HOW TO BE PROUD WHEN YOU ARE DONE

Mikalai Alimenkou

- Java Technical Lead/Scrum Master at **Zoral Labs**
- 6+ years in software development
- 4+ years of working by Agile methodologies
- **Expert** in Agile engineering practices
- Agile coach at **XP Injection**

Aleksey Solntsev

- Architect at **Infopulse Ukraine**
- Agile volunteer
- **Certified Scrum Practitioner**
- Coordinator of translation of the book "**Scrum and XP from the Trenches**" and "**Kanban and Scrum – making the most of both**" into Russian
- Agile coach at **XP Injection**

Everybody has own “dreams”

- ✓ Latest frameworks
- ✓ No overtimes
- ✓ No bug fixing
- ✓ Clean code

Team

- ✓ More features
- ✓ In time delivery
- ✓ No defects!

Customer

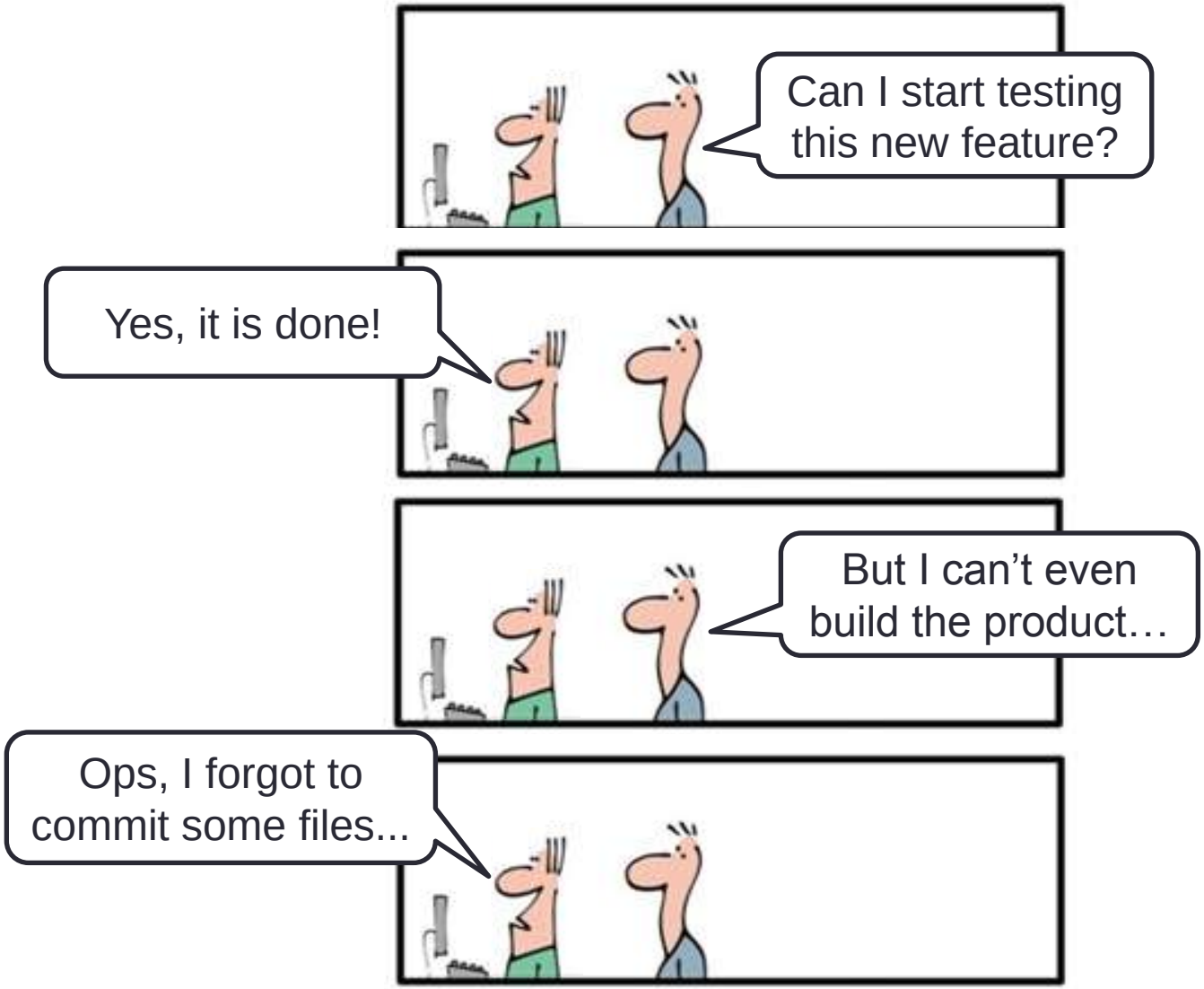
- ✓ Goal achievement
- ✓ Predictable plans
- ✓ On budget

