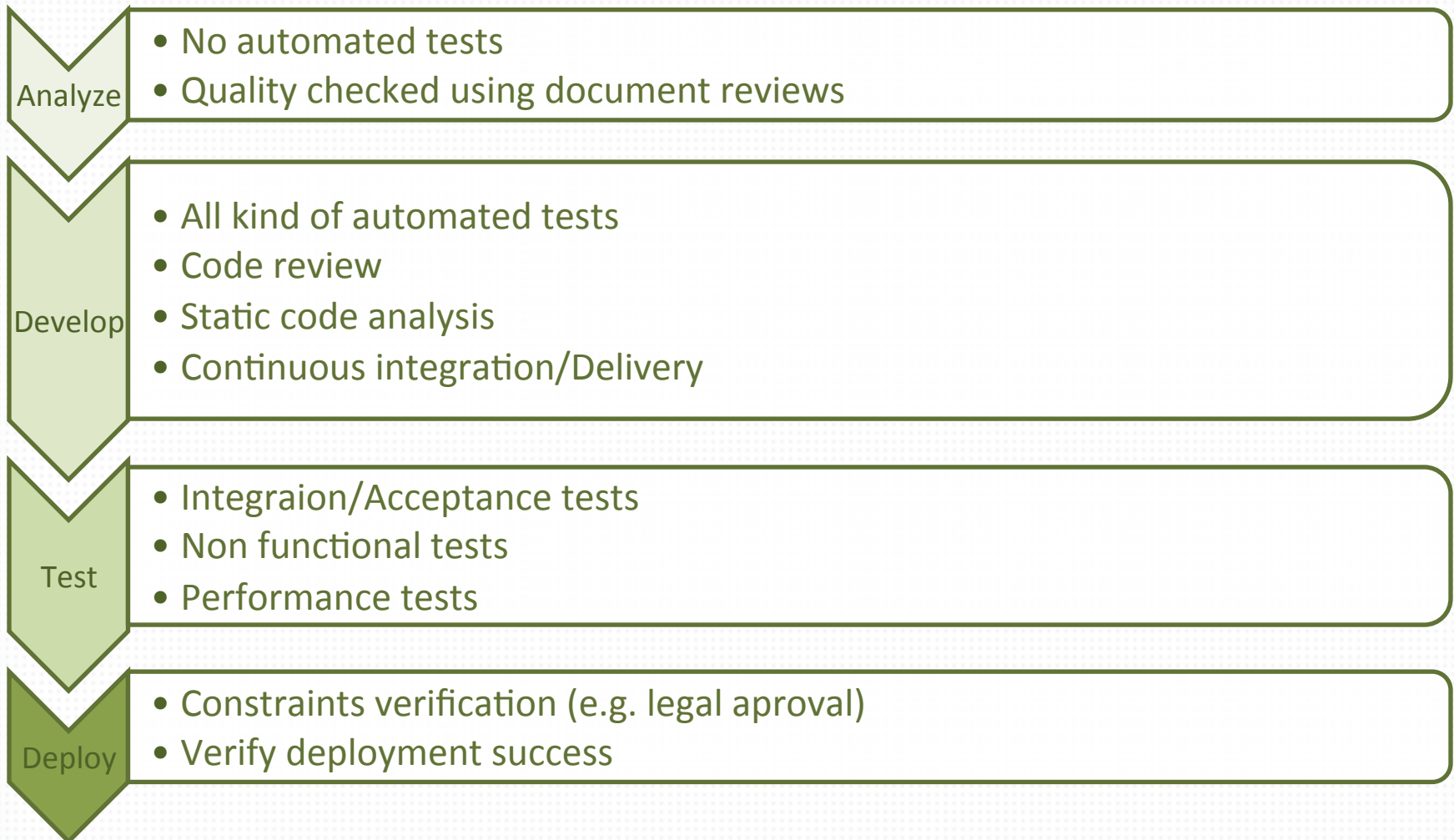


# TEST DRIVEN DEVELOPMENT AND CONTINUOUS INTEGRATION TOOLS



# DEVELOPMENT PROCESS REVISITED



- No automated tests
- Application switches for testability
- Automated unit/integration tests
- Test driven development (TDD)
- Continuous testing

