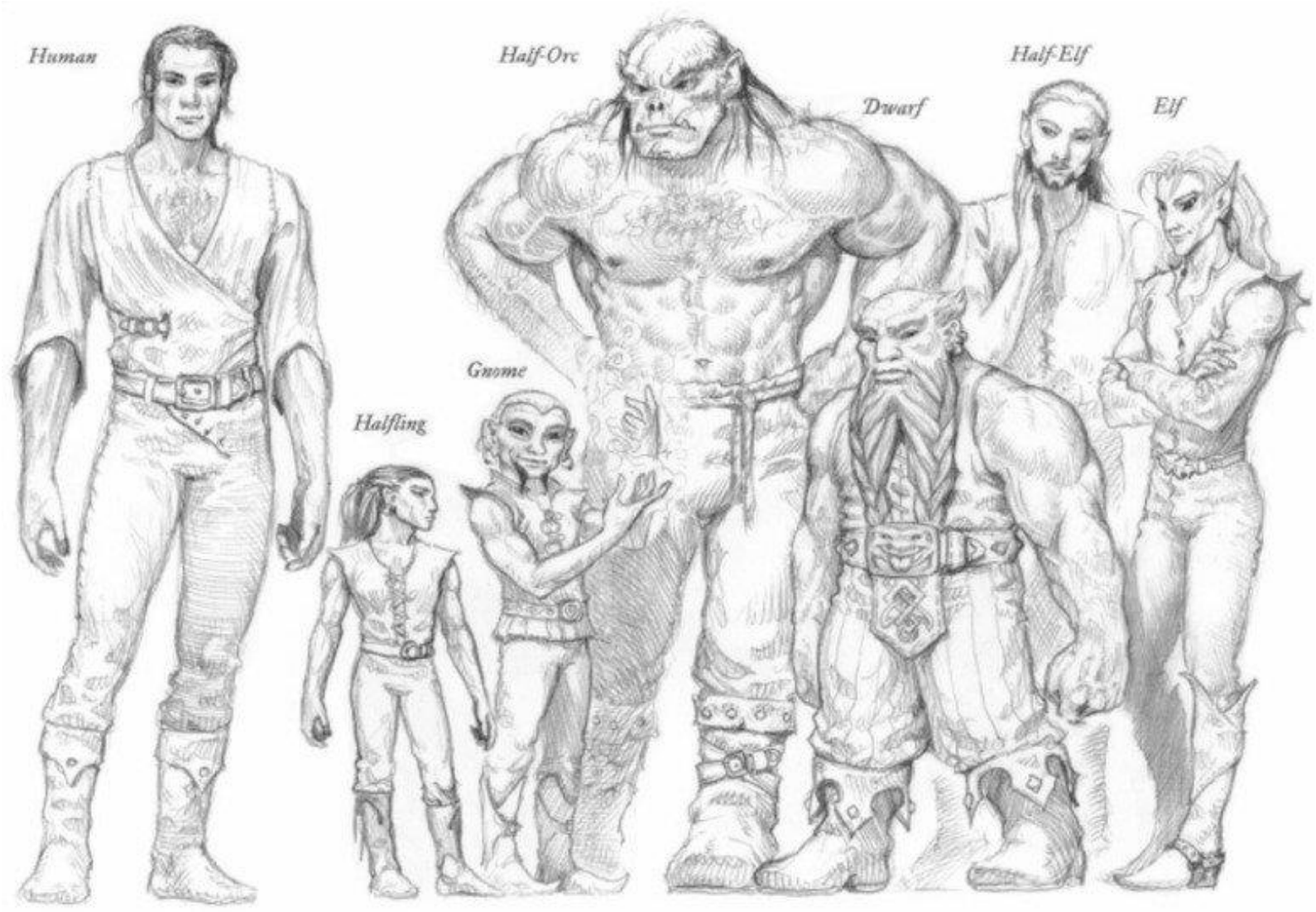
We perform three studies towards the general question: what leads to delivery delay?



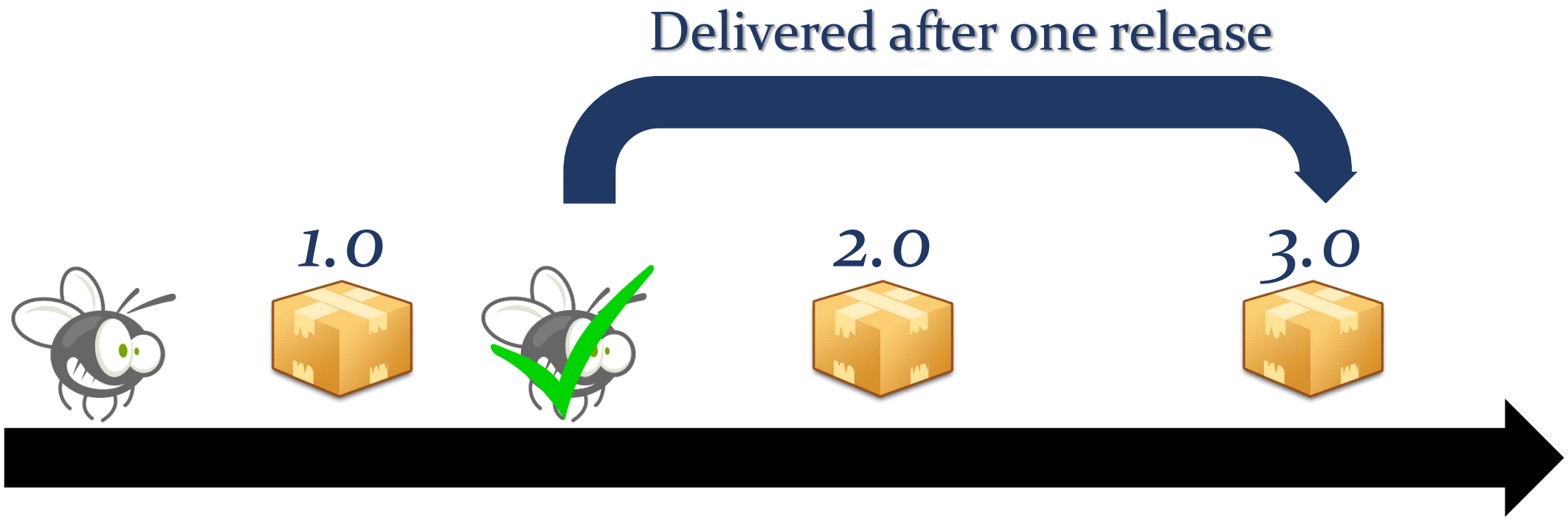
Agenda

Background

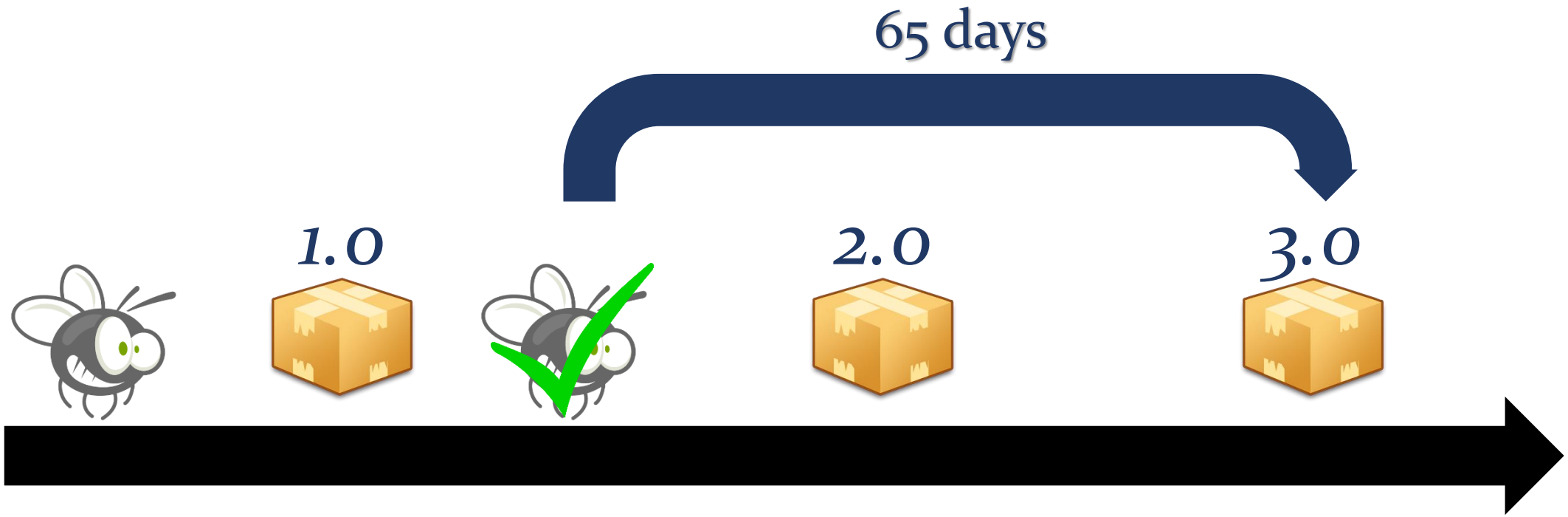
We study different types of delivery delay



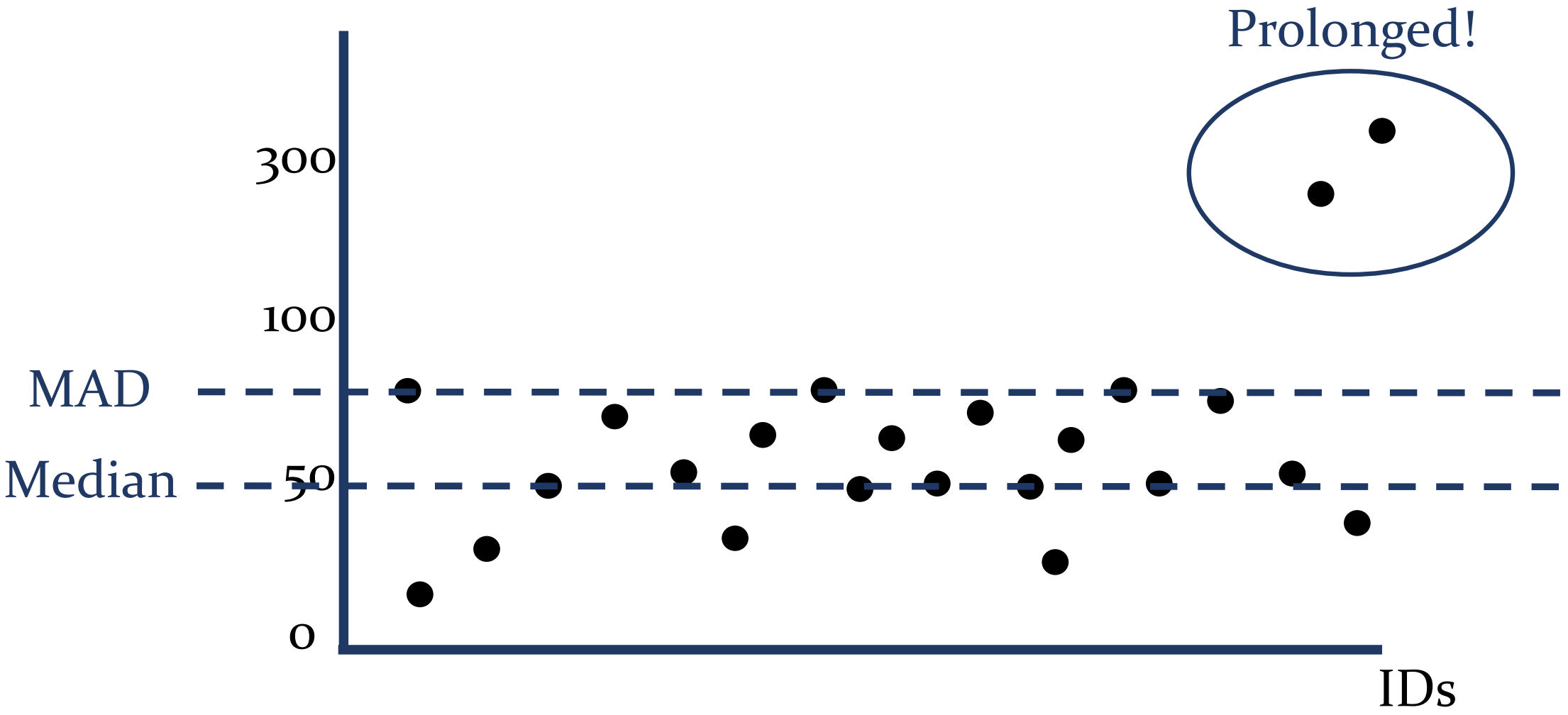
Type 1 – the number of releases that are shipped before an issue is delivered