Manager

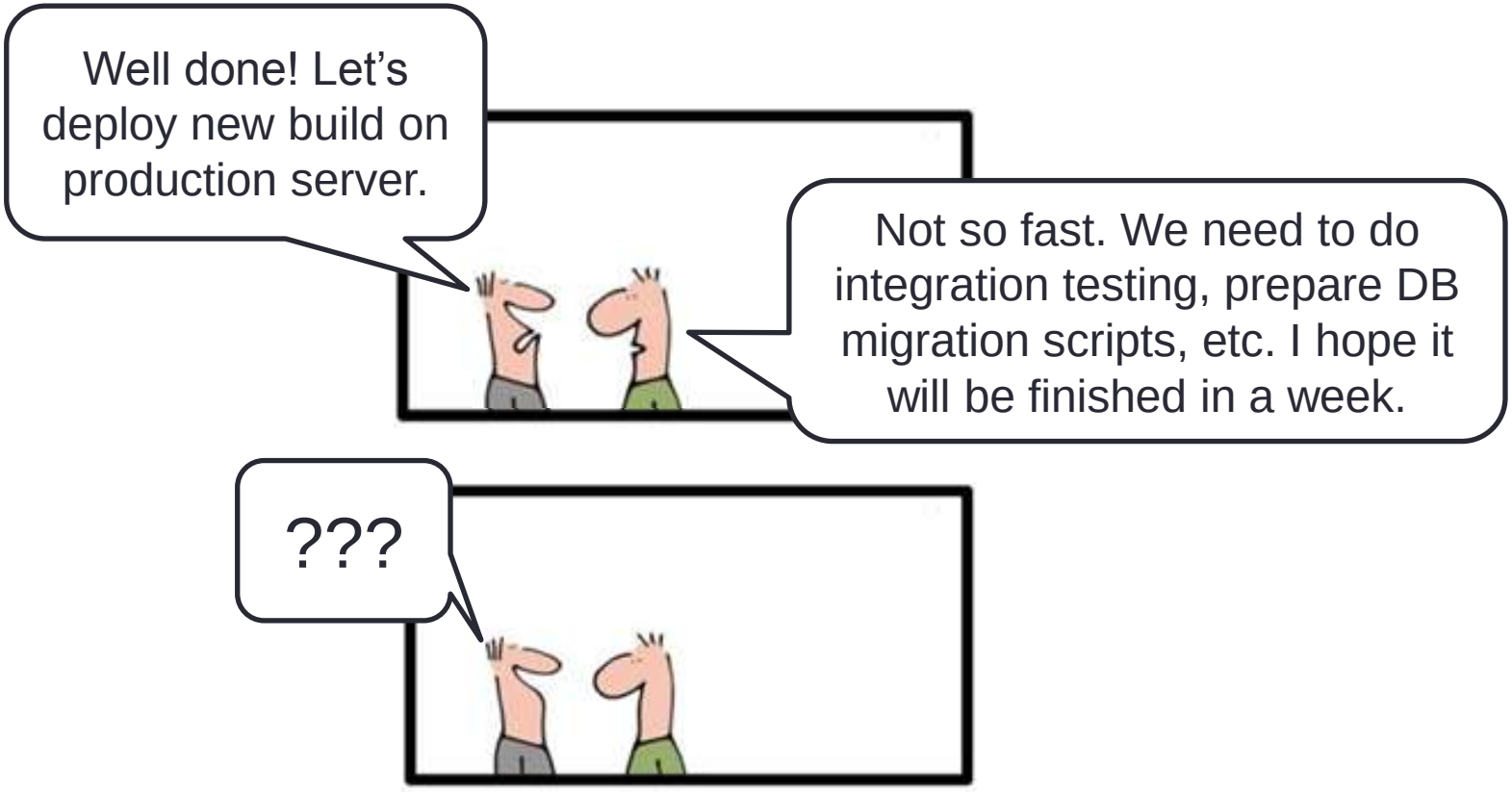
What all of them have in common?

Why in real life it is not so simple?

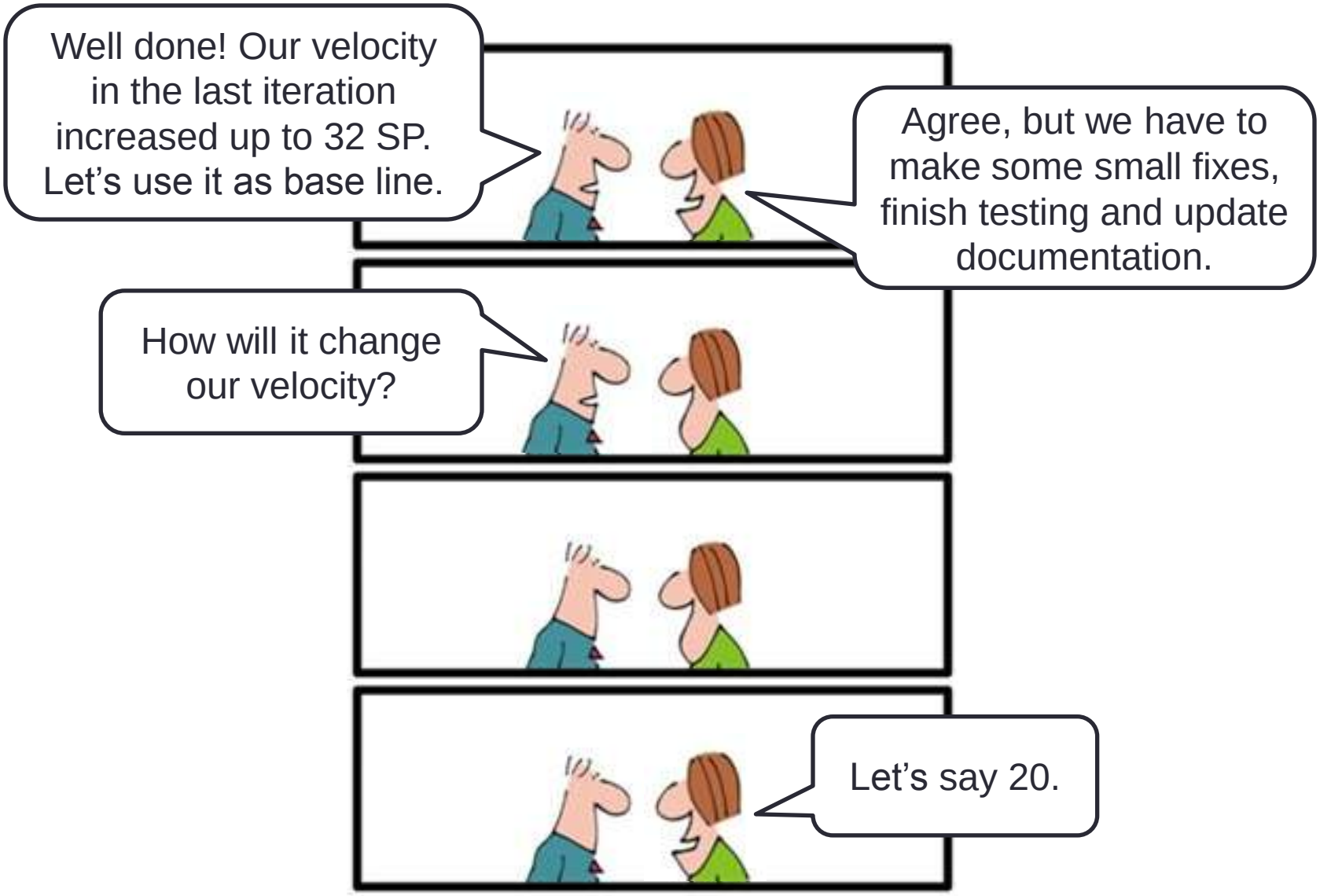
“Uncommitted stuff”



