Continuous Delivery / Continuous Deployment

How to automate your Deliveries

Bernhard Keprt

24.02.2014

WILLHABEN.AT®

Bernhard Keprt
Software Developer

- Teamleader Software Development Job at willhaben.at

- Likes
    - Agile Development
    - Test Driven Development
    - Software Architecture
- Hates
    - Bugs
    - Refill the coffee machine

willhaben.at

- Austria’s largest public portal
  - > 765 Mil. page impressions per month
  - > 22,8 Mil. visits per month
  - > 4,6 Mil. unique clients per month
  - > 33,7% reach
  - > 2,3 Mil. online adverts

  Source: owea.at, January 2014

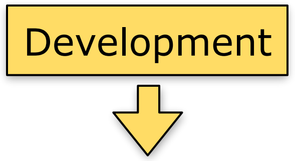
- Great Place To Work

Common Feature Lifecycle

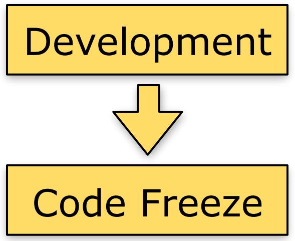
Common Feature Lifecycle

Development

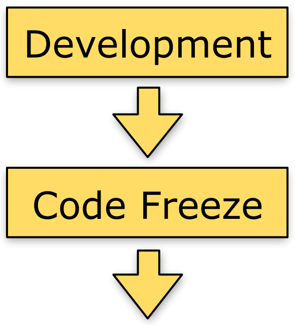
Common Feature Lifecycle



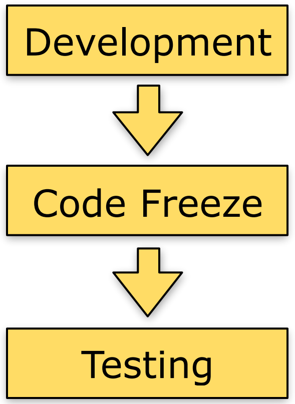
Common Feature Lifecycle



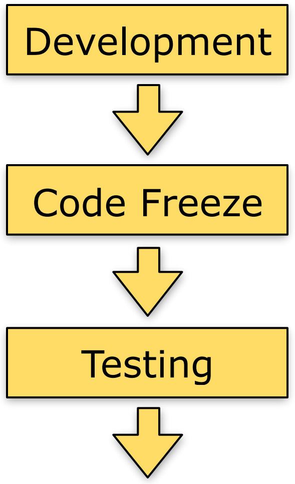
Common Feature Lifecycle



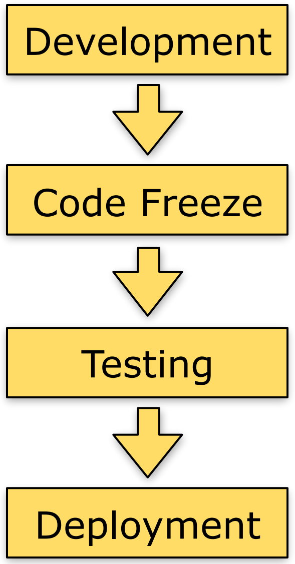
Common Feature Lifecycle



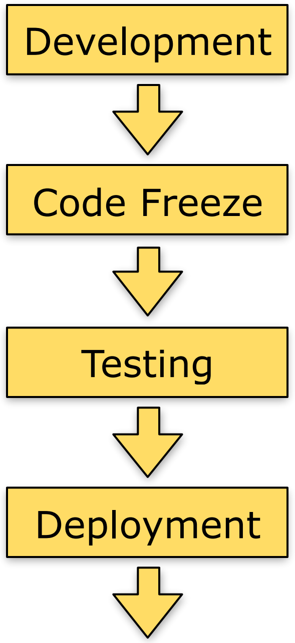
Common Feature Lifecycle



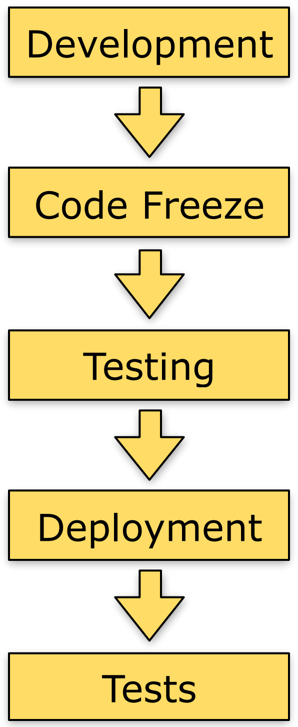
Common Feature Lifecycle