# TEST DRIVEN DEVELOPMENT

- Red-green-refactor technique
- Requires to know, which lines of code are tested
- Developer needs to know, what has changed since last submit
- No change can be done without automated tests

- Percentage of lines of code exercised by tests
- Extends test execution time (cca 3x in .Net)
- 100% is required by TDD
- 100% not feasible because of costs  
(development with tests is 2-4x more expensive)
- Sensitive to interpret results:  
100% coverage does not mean bug free software.  
Still it means that only few percent  
of possible application states are tested.

- How to get code coverage report in Visual Studio?
- How to find tests covering my lines?
- How do I practice TDD?





















- Implement complete development and verification pipe line using one tool (Ordered list of actions)
- Can measure basic project health statistics
- **Always green technique** (Stop developemnt, if something is wrong)
- Focus on integration of external tools:
  - Programming languages and platforms
  - Source controls
  - Build engines
  - Packaging repositories
  - Issue trackers
  - **Notifications** (mail, system tray, messangers)

# CI PIPE LINE STEPS

- Compile
- Test
- Static code analysis
  - Code coverage
  - Coding rules
  - Code issues
  - Code duplicities
- Package (download, create, publish)
- Deploy



# THE PROCESS

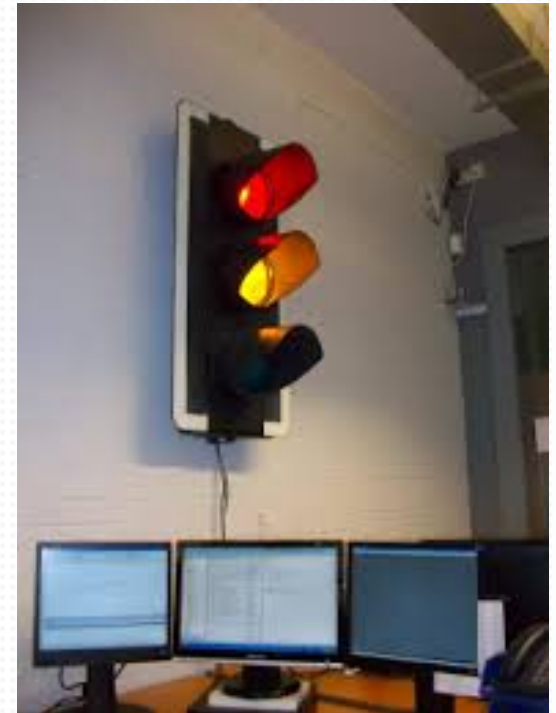
Build	Compile	Static code analysis	Tests	Package	Deploy
#1					
#2					
#3					
#4					

- **Success rate:**  
Percentage of failed builds during last iteration
- **Time to fix test:**  
Time interval in minutes between failed test discovery and till the fix is available
- **Average time to market:**  
Time interval in days between two versions delivered to the customer
- **Project health trends:**
  - Code coverage trend (more is better)
  - Number of code duplicities (less is better)
  - Number of code issues (less is better)
  - Number of coding rules violations (less is better)

- Configure simple pipe line using TeamCity
- Analyze current state of project health using TeamCity
  - Current state of project
  - Success rate and time to fix tests
  - Project lifetime Trends

# BONUS – TEAM GAMIFICATIONS

- Used to improve team morality
- Who causes build to fail  
pays money to team wallet
- Traffic lights are used to visualize state  
or board is shown on TV in kitchen
- Nobody can go home  
(door are locked) till issue is fixed
- Person who caused most issues  
during last iteration  
is presented on Board of fame



- MUNI course: PV179 Selected Topics in .NET Technologies
- Windows user group
  - Continuous testing using TeamCity
  - Test able code and test first in .Net

# CONTACT

- Jiří Pokorný  
senior developer at SolarWinds
- [Jiri.pokorny@solarwinds.com](mailto:Jiri.pokorny@solarwinds.com)
- Organizer at Windows User Group Brno
- Jobs and internship programs