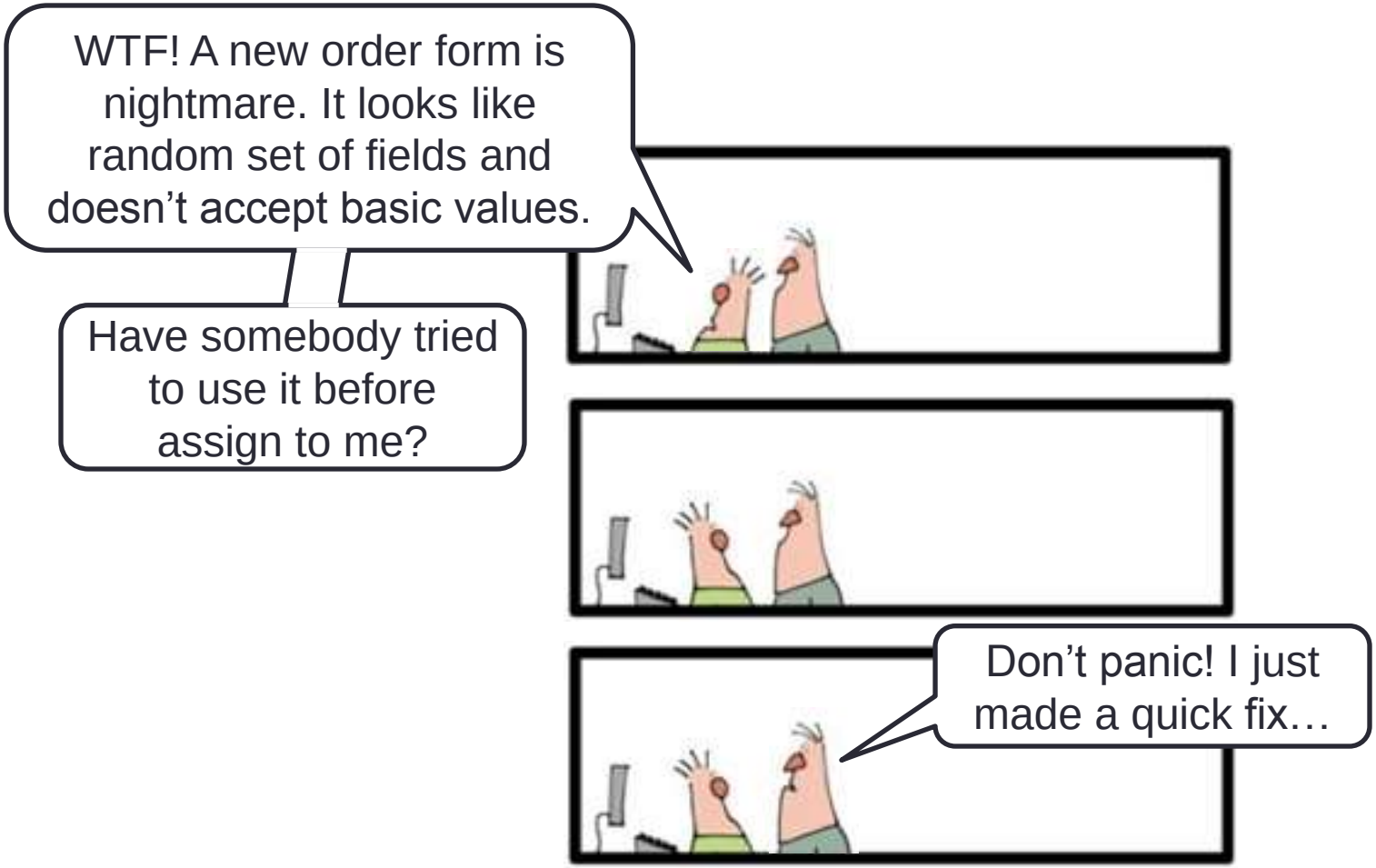
“Useless build”