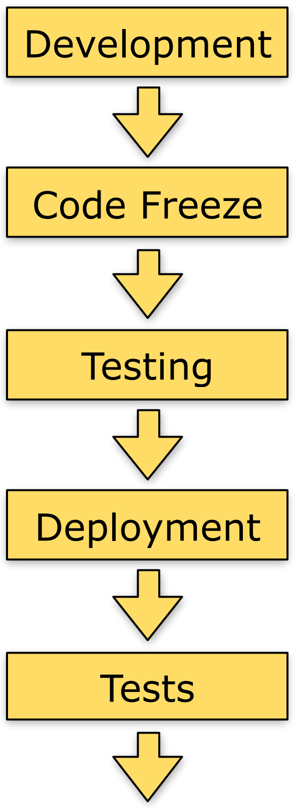
Common Feature Lifecycle



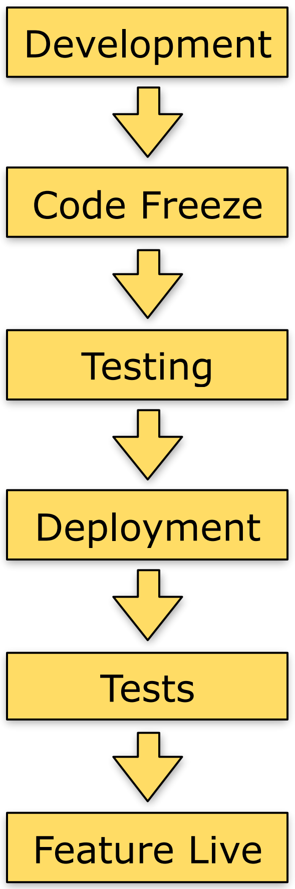
Common Feature Lifecycle



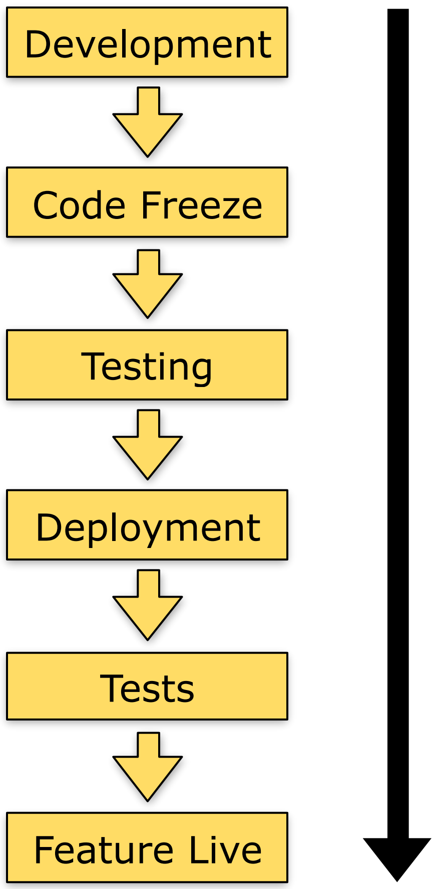
Common Feature Lifecycle



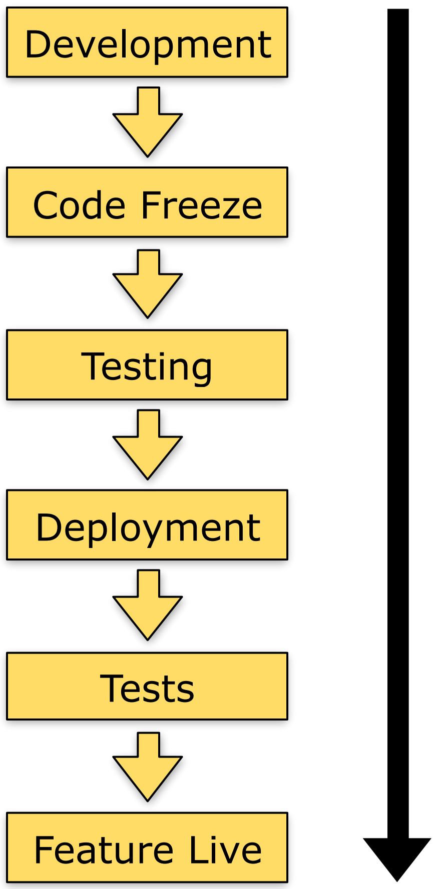
Common Feature Lifecycle

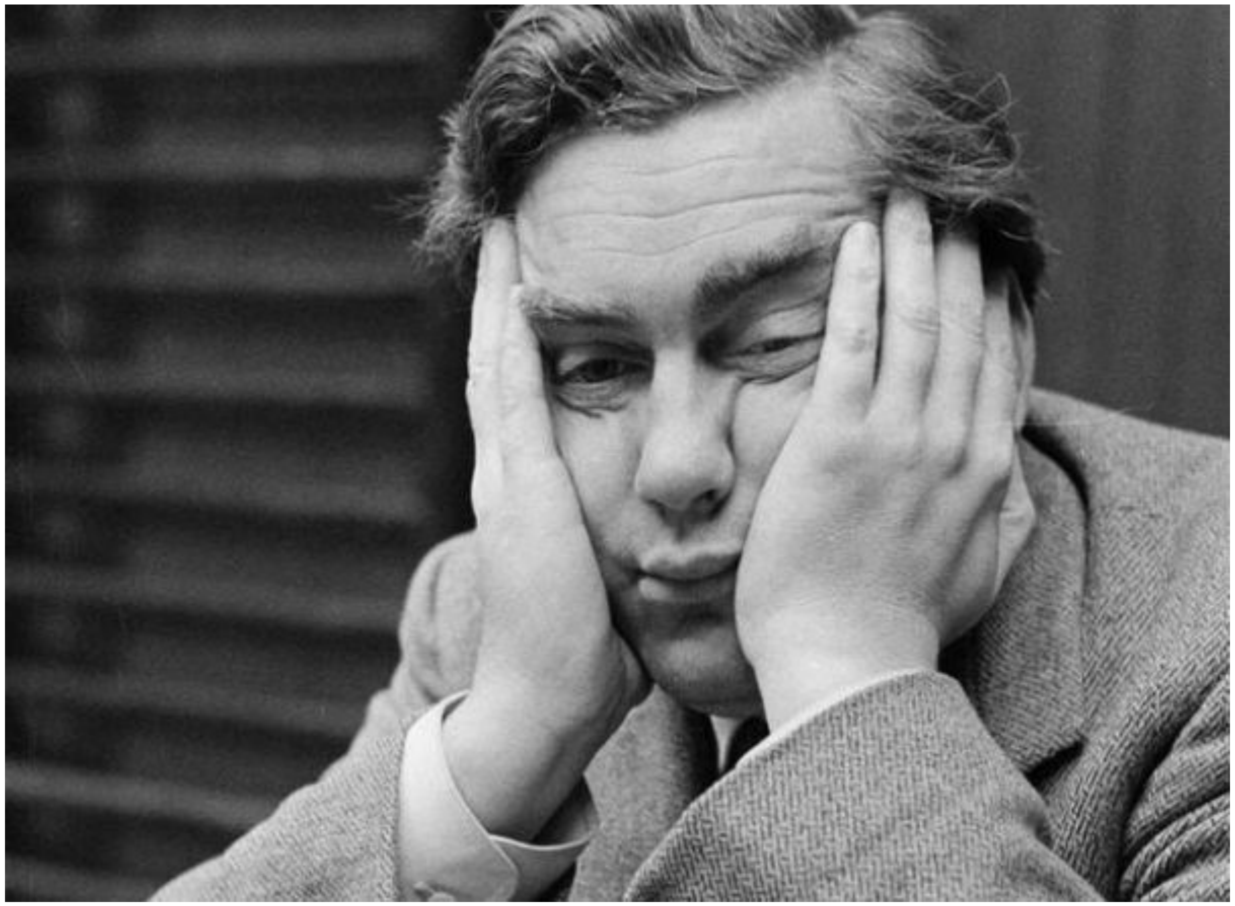


Common Feature Lifecycle



Common Feature Lifecycle





Common Feature Lifecycle

Common Feature Lifecycle

PM

Common Feature Lifecycle

PM

Dev Meetings

Common Feature Lifecycle

PM

Dev Meetings

Common Feature Lifecycle

PM

Dev Meetings

Dev

Common Feature Lifecycle

PM