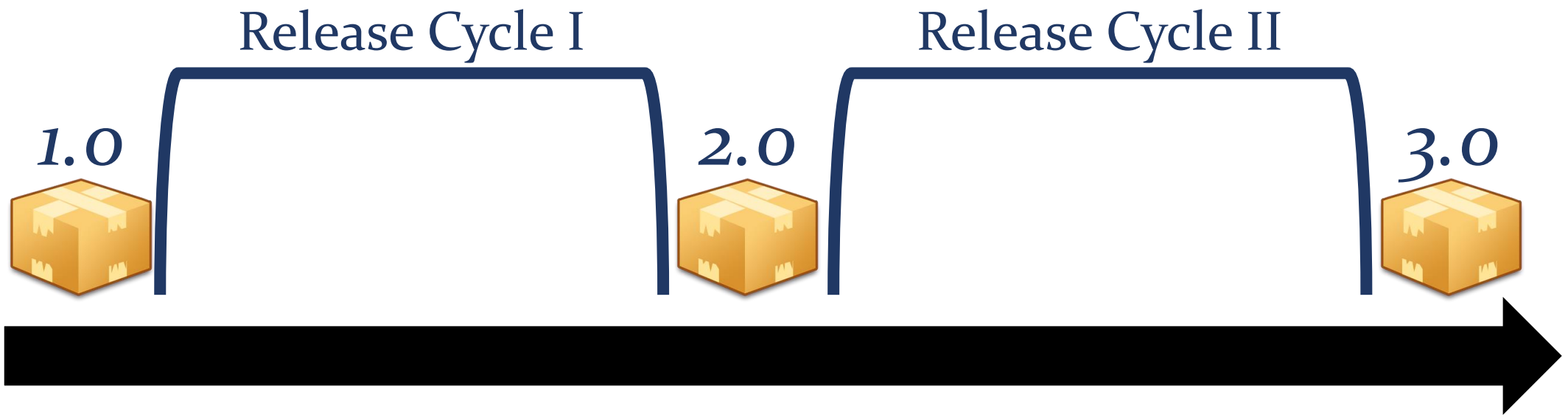
Type 2 – the number of days that are required to ship an addressed issue



Type 3 – the most prolonged delays of a given project



Release cycle: is the time period that the development team takes to ship a new release



Agenda

Study 1

How frequent is delivery delay?

Motivation

There is a lack of empirical research that investigate the frequency of delivery delays

This study is also motivated by investigating the factors that lead to prolonged delays

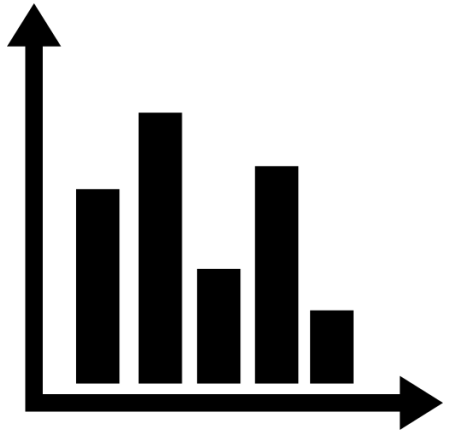
We set out to empirically study 20.995 addressed issues from the Firefox, ArgoUML, and Eclipse projects

Firefox

Eclipse

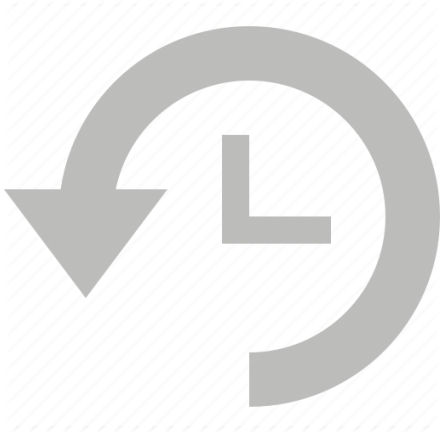
ArgoUML

We perform the following analyses



Frequency

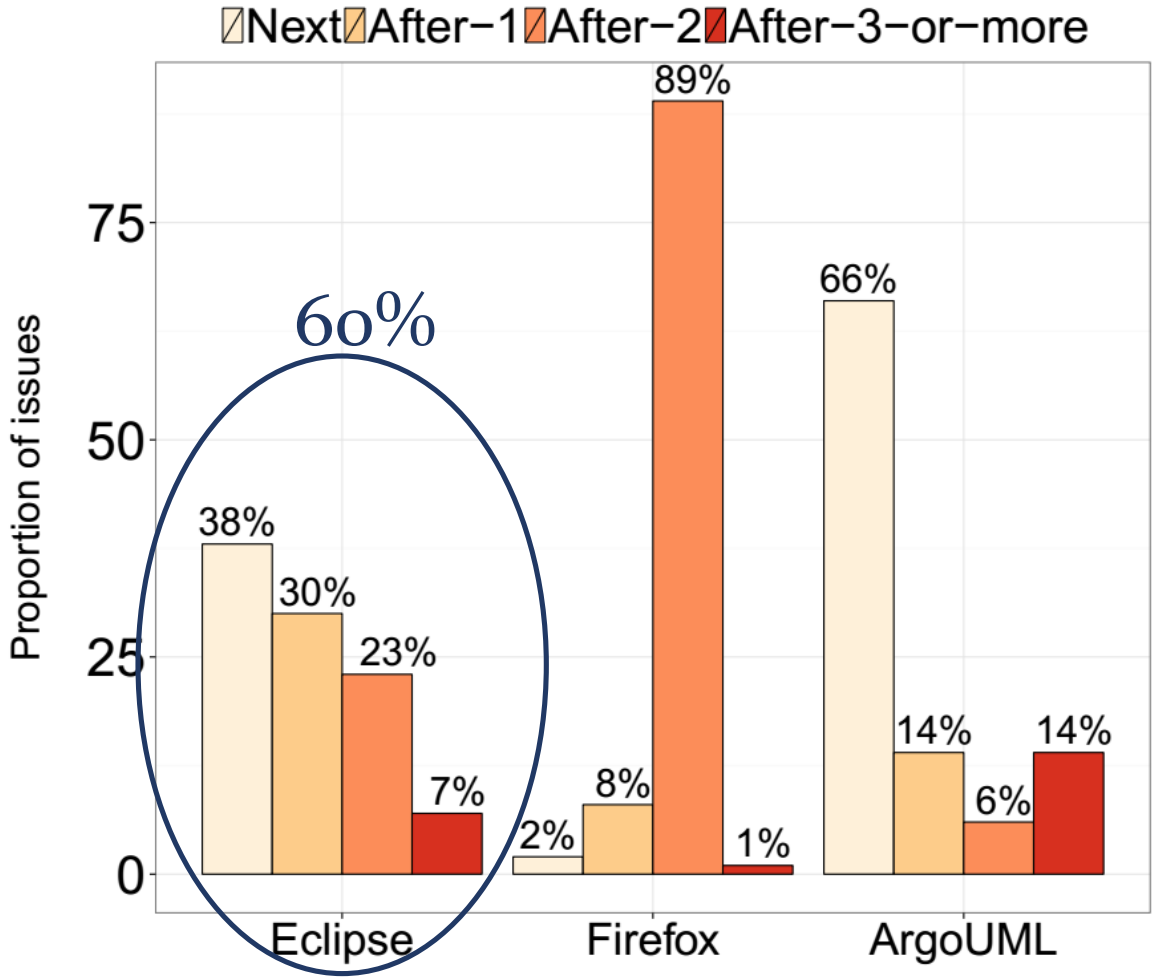
Explore factors



Prolonged delays

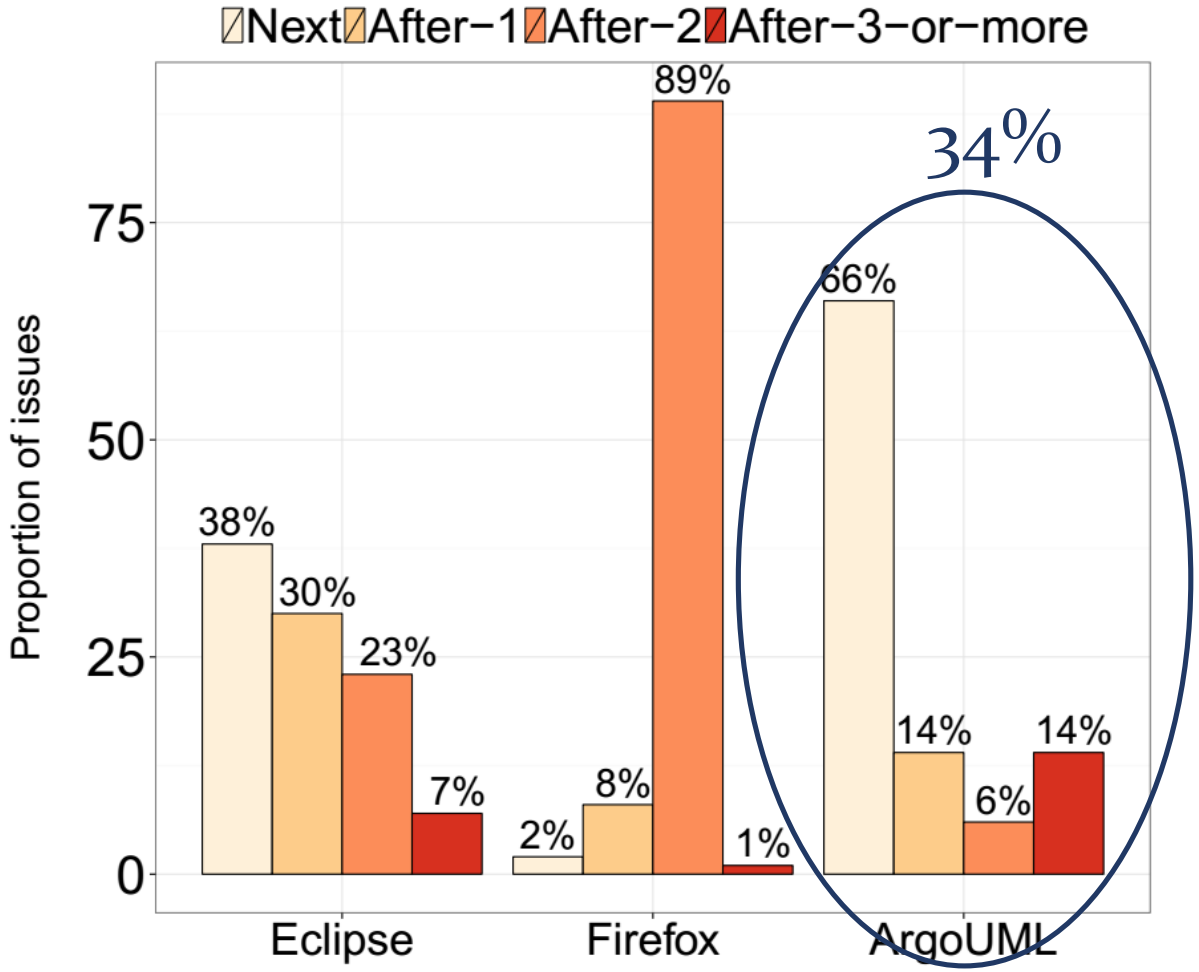
RQ1 – How often are addressed issues prevented from shipping?

34% to 98% of addressed issues had their integration prevented from at least one release



RQ1 – How often are addressed issues prevented from shipping?