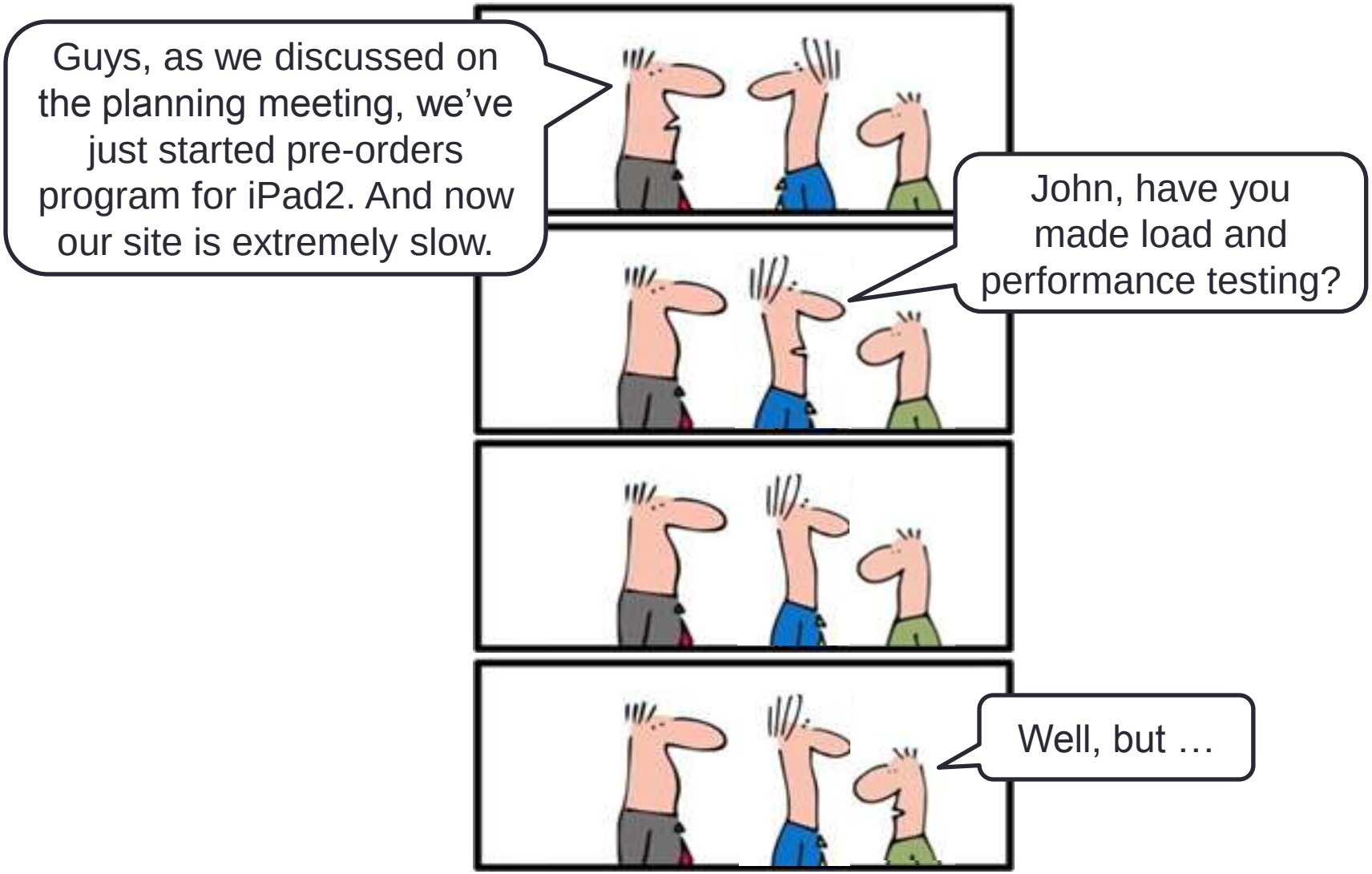
“Unstable velocity”



“Unverified tasks”



“Forgotten requirements”



Why this happens?

Hidden conflicts in goals

- **Developers** like always implementing interesting features, but **customers** want working software
- **Team** want to show productivity, but **customers** want predictability
- **Developers** believe they are perfect, but **customers** want software without defects
- **Management** want to deliver in time, but **customers** want ready to use software
- **Developers** see technical side of the project, but **customers** see it from end users perspective

And the main reason is …

Everybody has his own definition of words ‘done’, ‘fast’, ‘low price’, ‘good quality’!

Let’s start from definition

“Definition of Done” is

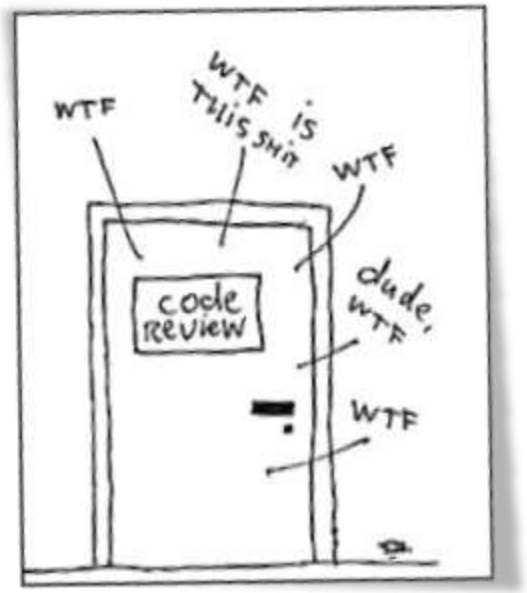
- ✓ a "contract" between all parties on subject when …
- ✓ a checklist of valuable activities for …
- ✓ gates your product has to go through before …

… you can label the product “Done“

4 main questions

How to start?