Dev Meetings

Dev

QA

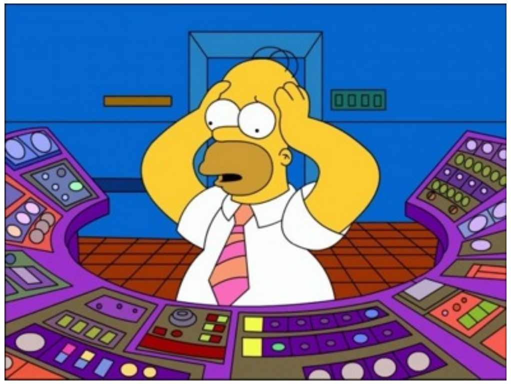
Common Feature Lifecycle

PM

Dev Meetings

Dev

QA



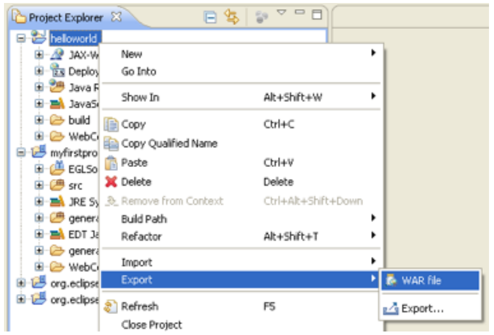
OPs

Continuous Delivery - DONTs

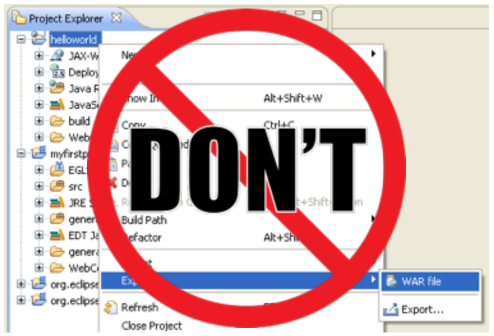
Continuous Delivery - DONTs

Continuous Delivery - DONTs

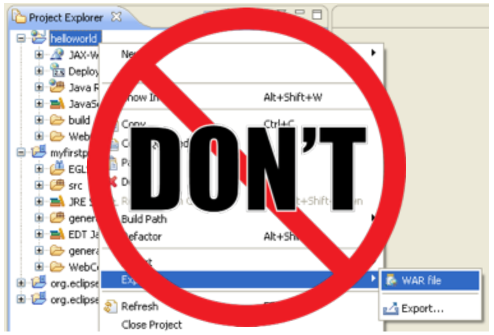
Continuous Delivery - DONTs



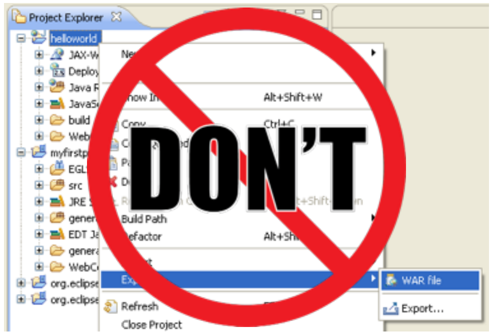
Continuous Delivery - DONTs



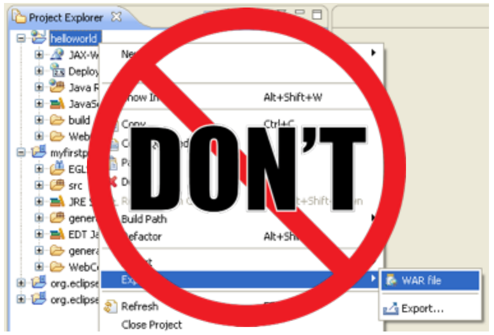
Continuous Delivery - DONTs



Continuous Delivery - DONTs



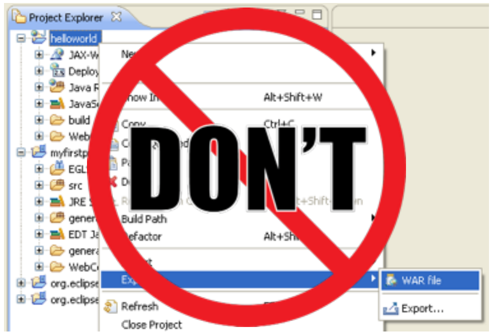
Continuous Delivery - DONTs

