Week 4

Devops

WRITE TOPICS HERE

Monday 5/13/19

Github branching

See diagram in notebook for github branching

Remote tracking branch -origin master

Local branch – master

Git pull

Git fetch

Updates remote tracking branches

Git merge

Merge origin/master into master

Web servers

* IIS
* IIS express
* Kestrel

In Visual studio

IIS Express <-web server

~~Kestrel <=app server~~ (Updated) no kestrel for vs on .net core 2.2, now runs directly on IIS express, “in-process hosting model”

Asp.net (our code)

Dotnet run can run asp.net

Agile development

Devops

//5/15/19

Added azure pipeline

CI-continues integration

* Frequently, developers integrate their work together using version control system, with automated checks for code correctness
* “CI pipeline”, “build pipeline”
* When there is a new commit on master branch,
  + pull code
  + run code
  + run automated test (such as Xunit)
  + static analysis (scans for common mistakes/makes suggestions, there’s tools for this)
  + package/publish build output
  + deploy to some development server

CDe-continuous delivery

* new builds of code deploy all the way to the client/public automatically, but with manual approval

CD-continuous deployment

* new builds automatically deploy to the end user automatically period

CI/CD tools

* Azure DevOps (formerly know as VSTS) (SaaS)
* Jenkins (software you install on computer)
* Appveyor (for .net)
* TravisCI
* CircleCi
* GoCD

Project board tools

* JIRA
* Azure Boards
* Trello
* Pivotal Tracker
* GitHub projects

guerrerof1904HardwareStore

franciscog123-github

Analyze "guerrerof1904HardwareStore": 1b664100ad20bec2888052cb25a203e5d73b4aa3

Code coverage

@of (lines, blocks)

That at least one test runs

Code smell

Technical debt

Artifact-anything we want to keep track of outside the build pipeline, any output files

SDLC (software development lifecycle)

* Models
* Requirements
* Design
* Build
* Test
* deploy

Big bang

1. Requirements
2. Design and build and test
3. Deploy

Suitable for: prototyping

Downsides:not sustainable. unmantainable

Waterfall

1. Re
2. quirements
3. Design
4. Build
5. Test
6. Deploy

Suitable for: high need for accountability, security, “mission-critical”

Downsides: unrealistically inflexible

-slow

**Iterative:**

Spiral



Mini waterfall

**Agile:**

* Communication between everyone concerned with the project (stakeholders) **especially the client**
* Divide requirements into pieces (user stories)
* Each story is designed semi-independent

CMMI:

Scrum and Kanban (both Agile)

|  |  |
| --- | --- |
| **Scrum** | **Kanban** |
| Scrum board  -User stories in columns according to status | Kanban board |
| Sprints of fixed length (eg 4 weeks) with pre-selected stories | No sprints |
| Sprint planning  -estimate effort needed for stories  -estimate capacity (effort points we can do) | Constantly developing and self evaluating |
| Daily standup/scrum meeting (scrumlord)  -what everyone did previous day  -what will everyone do  -blockers | We limit # points that can be in a column at one time |
| Sprint retrospective  -how many points actually completed (velocity) |  |
| Backlog – the user stories that aren’t done/started |  |

Devops

See devops diagram in notebook

Docker