Take from previous project



From code quality in mind

From business problems and wishes

Start from flow visualization

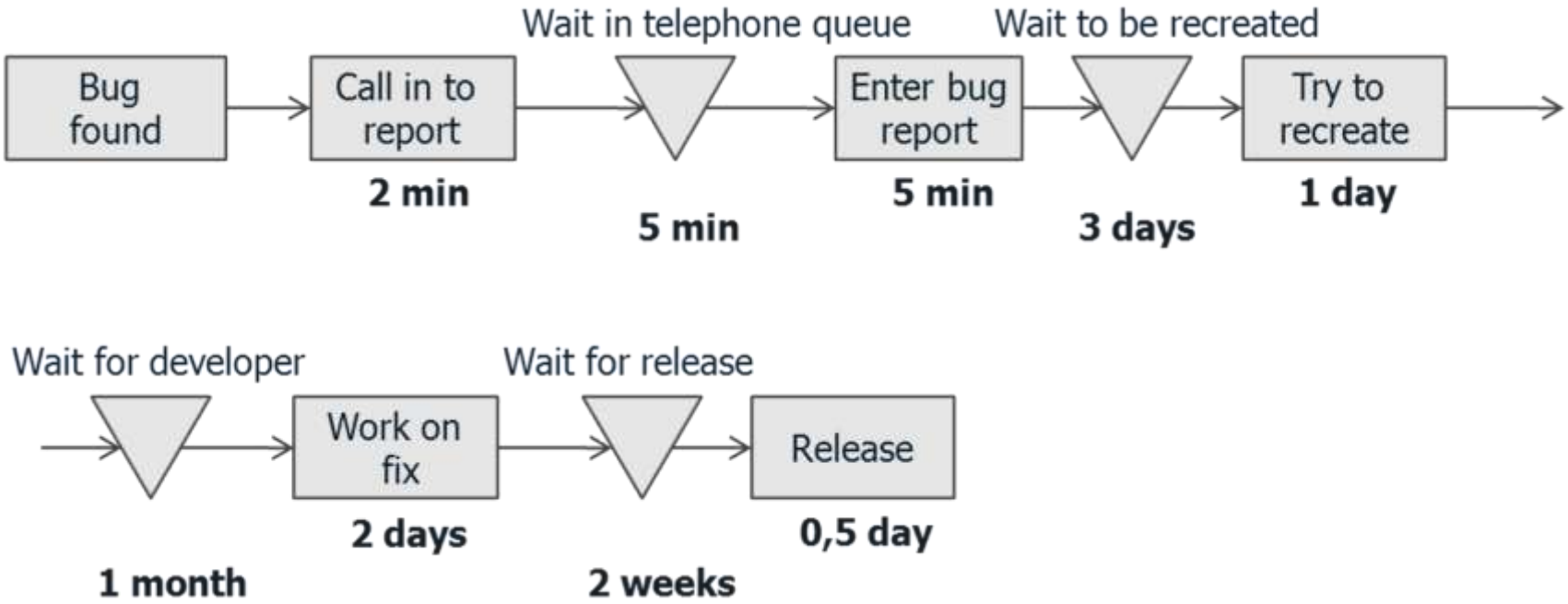


Image from Henrik Kniberg book “Scrum and Kanban - making the most of both”

Who should define?

Customer

Team

When to define?

Where to store?