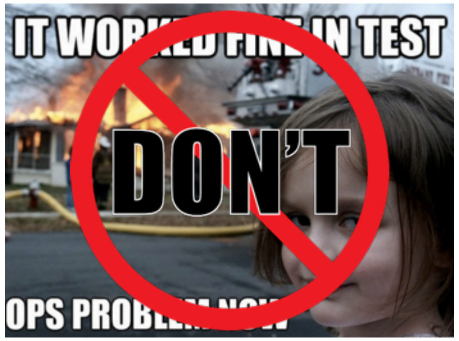


Continuous Delivery - DONTs

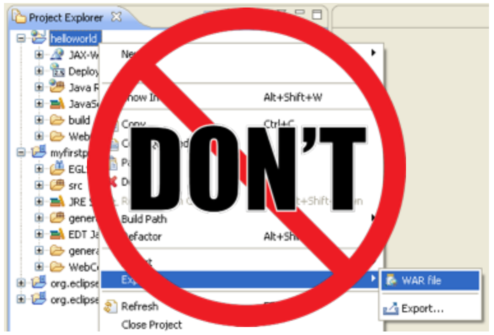
Continuous Delivery - DONTs

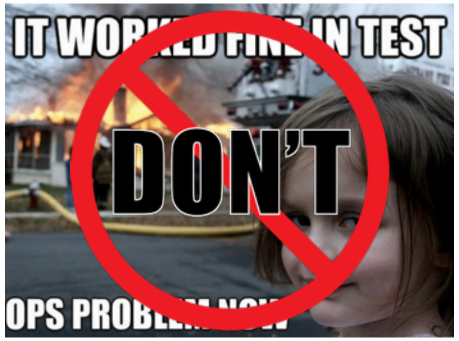
Continuous Delivery - DONTs



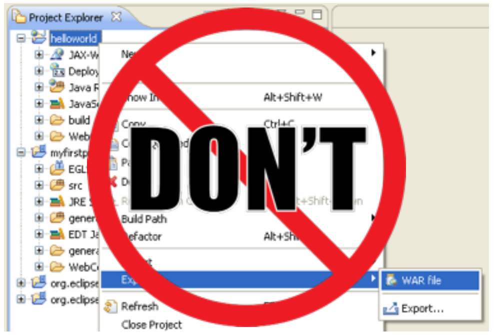


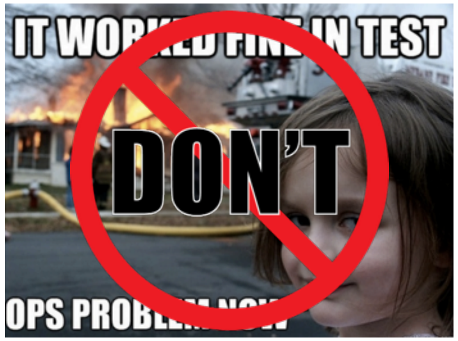
Continuous Delivery - DONTs





Continuous Delivery - DONTs





Benefits of CI/CD

Benefits of CI/CD

- Fast feedback

Benefits of CI/CD

- Fast feedback
    - Improves Quality

Benefits of CI/CD

- Fast feedback
    - Improves Quality
    - Finding a bug early lowers its costs

