CS314 Review:

**Team Development Stages**

Forming, Storming, Norming, Performing, Adjourning (in order). FS-NPA.

Forming: Orientation, testing, dependence

Storming: Resistance to group influence / task requirements

Norming: Openness to other group members

Performing: Constructive action

Adjourning: Disengagement

Group Structure Activity

Forming: Testing / Independence Orientation

Storming: Conflict Emotional Response to Task

Norming: Cohesiveness, new roles Open exchange, personal opinions

Performing: Roles = flexible, functional group energy channeled to task

Adjourning: Sadness about separating Self-evaluation

**Configuration Management**

Version Control – tracks component changes as they occur

System Building – Assembles component for testing and release

Change Management – Addresses stakeholder proposals

Release Management – plans / prepares for distribution

Version Control: can be *distributed* or *centralized*. Distributed is like Github, where multiple branches / copies exist and developers can work offline, while with centralized, master contains all versions (SVN). Both still offer version/release identification, change history recording, independent development, project support, and storage management.

System Building: involves everything used to ensure master can compile, be built / run, passes tests, integrated, and documented. Precisely, it involves: build script generation, build system integration with version control system, minimal recompilation, executable system creation, test automation, report success/failure of build/tests, automatically generated release notes and documentation.

Change Management: Ensures changes are applied in a controllable way (requirements, bugs, etc).

Release Management: Plan the release, including: prepare system for release (config files, data files, installation program, documentation, packaging and publicity).

**Github/ Git Usage**

Github = server side

Git = local clone / branch

Commands:

#update local copy: git pull origin master

#new branch: git checkout –b [newbranchname]

#verify it is branch (not master): git branch

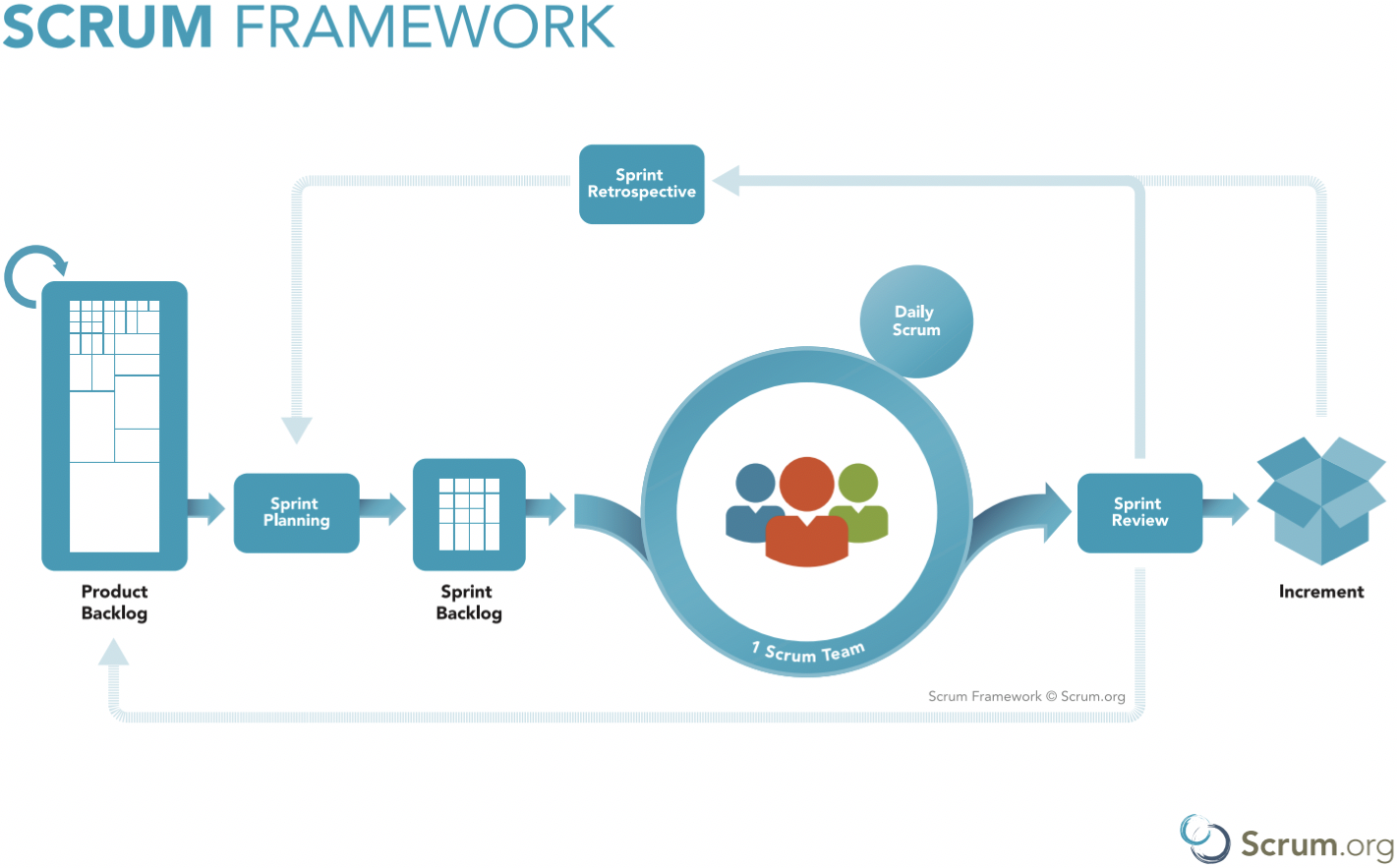
#add changes to branch: git add [filename(s)]