34% to 98% of addressed issues had their integration prevented from at least one release



Issues are being addressed well before an upcoming release

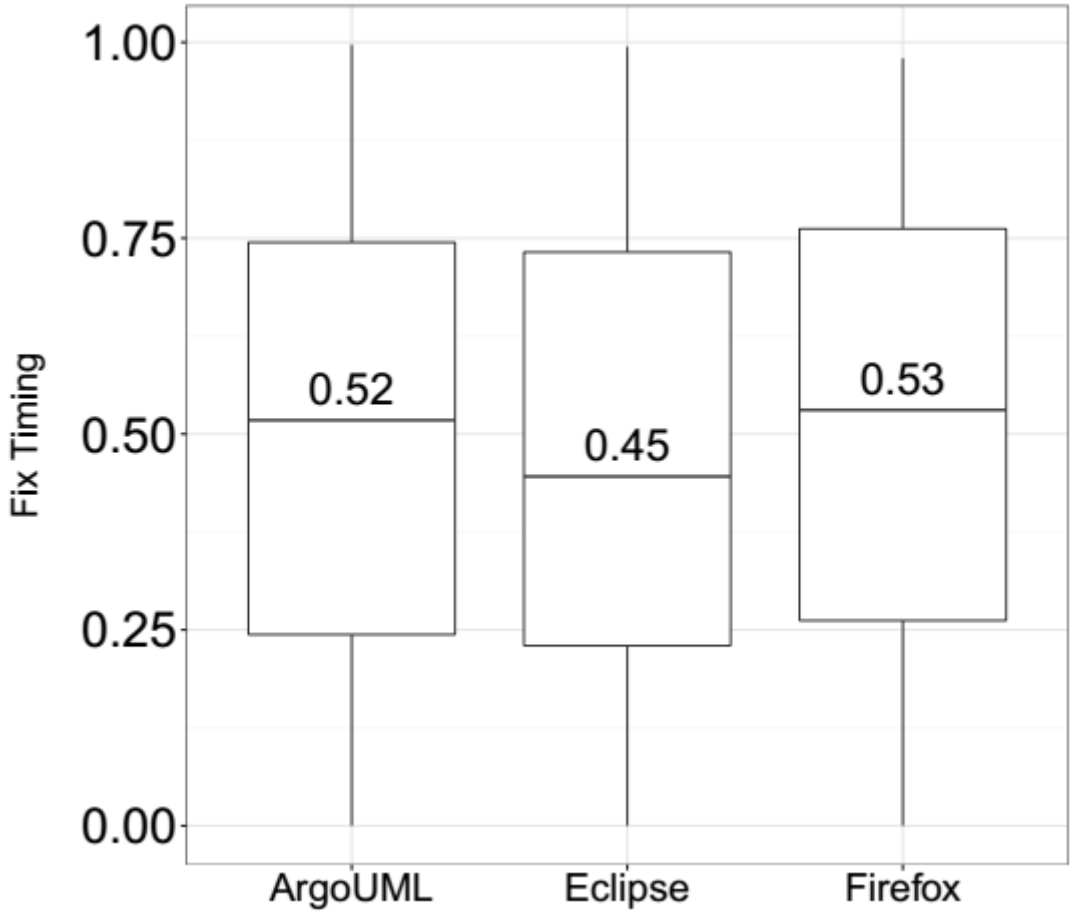
$$\text{Fix Timing} = \frac{\text{\# of days before a release}}{\text{release cycle duration}}$$

Issues are being addressed well before an upcoming release

$$\text{Fix Timing} = \frac{30 \text{ days}}{60 \text{ days}} = 0.5$$

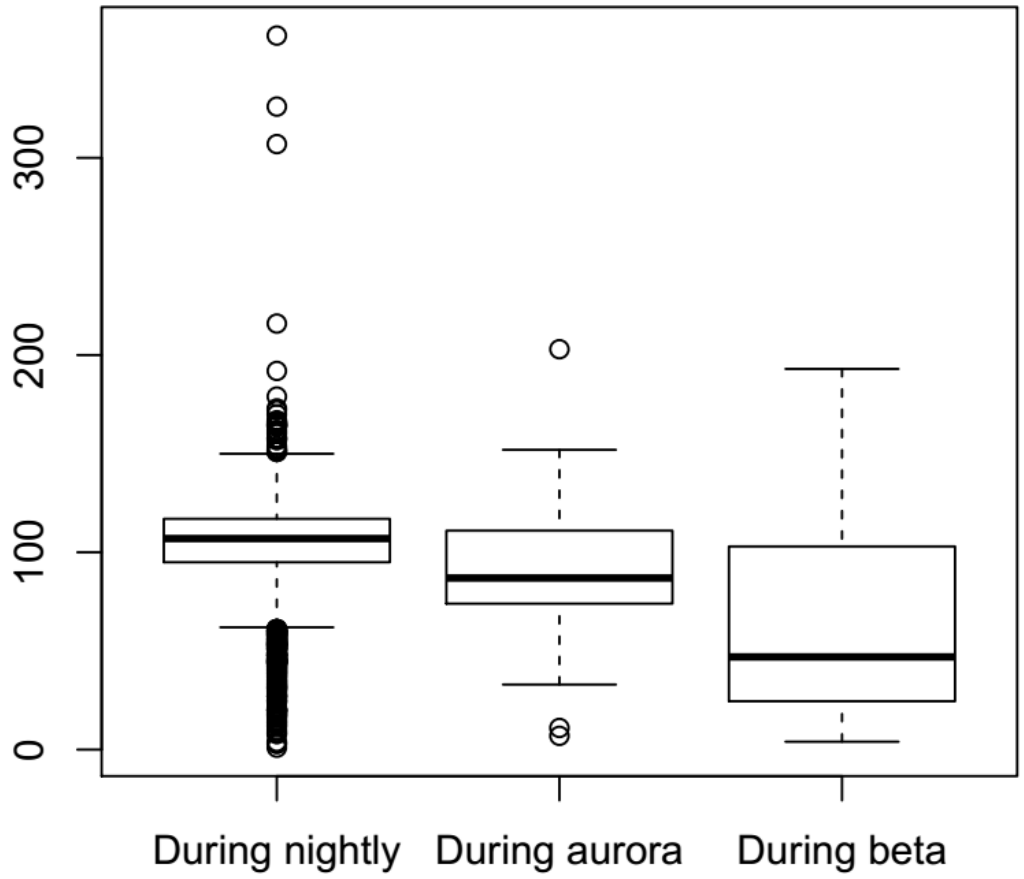
RQ1 – How often are addressed issues prevented from shipping?

Issues are being addressed well before an upcoming release

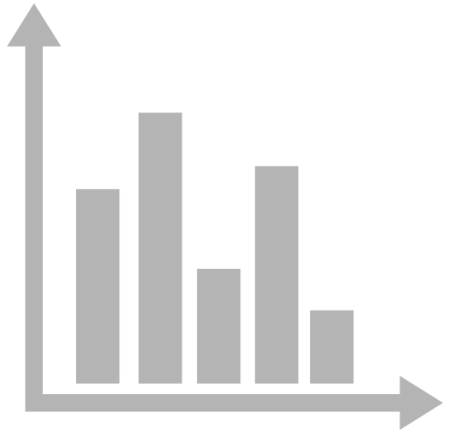


RQ2 – Does the stage of the release cycle impact delivery delay?

Issues that are addressed during more stable stages of a release cycle tends to have shorter delays



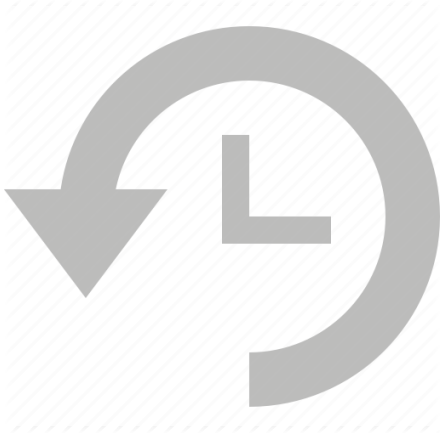
Empirical Study



Frequency

Indeed, delivery delay is frequent

Explore factors

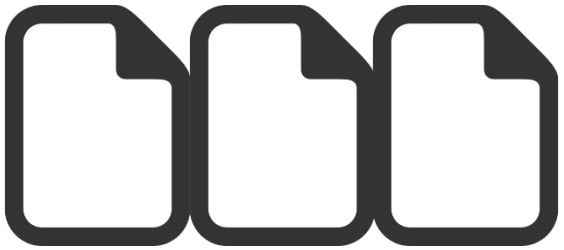


Prolonged delays

RQ3 – How well can we model the delivery delay of addressed issues?

We collect many factors that are computed using *code repositories* and *issue tracking systems*

Number of Impacted Files



Number of comments

I Think this bug is due to…

Experience of the resolvers

vs

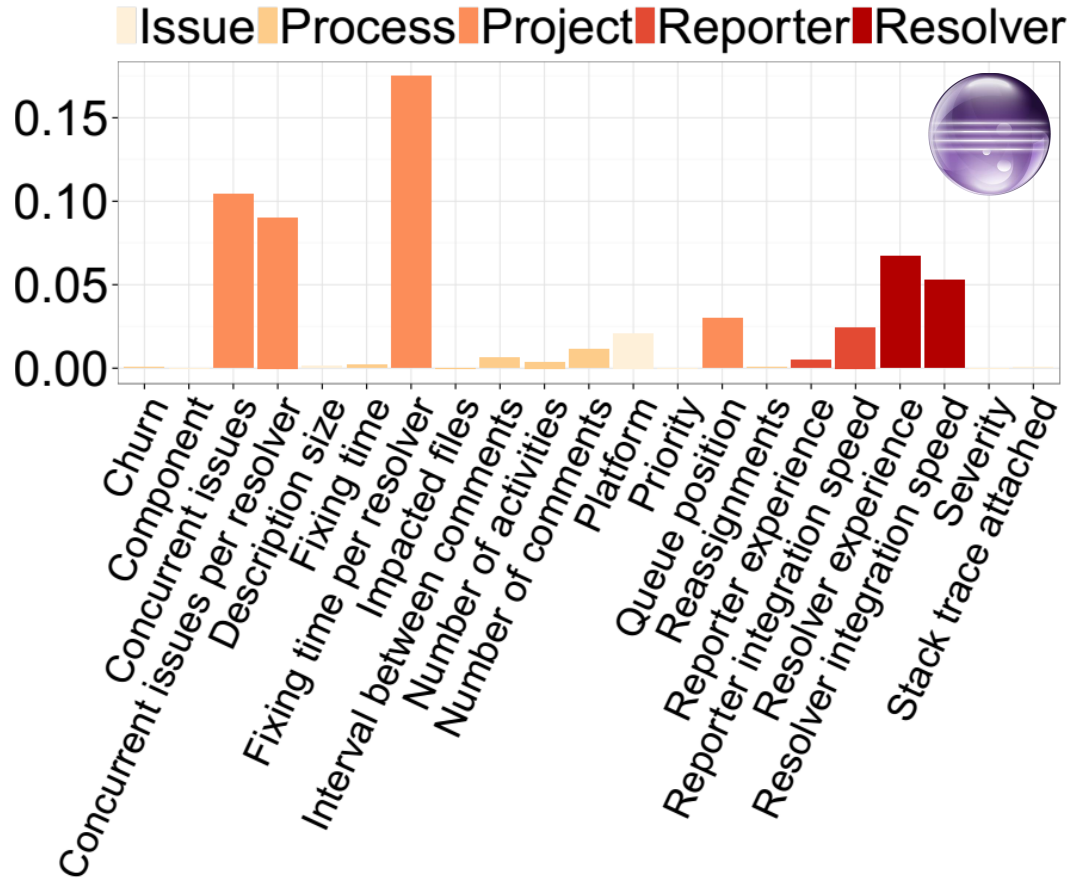
Then, we analyze both delivery delay in terms of number of releases and days

AUCs = 0.62 – 0.96

R^2 = 0.39 – 0.65

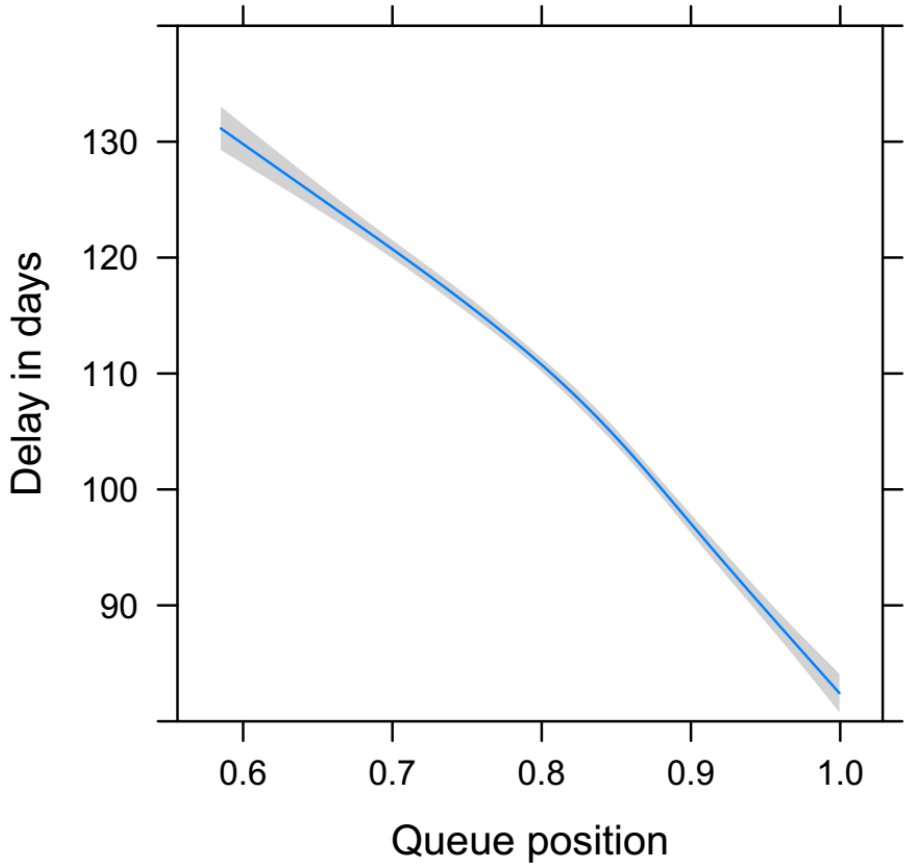
RQ4 – What are the most influential factors for modeling delivery delay?