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs

- Every commit is built

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs
- Every commit is built

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs

- Every commit is built

- Every commit is tested

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs
- Every commit is built
- Every commit is tested

Benefits of CI/CD

- Fast feedback
    - Improves Quality
    - Finding a bug early lowers its costs
- Every commit is built
- Every commit is tested
- Every environment is up to date in no time!

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs
- Every commit is built
- Every commit is tested
- Every environment is up to date in no time!

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs
- Every commit is built
- Every commit is tested
- Every environment is up to date in no time!
- Smaller features get released more often

Benefits of CI/CD

- Fast feedback
  - Improves Quality
  - Finding a bug early lowers its costs
- Every commit is built
- Every commit is tested
- Every environment is up to date in no time!
- Smaller features get released more often

Obstacles

- Technical
- Team
- Organization
- QA
- IT-OPs
- Hardware

Technical Obstacles

- Your software architecture is not ready
- Your start-up process is inadequate
- Your configuration is misplaced
- Your database / -connection is not useable for CD
- Your build-process is processed manually

Team Obstacles

- Your team is not using agile methods (waterfall)
- You don’t use sprints / SCRUM / Kanban
- Your team members don’t trust each other
- Someone in your team „owns“ code
- Your team is pointing finger at each other

Organizational Obstacles

- Team autonomy is a foreign word in your company
- Your company divides responsibilities rigorously
- Your organization misses courage
- New ideas are not welcome

QA Obstacles

- Your QA uses manual testing only
- Your QA don’t trust your testing environments

IT-Operations Obstacles

- Your IT-OPs want to rule the (server-)world
- OPs feel useless when implementing CD

Hardware Obstacles

- You don’t have the proper hardware
- You can not access the target hardware by scripts

Which roles are affected by CD?

- Product Owner
- Product Manager / Project Manager
- Scrum Master
- Developer
- QA
- IT-Operator
- Customer

- **What are the motivations for each role?**

Product Owner / Product Manager

- Time to market increased
- Short feedback loop for features
- Smaller features means more control
- Feature-Pipeline will not be blocked
- Planning becomes more feature oriented
- Smaller planning iterations
- Less specification at a time
    - => Specification on demand
- Immediate technical feedback
- Lower risk
- High PO satisfaction
- High customer satisfaction

Scrum Master

- Small sized, release-able features
- Higher sprint success rate
- Less time spent for planning / retro-meetings

Developer

- Code for Testing vs. Code for Production
  - Every Code is potentially deployed to production environment
- Test Driven Development
  - More important than ever
- Fast Feedback is necessary
  - from QA, PM
- Releasing features is routine

- **Fast feedback improves quality**

QA

- Testing is more focused
- Testing effort is reduced within a sprint
- High quality and even improving
- Test automation is a useful assistant
- More trust in software quality

IT-Operator

- Standardized methods for deployment
- No manual effort necessary for a release
- Focus on stabilizing deployment process
- Focus on monitoring

How can you tame the QA dragon?

How can you tame the QA dragon?

How can you tame the QA dragon?

- A lot of justified arguments:

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!

How can you tame the QA dragon?

- A lot of justified arguments:
    - Developers produce bugs!
    - Environments are not the same!
    - What about cascaded bugs?
    - You don’t have enough coverage!
    - What if you forget about config?

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!
  - What if you forget about config?
  - That’s not enough specification!

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!
  - What if you forget about config?
  - That’s not enough specification!
  - Yeah… of course… „The bug is fixed now“…

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!
  - What if you forget about config?
  - That’s not enough specification!
  - Yeah… of course… „The bug is fixed now“…

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!
  - What if you forget about config?
  - That’s not enough specification!
  - Yeah… of course… „The bug is fixed now“…
- They do their job:

How can you tame the QA dragon?