#verify proposed changes: git status

#associate with task/issue: git commit –m “closes #999”

#push branch: git push origin [newbranchname]

**Scrum – Process, Participant, Artifacts, Events**



Artifacts: Product & Sprint Backlog (Epics), Product Increment

Events: Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective

Participants(?): Product Owner? Development Team? Scrum Master?

Process: Product Owner defines objectives, scrum team works to understand what is necessary, development team selects Product Backlog items (epics), and scrum team defines a sprint goal.

**Scrum Planning – Milestones, Epics, Tasks, Estimates**

Epics = backlog items, or additional functionality to be implemented

Tasks = Epics broken into smaller pieces, or how Epics are completed

Estimates = can be inaccurate, can use “story points”, or essentially sizes in relation to other tasks. Can also do planning poker, or fibonacci cards…

Sprint plan = Epics and tasks to accomplish those epics.

**Test Driven Development Stages**

1) Write method signatures, Javadocs

2) Write tests (fail)

3) Write code to make tests work (pass)

4) Clean, refactor code (pass)

assertEquals(ExpVal, testVal);

**Black Box Testing / JUnit**

Black box refers to giving an input, and testing output is as expected without even looking at code (black box).

Equivalence Classes: inputs that should be treated similarly (i.e. testing 4, then 5 – instead of 0, or -99, etc.).

Equivalence Partition testing: Take at least 1 input from each equivalence class for tests

Boundary Value testing: test edge cases

Test methods should have @Test above method declaration, and have return type = void

**CMMI level 2/3 process areas**

Process areas of CMMI: Process Management, Project Management, Engineering, and Support. Specific process areas:

Level 2: REQM, PP, PMC, SAM, MA, PPQA, CM

Level 3: RD, TS, PI, VER, VAL, OPF, OPD, OT, IPM, RSKM, DAR

In the above: C is typically configuration, M management, P project / process, A analysis,

O organization, T training / technical, I integration, PP project planning, Q quantitative, S solution / supplier, V validation, R requirements / risk, D development / definition / decision

Common features: commitment to perform, ability to perform, activities performed, measurement and analysis, verifying implementation.