The time invested on addressing issues and workload factors play an influential role to model delay in terms of releases

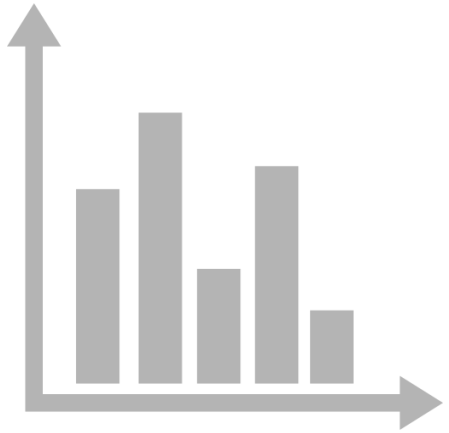


Severity and priority levels have little influence on the delivery delay in terms of releases

The time at which issues are addressed during a release cycle plays a major role in modeling delay in terms of days



Empirical Study

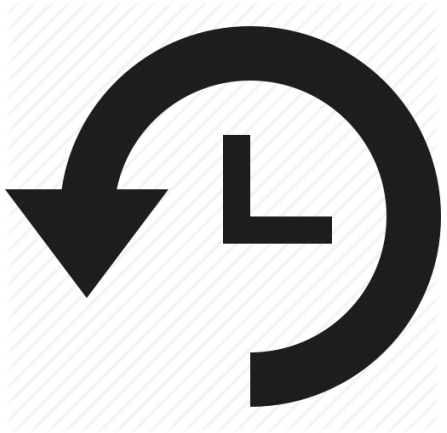


Frequency

Indeed, delivery delay is frequent

Explore factors

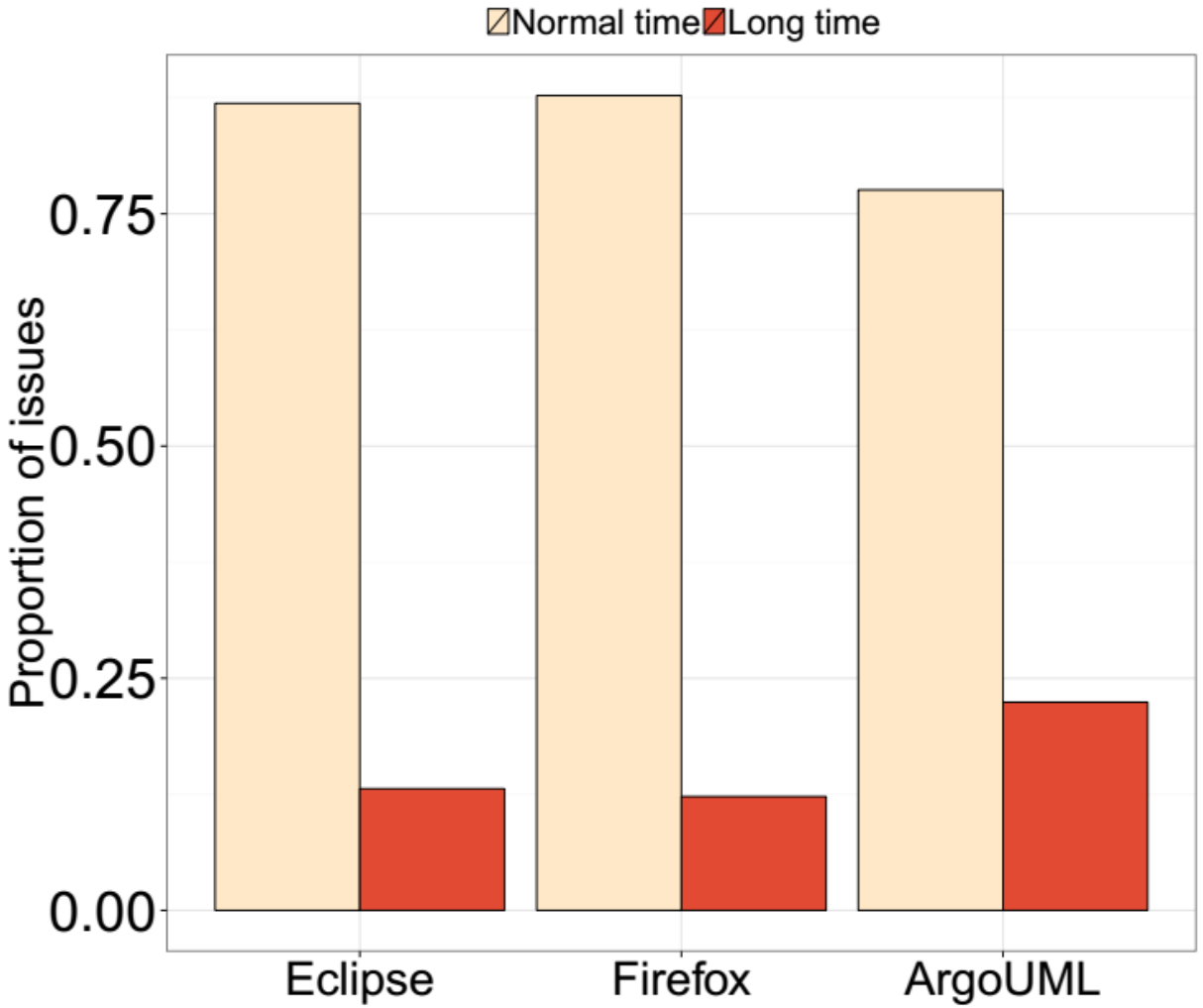
Project factors are the most important



Prolonged delays

RQ5 – How well can we identify issues that will suffer from a prolonged delay?

The proportion of issues that suffered from prolonged delays is under 25% for each dataset



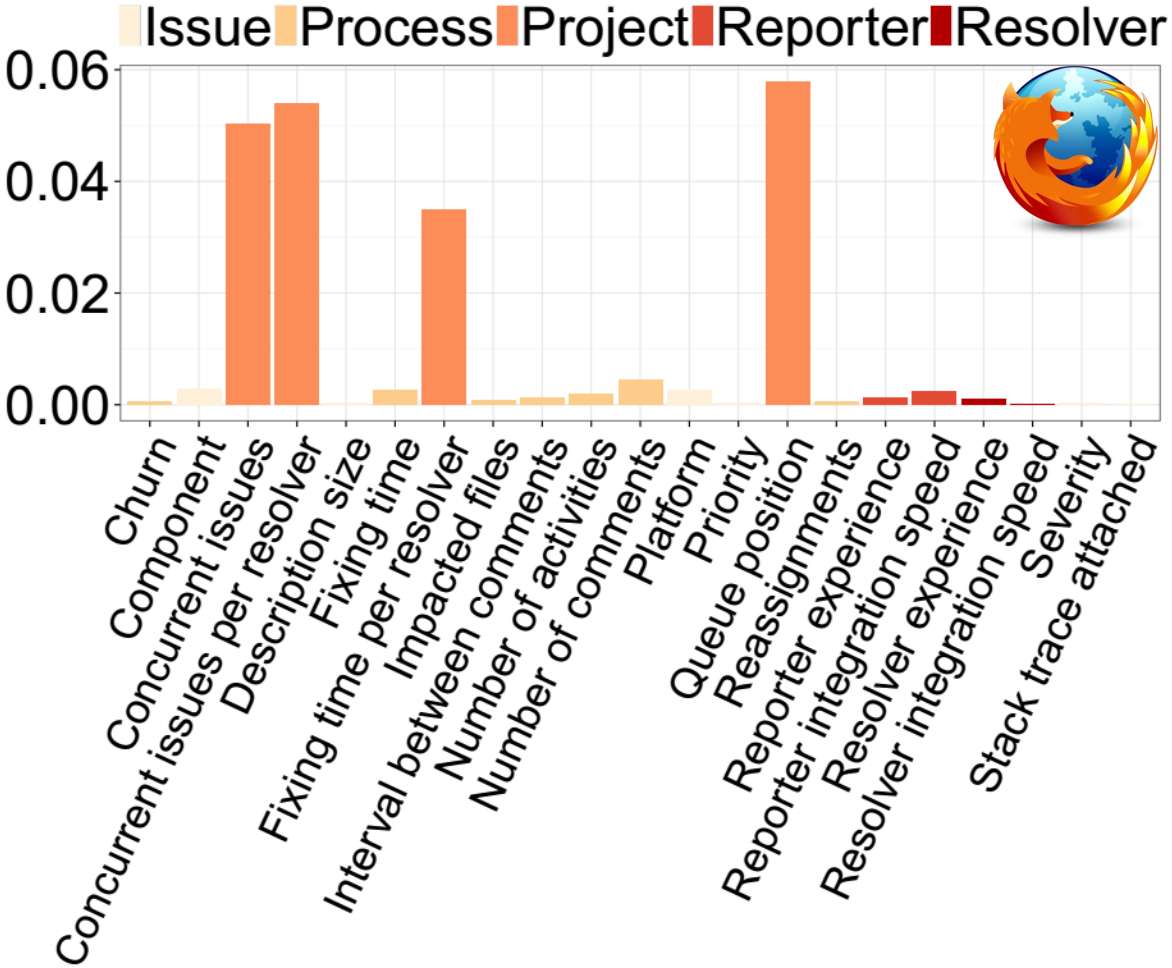
RQ5 – How well can we identify issues that will suffer from a prolonged delay?

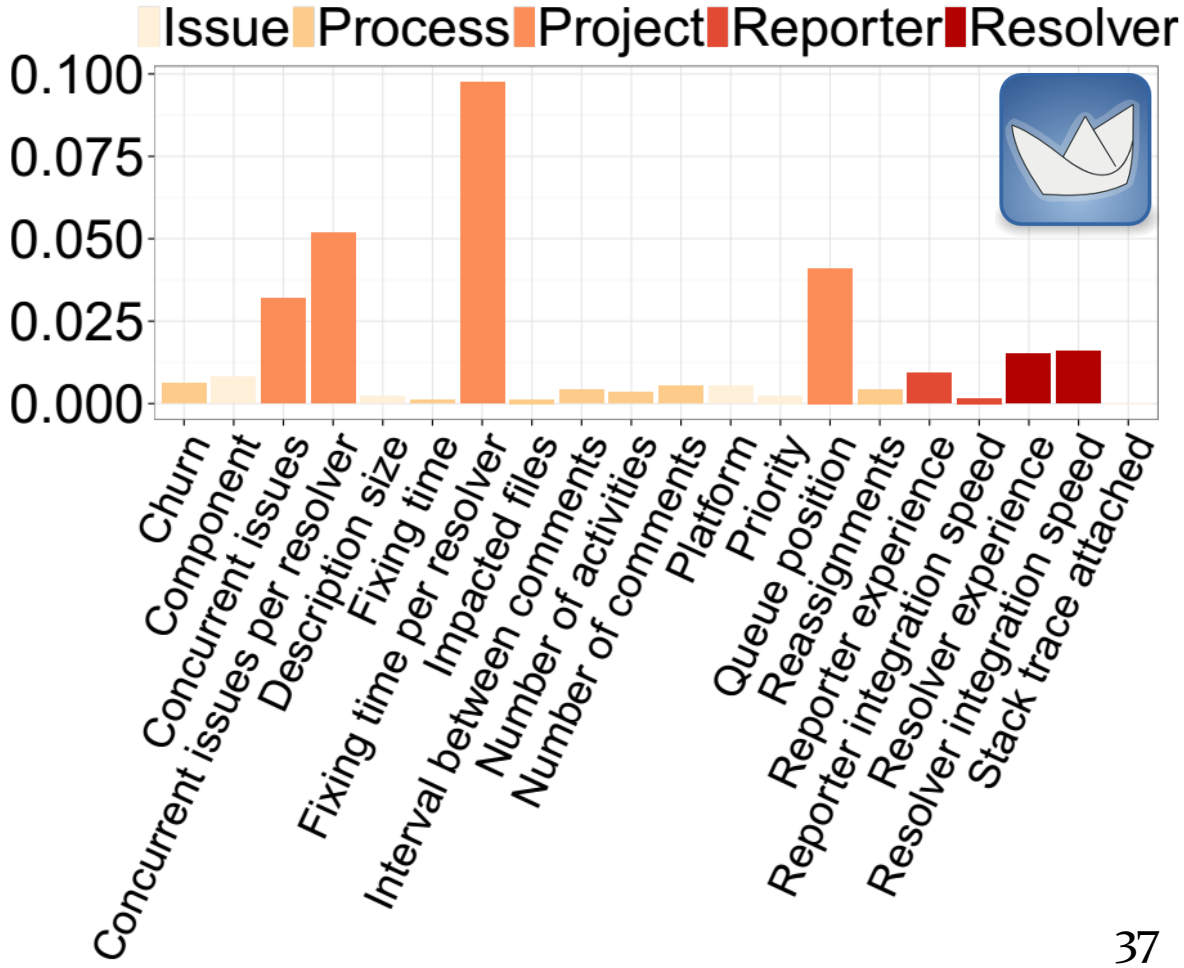
Our statistical models achieve a good fit

AUCs = 0.82 – 0.96

RQ6 – What are the most influential factors to model prolonged delays?