- A lot of justified arguments:
  - Developers produce bugs!
  - Environments are not the same!
  - What about cascaded bugs?
  - You don’t have enough coverage!
  - What if you forget about config?
  - That’s not enough specification!
  - Yeah… of course… „The bug is fixed now“…

- They do their job:

How can you tame the QA dragon?

- A lot of justified arguments:
    - Developers produce bugs!
    - Environments are not the same!
    - What about cascaded bugs?
    - You don’t have enough coverage!
    - What if you forget about config?
    - That’s not enough specification!
    - Yeah… of course… „The bug is fixed now“…
- They do their job:
    - **Don’t trust developers!**

Integrate QA to development process

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium

Integrate QA to development process

- Test automation
    - REST-tests
    - Selenium

- Specification / Test Cases

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium

- Specification / Test Cases

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium
- Specification / Test Cases
- Estimation, planning, releasing

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium
- Specification / Test Cases
- Estimation, planning, releasing
- Earn their trust!
  - Enable CD for individual environments one after another
  - Every reported bug leads to an automated test

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium
- Specification / Test Cases
- Estimation, planning, releasing
- Earn their trust!
  - Enable CD for individual environments one after another
  - Every reported bug leads to an automated test
- Their feedback is essential!

Integrate QA to development process

- Test automation
  - REST-tests
  - Selenium
- Specification / Test Cases
- Estimation, planning, releasing
- Earn their trust!
  - Enable CD for individual environments one after another
  - Every reported bug leads to an automated test
- Their feedback is essential!

Combine CD with Scrum

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done“; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**
- Design your features small enough:

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**
- Design your features small enough:
  - Specified and documented

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**
- Design your features small enough:
  - Specified and documented
  - They finish within the sprint

Combine CD with Scrum

- ***Scrum emphasizes working product at the end of the sprint that is really „done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable.*** (wikipedia)
- Use short sprint durations!
- User stories / features within the sprint are going to be released!
- **Size matters!**
- Design your features small enough:
  - Specified and documented
  - They finish within the sprint
  - They can be tested by QA

Combine CD with Scrum

Combine CD with Scrum

- Don’t branch!

Combine CD with Scrum

- Don’t branch!

Combine CD with Scrum

- Don’t branch!

- Use feature trigger instead
  - Configuration enables features
  - Use bridge-pattern:
    - Build a new bridge
    - Redirect traffic
    - Tear down old bridge

Combine CD with Scrum

- Don’t branch!

- Use feature trigger instead
  - Configuration enables features
  - Use bridge-pattern:
    - Build a new bridge
    - Redirect traffic
    - Tear down old bridge

- Configuration is part of a deployment!

Combine CD with Scrum

Combine CD with Scrum

- Your feature toggle comes first
    - Need a hotfix? No problem!

Combine CD with Scrum

- Your feature toggle comes first
    - Need a hotfix? No problem!

- There is no „step back", we just deploy forward

Use the power of communication

Use the power of communication

- Encourage communication

Use the power of communication

- Encourage communication

Use the power of communication

- Encourage communication
- Claim for early feedback
  - Both ways (PM <-> Dev)

Use the power of communication

- Encourage communication
- Claim for early feedback
  - Both ways (PM <-> Dev)
- QA creates test scenarios along with specification

Use the power of communication

- Encourage communication
- Claim for early feedback
  - Both ways (PM <-> Dev)
- QA creates test scenarios along with specification

Use the power of communication

- Encourage communication
- Claim for early feedback
  - Both ways (PM <-> Dev)
- QA creates test scenarios along with specification
- Got a question during a sprint? Ask right away!

Use the power of communication

- Encourage communication
- Claim for early feedback
  - Both ways (PM <-> Dev)
- QA creates test scenarios along with specification
- Got a question during a sprint? Ask right away!

Will DANKE sagen für die Aufmerksamkeit

WILLHABEN.AT®