Electronic

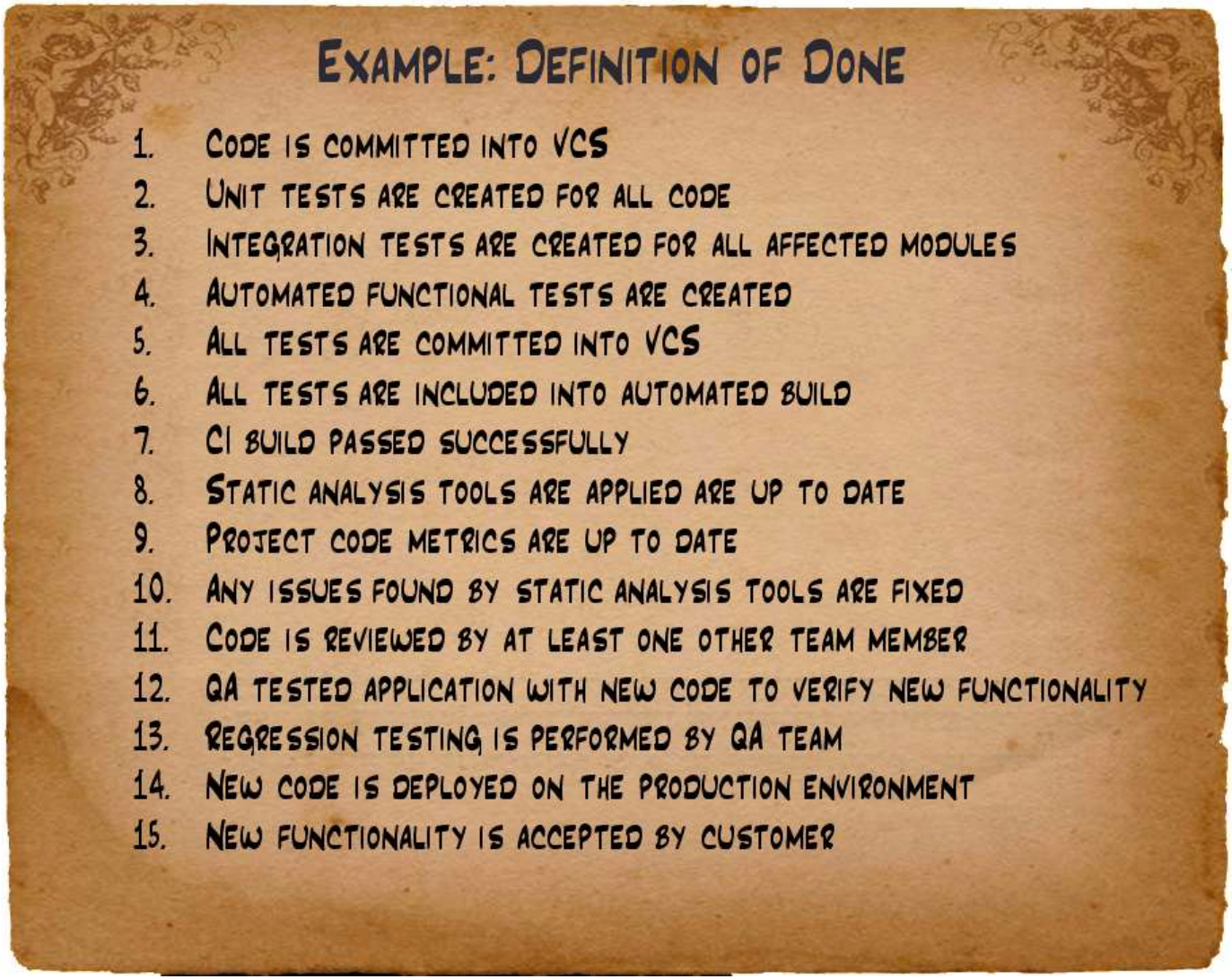
More common

More personal

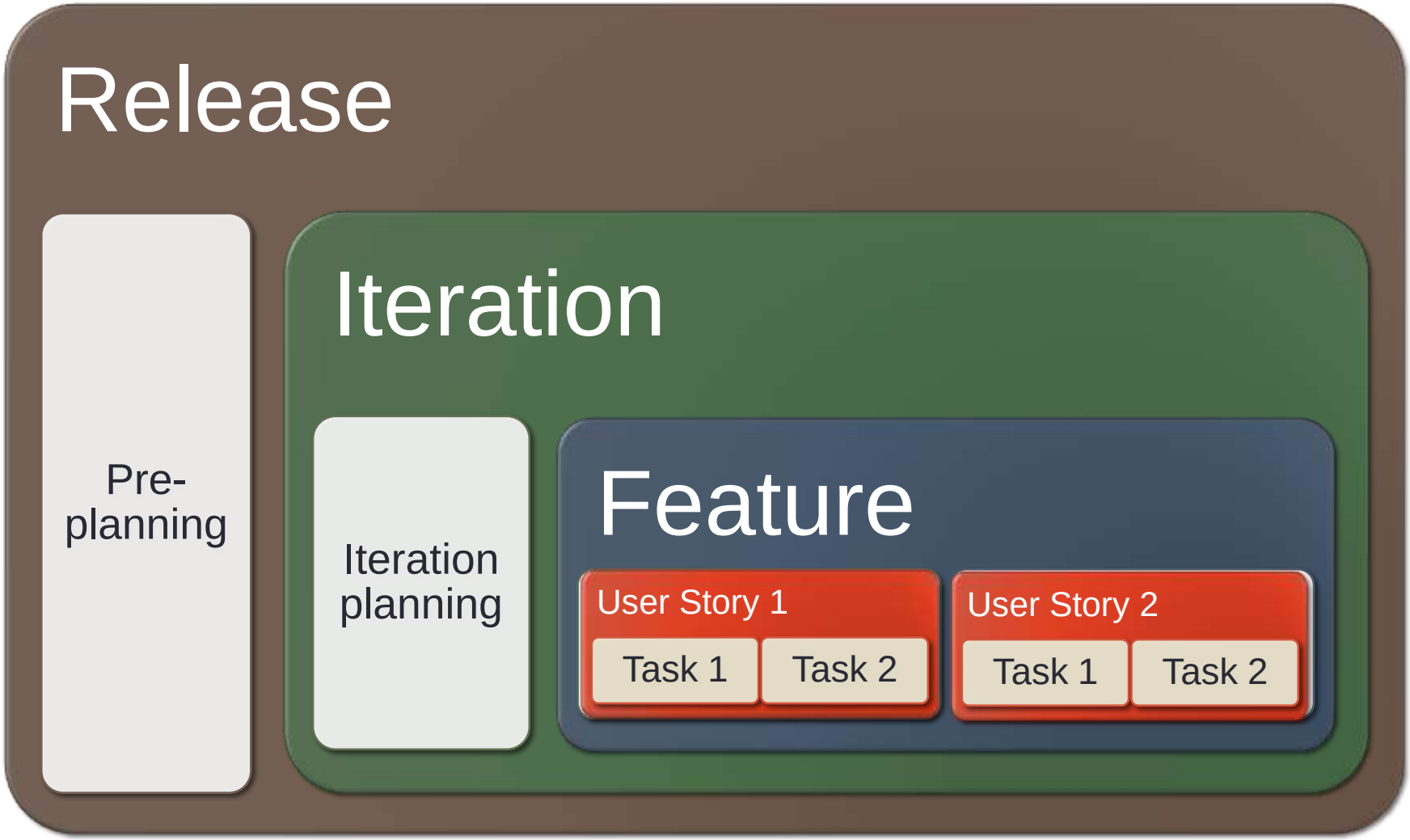
Paper

3 main formats

Plain list


Example: Definition of Done
1. Code is committed into VCS
2. Unit tests are created for all code
3. Integration tests are created for all affected modules
4. Automated functional tests are created
5. All tests are committed into VCS
6. All tests are included into automated build
7. CI build passed successfully
8. Static analysis tools are applied are up to date
9. Project code metrics are up to date
10. Any issues found by static analysis tools are fixed
11. Code is reviewed by at least one other team member
12. QA tested application with new code to verify new functionality
13. Regression testing is performed by QA team
14. New code is deployed on the production environment
15. New functionality is accepted by customer