Project factors are most important to model prolonged delays





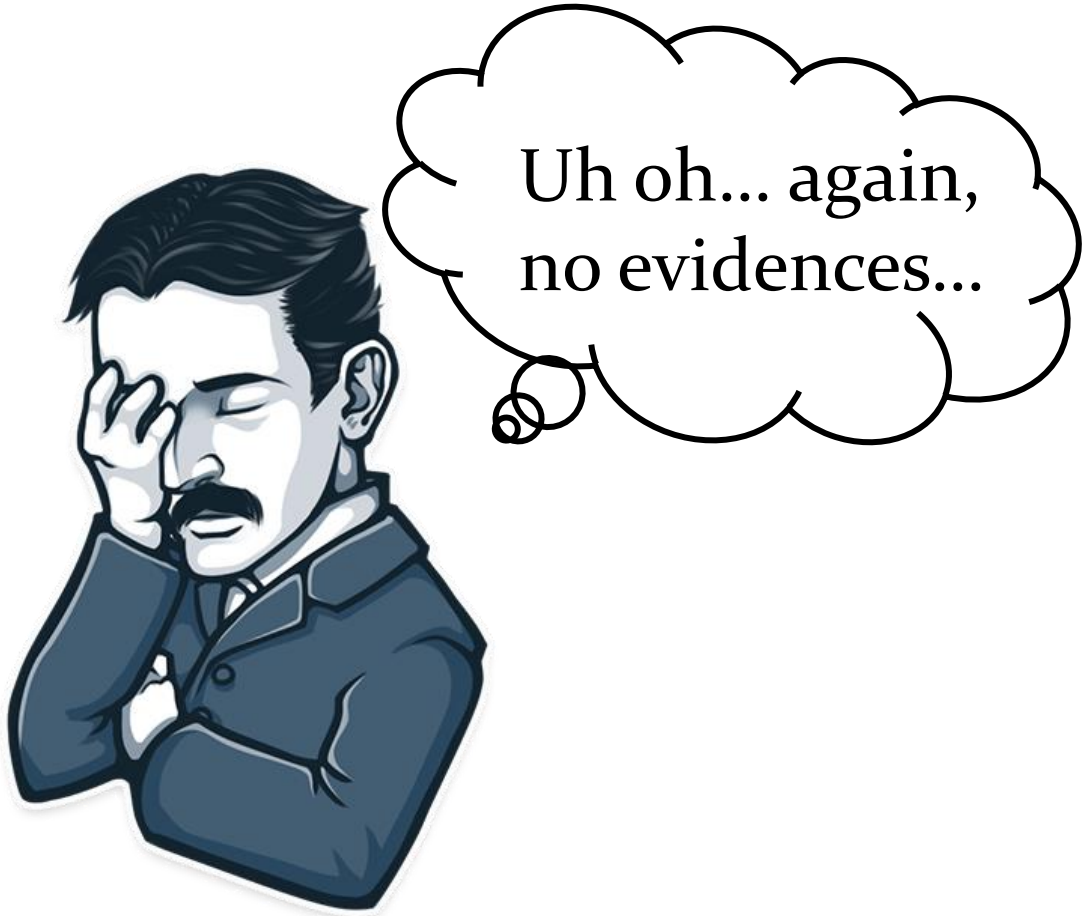
Delivery delays represent an overhead that development teams need to manage

Study 2

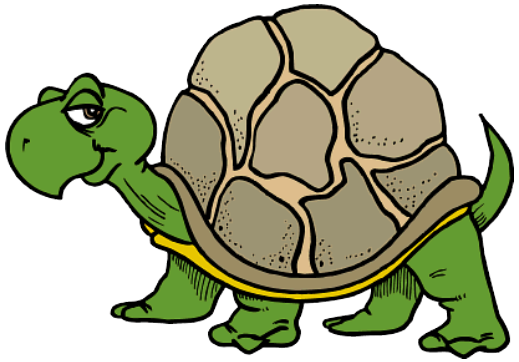
Do rapid releases reduce delivery delays?

The allure of delivering new features more quickly has led many projects to adopt rapid release cycles

However, the speed at which features are delivered using rapid releases still needs to be empirically checked



We set out to empirically compare the delivery delay of traditional and rapid releases of the Firefox project



Traditional

111 releases

34,673 issues

1.0 to 4.0

VS

Rapid

73 releases

37,443 issues

10 to 27

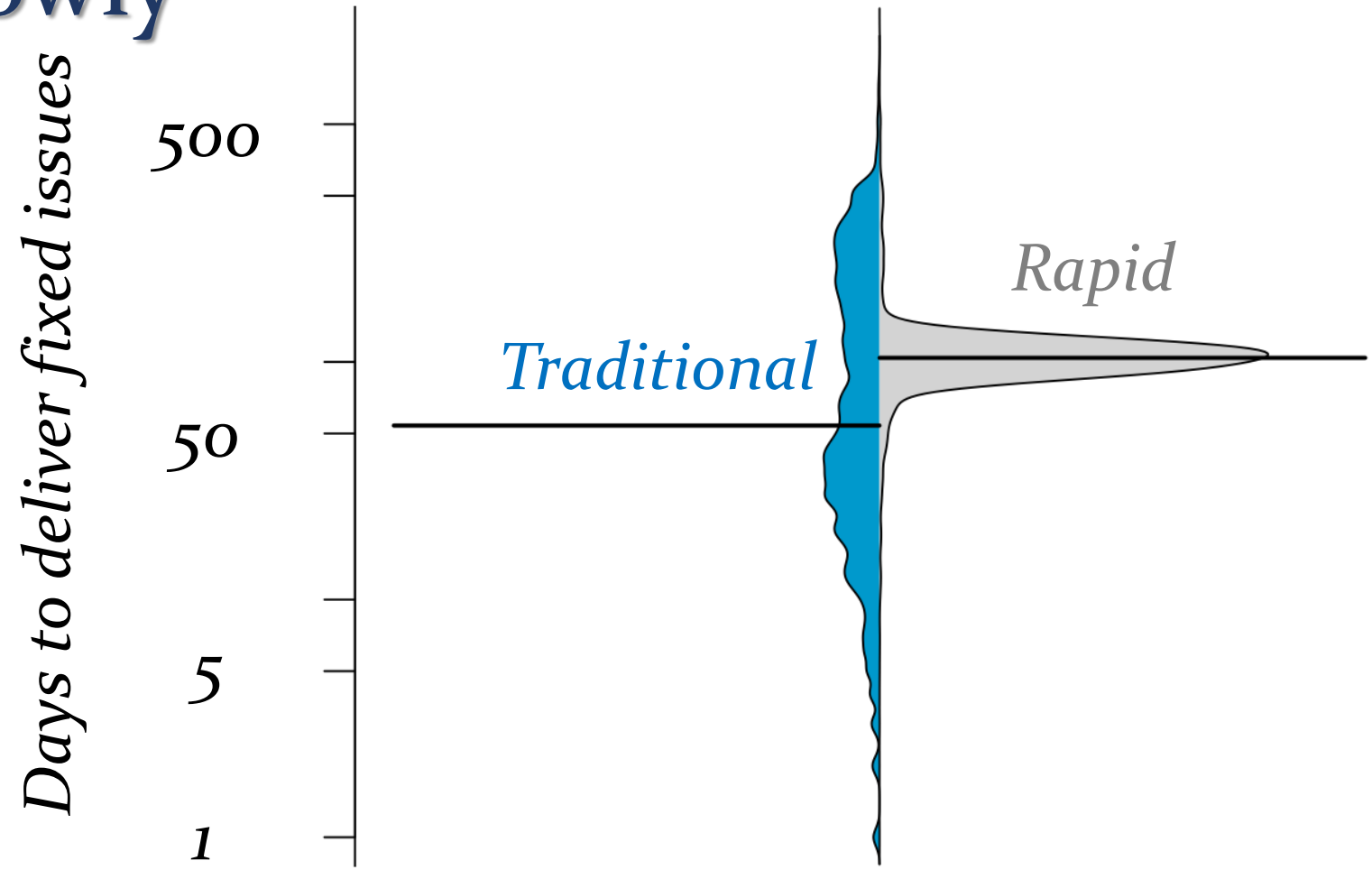
We perform the following analyses

Comparing delivery delays

Major vs minor releases

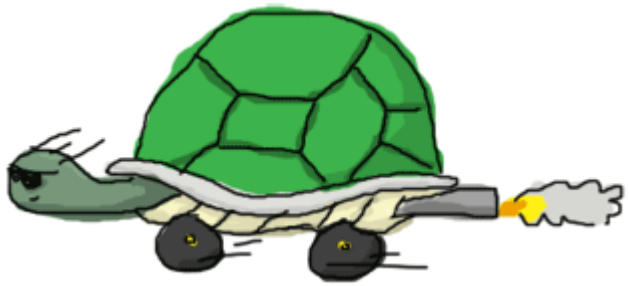
Analyzing important factors

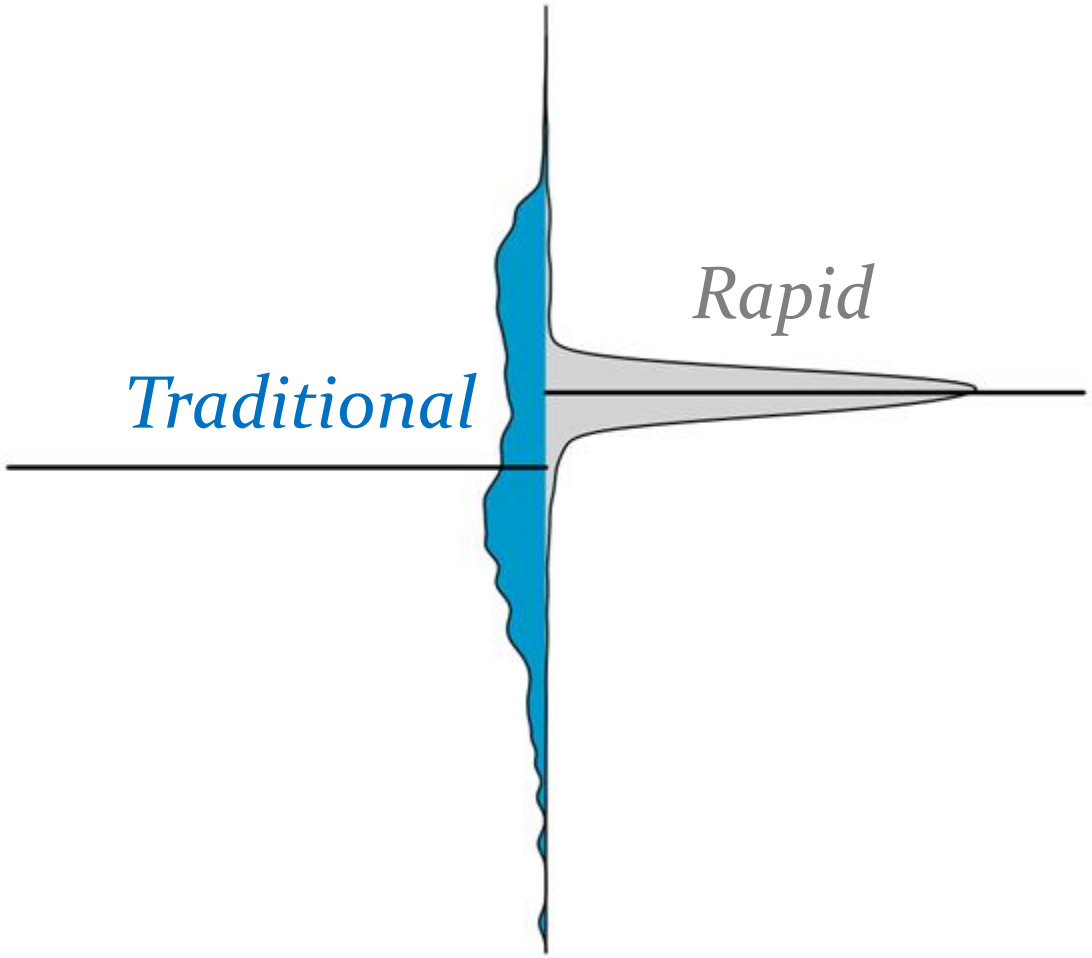
Although issues are fixed more quickly in rapid releases, they are delivered more slowly



Fixed issues are delivered more consistently in rapid releases

I can be faster!





I have consistent gains!

Empirical study

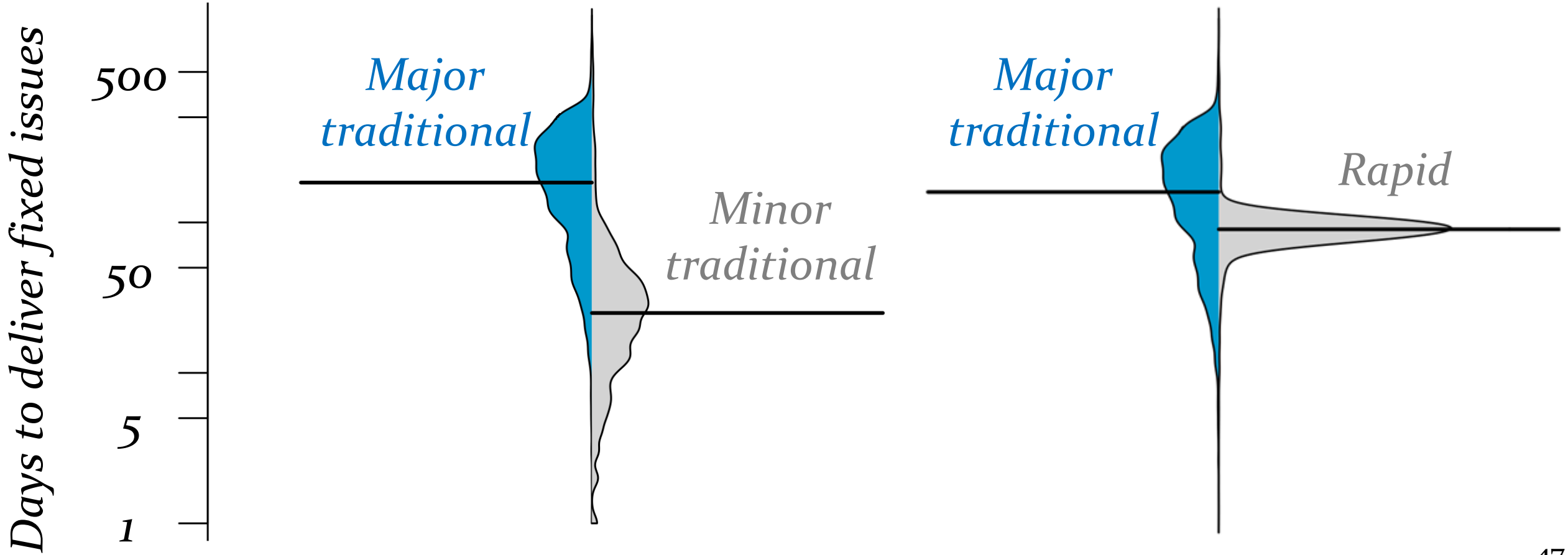
Comparing delivery delays

Rapid releases are not '*that rapid*'

Major vs minor releases

Analyzing important factors

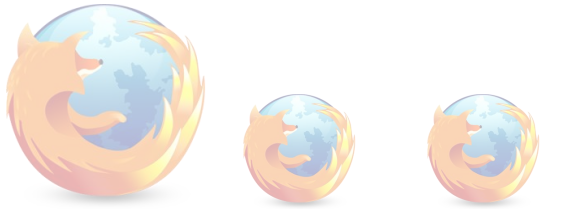
Minor releases: a key reason as to why traditional releases can deliver fixed issues more quickly



Empirical study

Comparing delivery delays

Rapid releases are not '*that rapid*'



Major vs minor releases

Minor releases quicken the delivery time

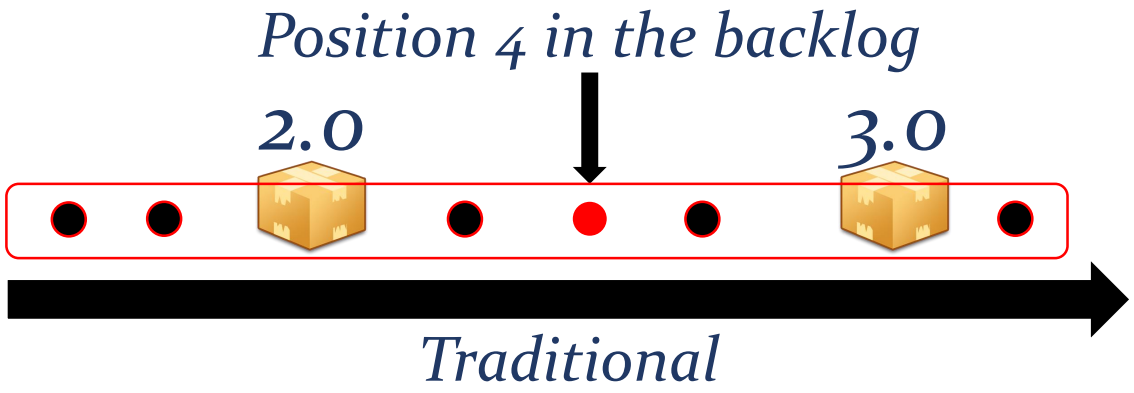
Analyzing important factors

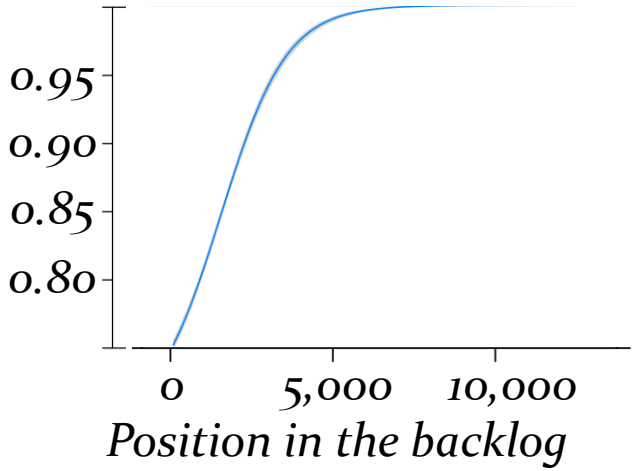
RQ3 – Does the release strategy have an impact on delivery delay factors?

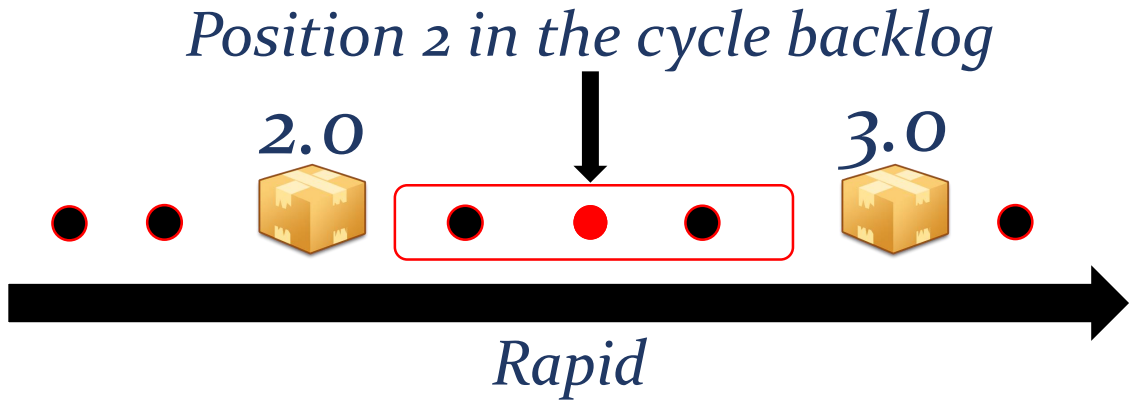
As in Study 1, we collect several factors that we suspect to share a relationship with delivery delay

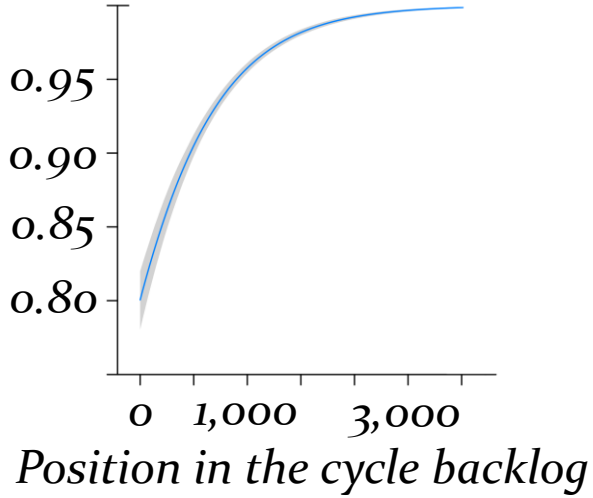
RQ3 – Does the release strategy have an impact on delivery delay factors?

Prioritization is different among release strategies

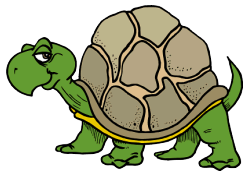


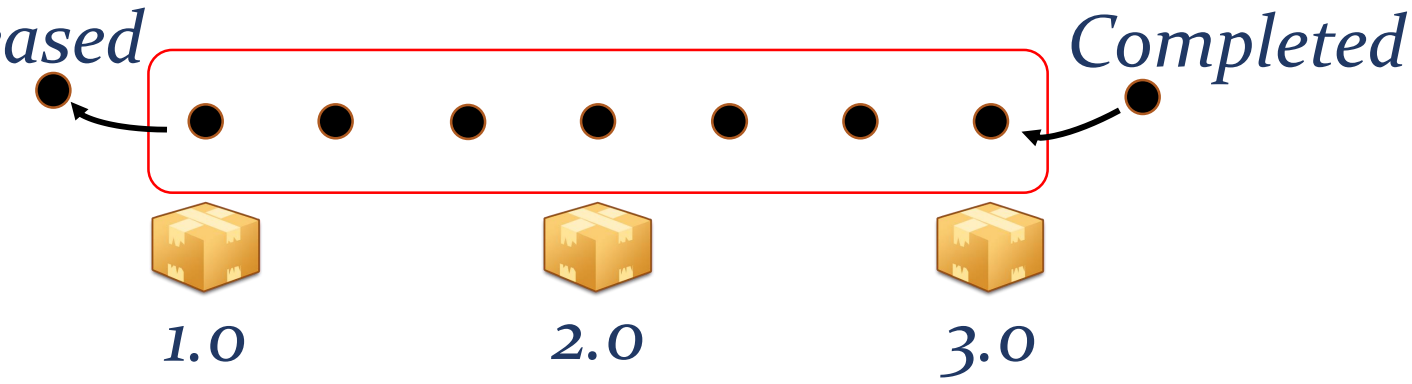




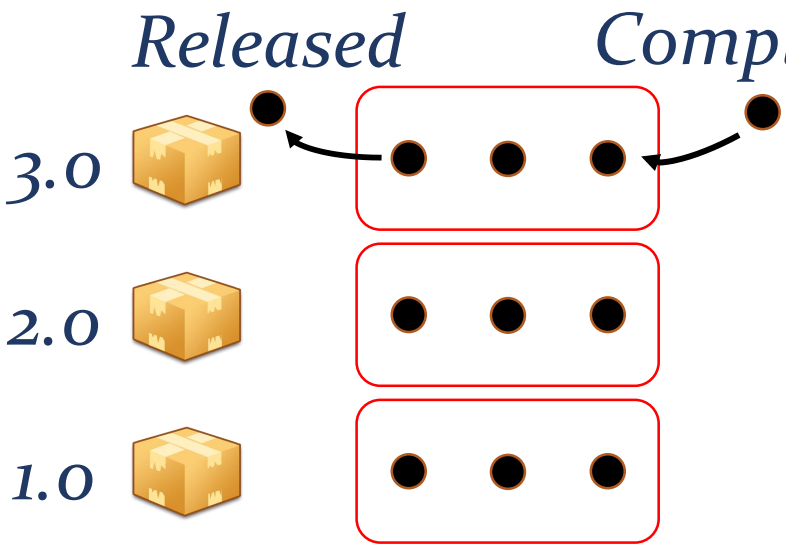


Traditional releases behave as a queue while rapid releases behave as a stack





VS



Empirical study

Comparing delivery delays

Rapid releases are not '*that rapid*'