Different levels of granularity



List categorized by level

Pre-planning

- Criteria 1
- …
- Criteria N

User Story

- Criteria 1
- …
- Criteria N

Iteration

- Criteria 1
- …
- Criteria N

Task

- Criteria 1
- …
- Criteria N

Feature

- Criteria 1
- …
- Criteria N

Release

- Criteria 1
- …
- Criteria N

Complex structure

Pre-planning

User experience	
Test	**Has QA been part of preplanning?**

Iteration

User experience	
Test	
Documentation	

Feature

User experience	
Test	
Documentation	

User Story

User experience	
Documentation	**Has the end-user documentation been updated?**

Task

Working software	**Has the implementation been reviewed be a peer?**
Test	

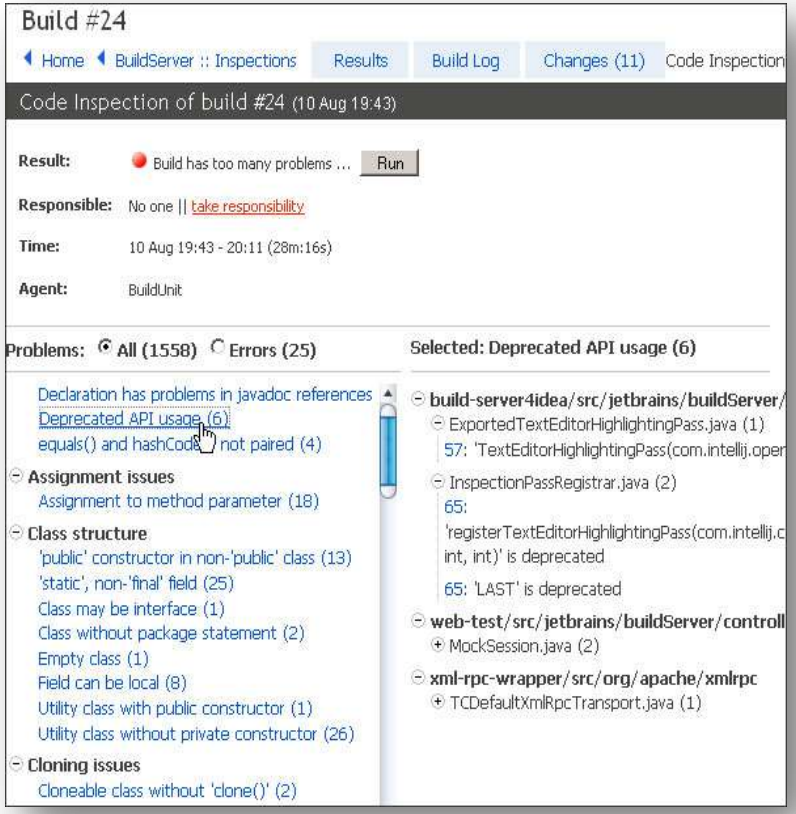
Release

User experience	
Test	**Is the scalability testing done?**
Documentation	

3 main control principles

Automation

Use SVN hook for verify comments in commits

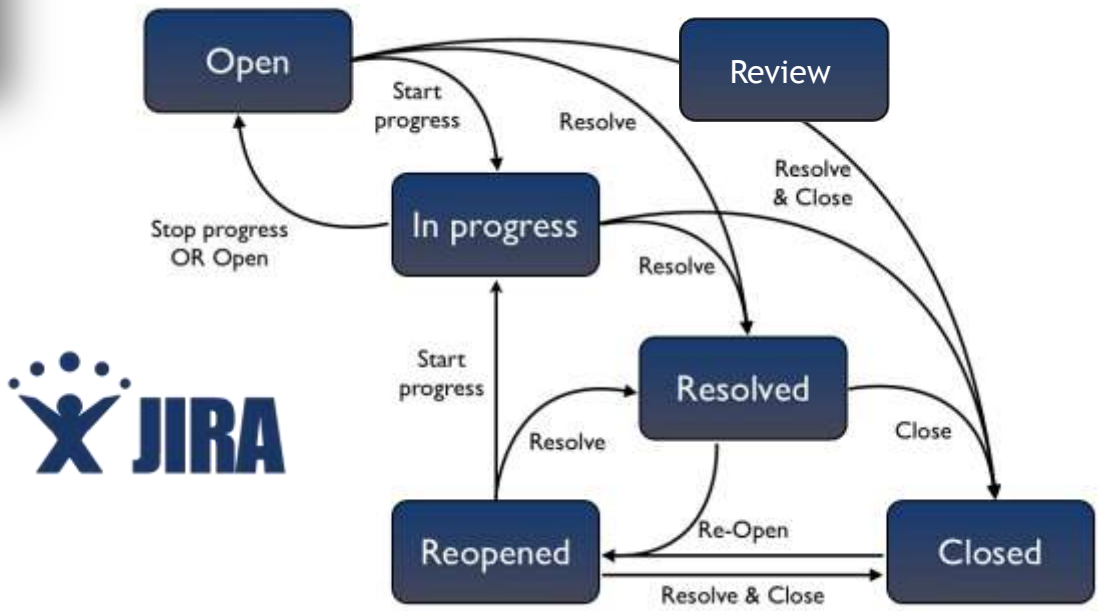


Add static analyzer to build