Major vs minor releases

Minor releases quicken the delivery time

Analyzing important factors

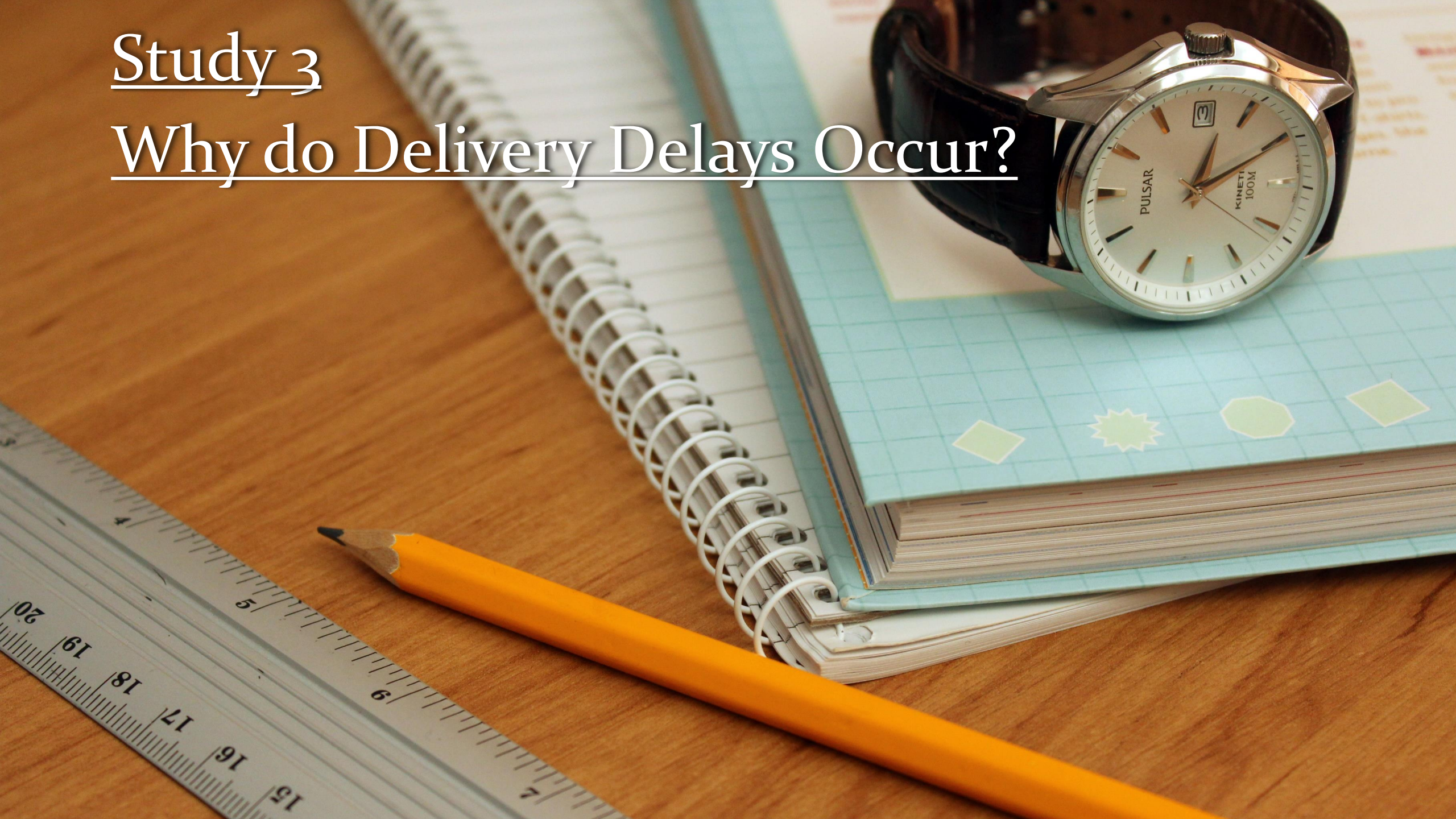
Different prioritization

Conclusions

A rapid release strategy is not a silver bullet to reduce delivery delay

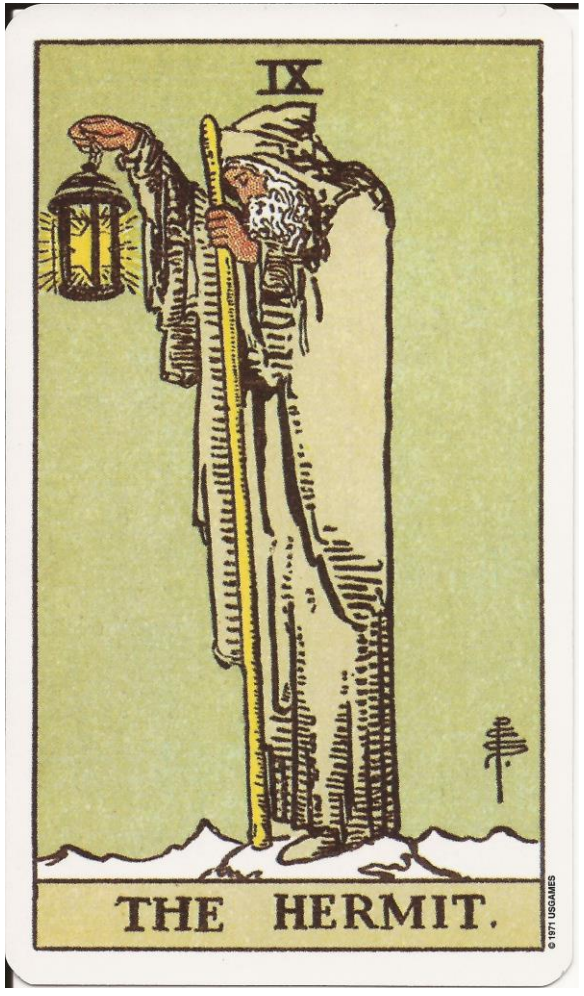
Study 3

Why do Delivery Delays Occur?



Motivation

Reach deeper knowledge that was gained through quantitative analysis



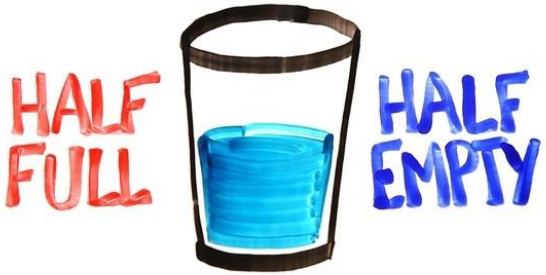
Verify to which extent our quantitative findings can be confirmed by our participants

We survey 37 participants from the Firefox, ArgoUML, and Eclipse projects

We performs an extensive Open Coding analysis that generated 175 unique codes

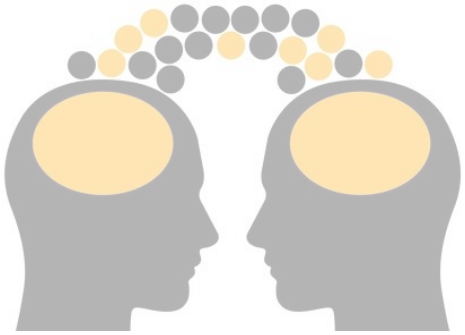
Empirical study

We perform the following analyses



General perceptions

Impressions about rapid releases



Do our findings make sense?

RQ1 – What are developers' perceptions as to why delivery delay occur?

The most recurrent themes are '*development activities*', '*decision making*', '*risk*', '*frustration*', and '*team collaboration*'

Development activities

Additional testing is one of the main perceived reasons as to why delivery delay occurs

Lack of "*actual user testing beyond what QA can provide*"

F17

Another main perceived reason is the workload

"As the delayed completed issues stack up, they are harder to integrate (the codebase is constantly changing, merge issues might emerge)"

E30

Frustration

The majority source of frustration is users' *'expectation'*

"as a user, it's like when you are waiting your suitcase in the airport to come out on the belt. You know it has to be there, but you keep waiting"

F07

"it's like a gift for Christmas, but the day of Christmas is postponed"

F14

Team collaboration

An overhead may be introduced when collaboration between teams is needed

"sometimes, issues that require cross-team cooperation may be delayed when the issue is differently prioritized by each team."

F23

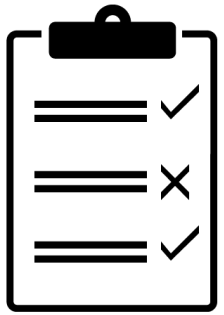
"generally happens when marketing wants to make a splash"

F21

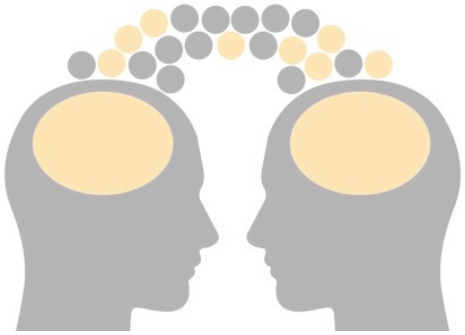
Empirical study

General perceptions

Decision making, frustration, risk, external teams, dev. activities



Impressions about rapid releases



Do our findings make sense?

The main themes about the impact of adopting rapid cycles can be grouped into: '*management*', '*delivery*', and '*development*'

Management

The most recurrent theme regarding management is *'flexibility'*

"it is more flexible, since if an important issue pushed back a less important change and it misses the release cycle, it's not a huge deal with rapid releases"

F01

Management

Another recurrent theme regarding management was '*risk mitigation*'

"able to identify issues sooner. It is easier to identify issues when you have only deployed 3 new commits than 100"

F07

Delivery

By far, the most recurrent theme was the allure of '*faster delivery*' of issues

"*increasing speed of getting new features to users*"

F05

"*getting new features to users sooner*"

F07

Delivery

However, not all participants agree that rapid cycles deliver fixed issues faster

"rapid releases reduce the time to deliver issues to end users in some cases, and lengthen them in others"

F22

"Low priority issues (new features) take less time to be delivered, whereas high priority ones (important bugs) take more time"

F24

Delivery

'Faster user feedback' is another important theme regarding delivery

"you don't find yourself fixing a bug that you introduced two years ago which the field only discovered on the release"

E29

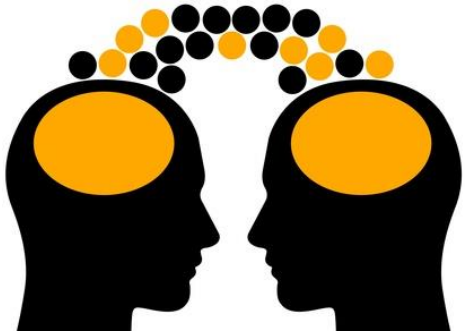
Empirical study

General perceptions

Decision making, frustration, risk, external teams, dev. activities

Impressions about rapid releases

Management, delivery, development activities

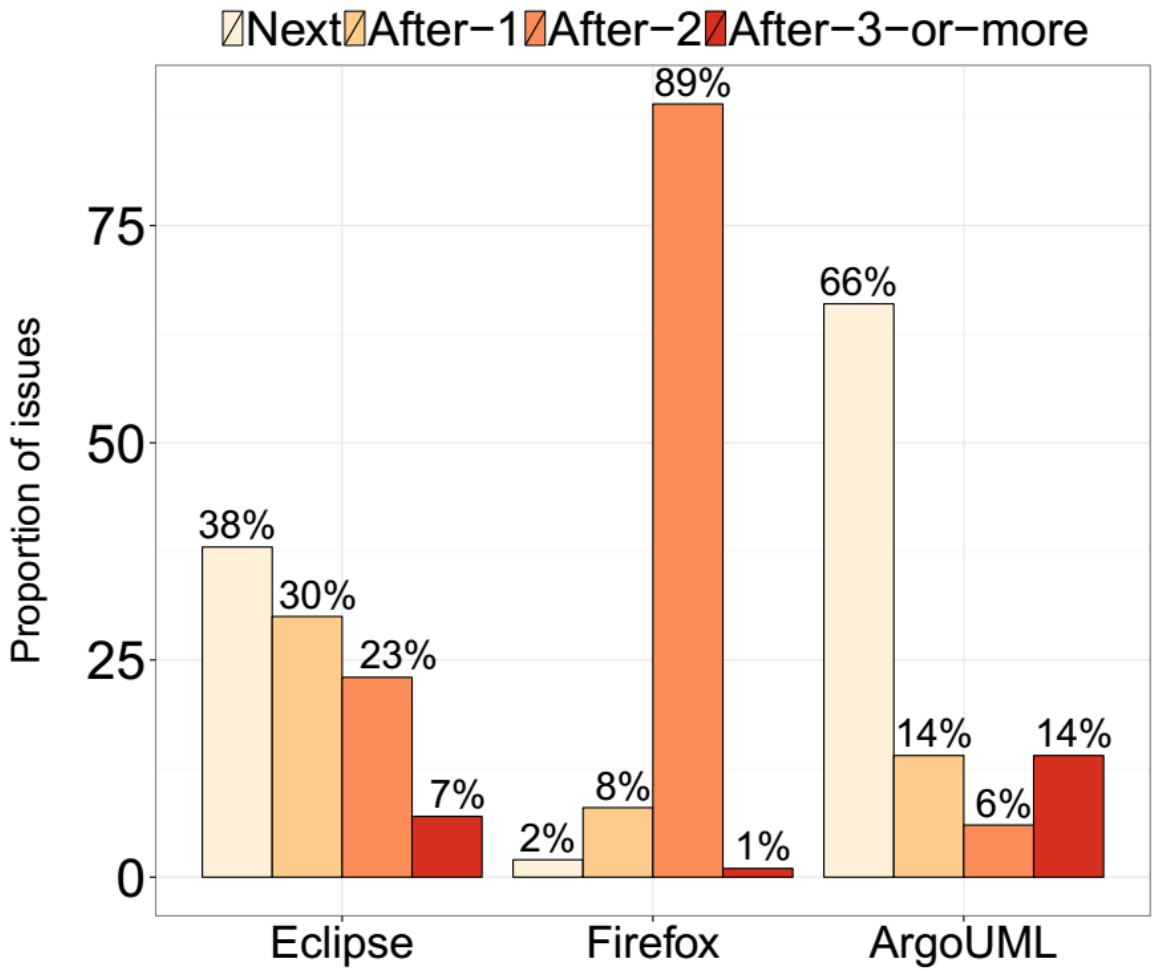


Do our findings make sense?

RQ3 – To what extent do developers agree with our quantitative findings?

We request feedback about our two studied themes: *reasons for delivery delay* and *impact of rapid releases*

We present our data that was obtained in Study 1



Reasons for delivery delay

The most recurrent theme cited by our participants was '*workload*'

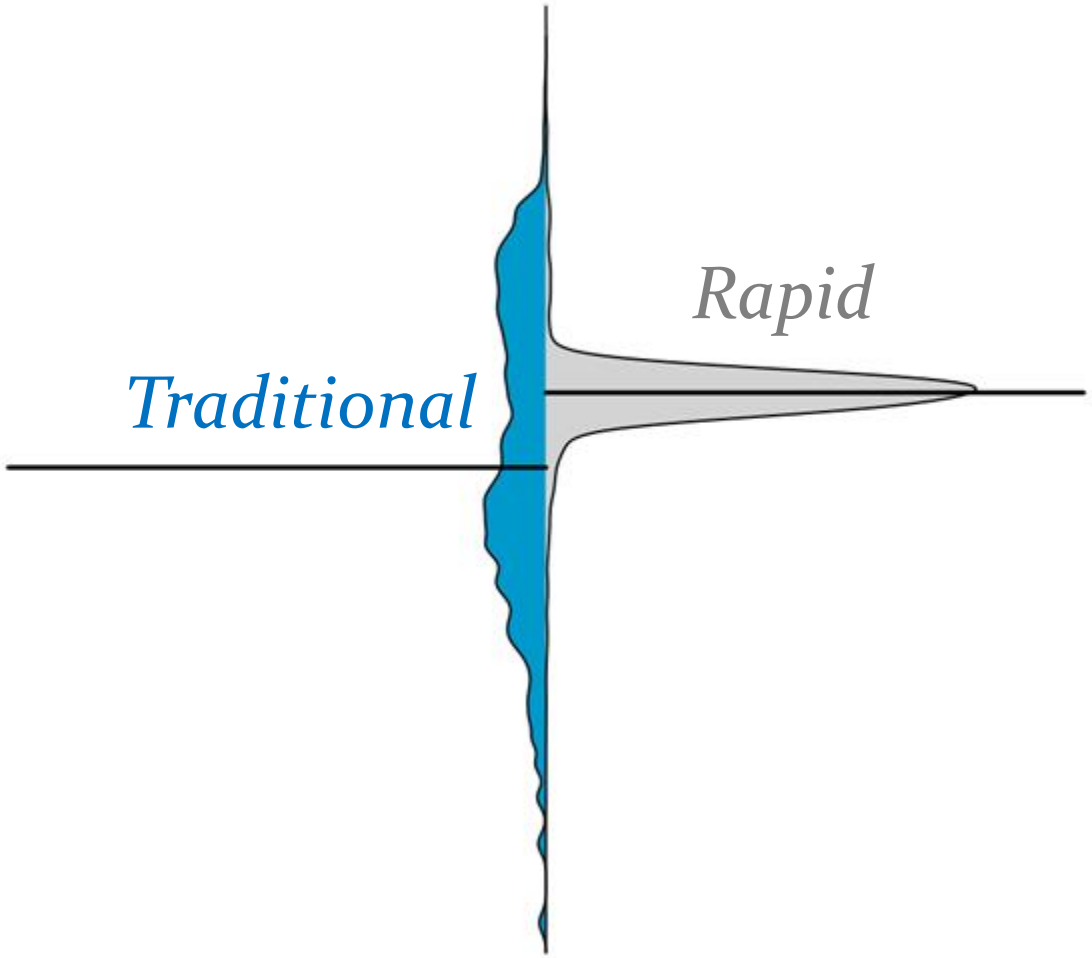
"committers are too busy"

E27

"there might be delays in reviews when issues are completed"

E26

We present our data that was obtained in Study 2



Five participants explicitly disagree with our obtained data

"I'm not surprised that there are things in that bucket. [Instead,] *I'm surprised that there are many of them."*

F06

Six participants explicitly agree with our obtained data

"since missing a release cycle isn't a big deal, more features are kept from being released until they're properly polished instead of being rushed at the end of a long release cycle"

F15

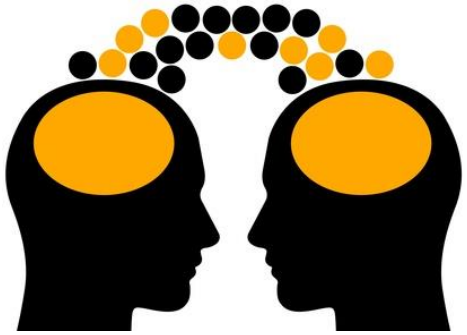
Empirical study

General perceptions

Decision making, frustration, risk, external teams, dev. activities

Impressions about rapid releases

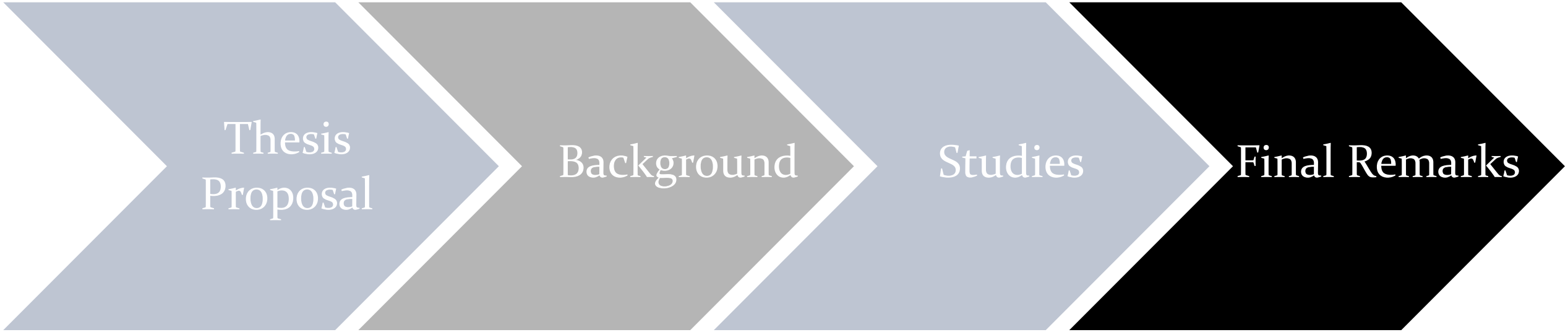
Management, delivery, development activities



Do our findings make sense?

Our data helped us achieving deeper insights

Agenda



Limitations

Generalizability is hindered by the number of investigated systems

Lack of open datasets from projects that have transitioned from a traditional release strategy to a rapid release strategy

Study limitations

Small sample of participants in our qualitative analysis

Related Work

Jiang 2013

Will my patch make it?

Morakot 2015

Risk of postponing an upcoming release schedule

Conclusions

Conclusions

Even though issues are addressed, they may still suffer from delays that development teams need to manage

Rapid releases are not silver bullets to reduce delivery delays

Conclusions

Workload, limited tests, and lack of code reviews are main reasons as to why delivery delays may occur

Conclusions

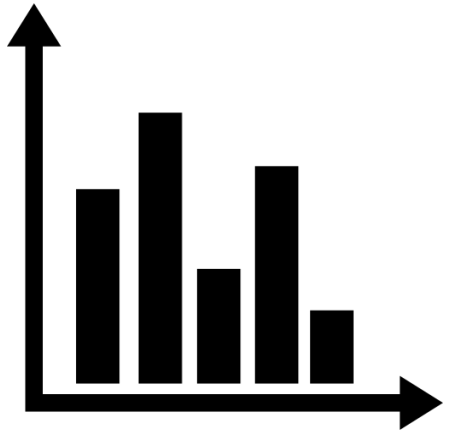
Our qualitative analysis helped us to deepen our insights related to our quantitative analysis

Future Work

Many possibilities for future research: replication, software quality, tooling, and prediction

Summary

Study 1

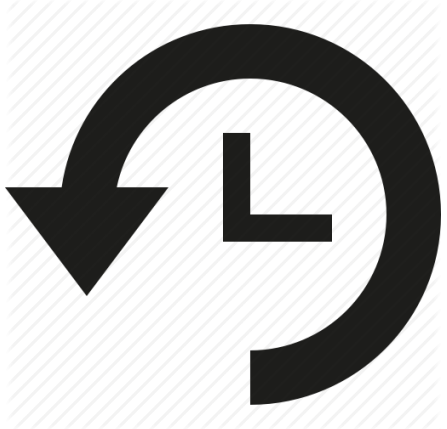


Frequency

Indeed, delivery delay is frequent

Explore factors

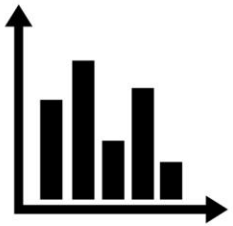
Project factors are the most important



Prolonged delays

Again… project factors are the most important

Study 1

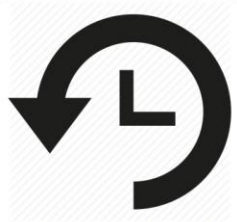


Frequency

Indeed, delivery delay is frequent

Explore factors

Project factors are the most important



Prolonged delays

Again… project factors are the most important

Study 2

Comparing delivery delays

Rapid releases are not '*that rapid*'

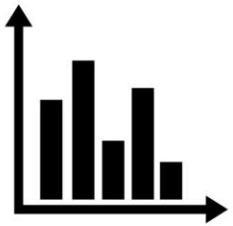
Major vs minor releases

Minor releases quicken the delivery time

Analyzing important factors

Different prioritization

Study 1

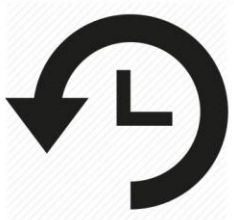


Frequency

Indeed, delivery delay is frequent

Explore factors

Project factors are the most important



Prolonged delays

Again… project factors are the most important

Study 2

Comparing delivery delays

Rapid releases are not '*that rapid*'

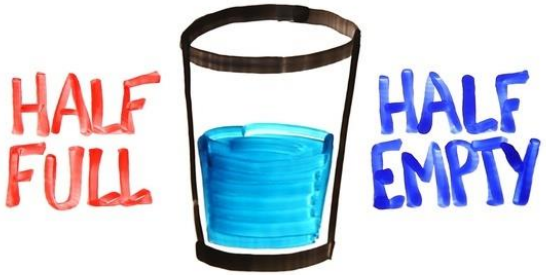
Major vs minor releases

Minor releases quicken the delivery time

Analyzing important factors

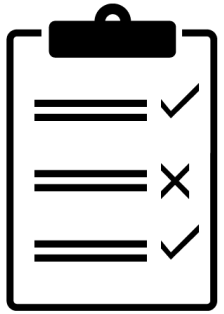
Different prioritization

Study 3



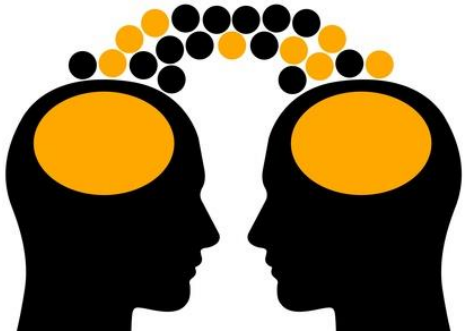
General perceptions

Decision making, frustration, risk, external teams, dev. activities



Impressions about rapid releases

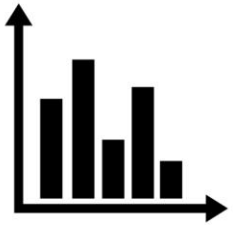
Management, delivery, development activities



Do our findings make sense?

Our data helped us achieving deeper insights

Study 1

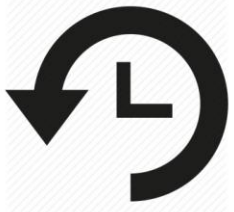


Frequency

Indeed, delivery delay is frequent

Explore factors

Project factors are the most important



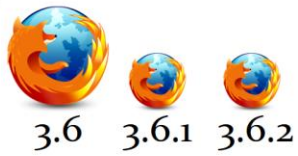
Prolonged delays

Again… project factors are the most important

Study 2

Comparing delivery delays

Rapid releases are not '*that rapid*'



Major vs minor releases

Minor releases quicken the delivery time

Analyzing important factors

Different prioritization

Study 3

General perceptions

Decision making, frustration, risk, external teams, dev. activities

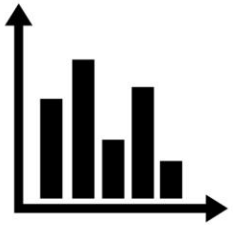
Impressions about rapid releases

Management, delivery, development activities

Do our findings make sense?

Our data helped us achieving deeper insights

Study 1

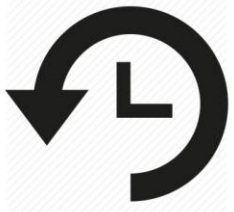


Frequency

Indeed, delivery delay is frequent

Explore factors

Project factors are the most important



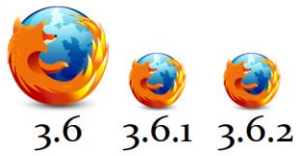
Prolonged delays

Again… project factors are the most important

Study 2

Comparing delivery delays

Rapid releases are not '*that rapid*'



Major vs minor releases

Minor releases quicken the delivery time

Analyzing important factors

Different prioritization

Study 3

General perceptions

Decision making, frustration, risk, external teams, dev. activities

Impressions about rapid releases

Management, delivery, development activities

Do our findings make sense?

Our data helped us achieving deeper insights

Thank you!