Fixed workflow

Fix and document workflow

Task tracking system can help



Responsible persons

How does it work in real world

Binary state

Not done

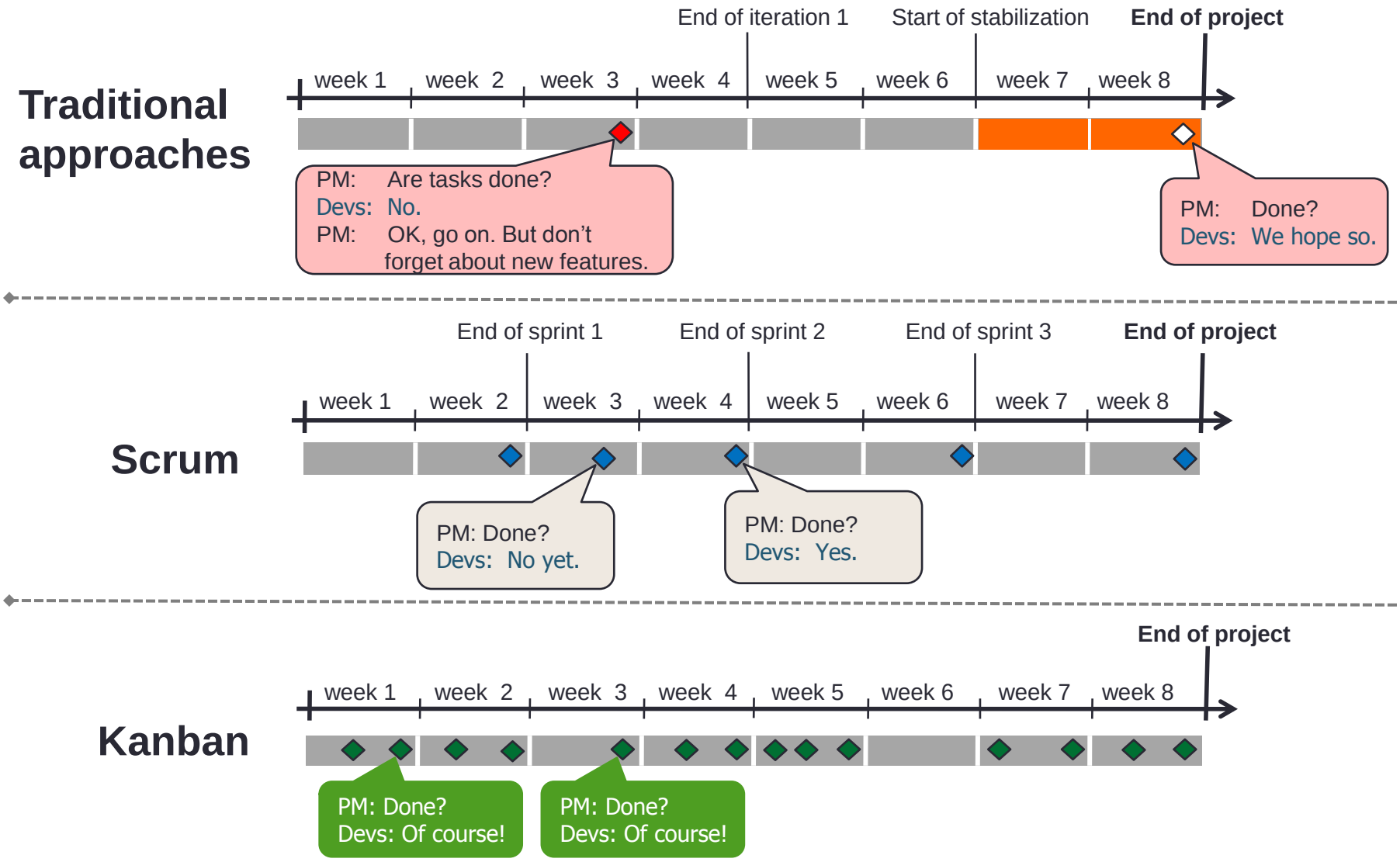
Will be done

In progress

Nearly done

Done

Depends on context



7 main problems

No common understanding

No commitment

Unrealistic

Too ideal

Partially done tasks are accepted

"Broken windows" principle

“Development” only

Let’s try to build Definition of Done!

Conclusions

Common vocabulary helps us to avoid hidden conflicts …

… and work together as a team to archive our goals!

- Introduce “Definition of Done” ASAP to avoid broken expectations
- All parties must take part in definition process
- Automate as much as possible
- Don't lie yourself, DONE means DONE
- Learn from your mistakes
- Inspect and adapt continuously

Q&A

Email us:

- mikalai.alimenkou@xpinjection.com
- aleksey.solntsev@xpinjection.com

XP Injection