# Automation

JD Kilgallin

CPSC:480

11/09/22

*Pressman Ch 19-20*

*Why use many word when few do trick?* -Google automatically suggested search term

## Notes

- Exercise 6 repository has updated solution and grading script.
- Project 3 checkpoint tonight; all reviews should be approved & merged.
- Project 3 first early submission deadline this Saturday for 2% extra credit, final submission deadline Monday.
- Project 4 final handout will be released Monday.
- Mikyla guest lecture, 1 week from today on UI/UX & product ownership
- Bonus presentation topics have been claimed for IoT development, cloud computing, and joining a team at entry level.
- Final exam 4 weeks from today; practice final out in 1 week.

## Project 4 – Objectives

- Continue to participate in the software development lifecycle activities like planning, design, project management, construction, and testing.
- Gain experience with software releases, prototypes, documentation, & SCM.
- Present software prototypes to stakeholders and incorporating feedback.
- Respond with agility to changing requirements.
- Assess software quality metrics and project viability.
- Document your software development process for other engineers that might work on your project.
- Report on strengths and weaknesses of the product, project, team, and individual contributions.

## Project 4 – Checkpoints

- Part 1: Adapting Project Requirements. Reprioritize remaining work to focus on the best prototype demonstration Nov 28. To be discussed more Monday.
- Part 2: Completing Demoable Prototype. Checkpoint 1 will be to reach a "code complete" state for your demo prototype. This will be due **Tuesday, Nov 22, 11:59 PM**, just under two weeks from today. It is only expected that your prototype will reflect 1-2 sprints' worth of work, similar to what you might show at a stakeholder feedback meeting after the first 1-2 sprints of a regular software development project that would still be ongoing. It is *not* required that everyone contribute to this development.
- Part 3: Project Summary. Checkpoint 2 will be to complete ancillary materials describing the current state of your project, the work that would be required to continue development toward a viable product, documentation of your software development process, and other features. This will be due **Sunday, Nov 27, 11:59 PM.** To be discussed more Monday.

# Project 4 – Demonstration and Presentation

- Part 4: Stakeholder Presentation. You will take 10 minutes of class time on **Monday, Nov 28**, to demonstrate your software prototype and present other elements of your development project and plan to the rest of the class.
- The scenario you should target is that I am planning on taking over your project at the end of the course and hiring the rest of the class to continue development over the next semester/year.
- The rest of the class will review the presentation as though they are choosing an engineering project to join. They should be convinced that the product design has merit, the software quality is high enough to facilitate development, the existing team members are committed, and the requirements and timeline to complete the project are clear.
- I will review the presentation as if I'm selecting one of the projects to buy and move forward with as the project's new owner-director, and employing the team. I should be convinced that your project is worth investing in and that I can trust the team to execute the project as planned.

## Project 4 – Final Submission

- Part 5: Incorporating Stakeholder Feedback. You will review the feedback you received from the presentation and other sources up to this point and determine how you might adjust your design, code,
plan, or other components and make any final updates to the project.
- Part 6: Final Report. You will each assess your team's development project and progress over the semester, your own contributions, what you learned working on it, and your recommendations for my course of action as the new owner-investor-director of all six teams' projects. You should write this for the perspective that I'm considering hiring you individually to continue development on my selected project, and convince me that you would be an effective member of my development team.
- Project Survey. You will complete another survey quiz on project 4.
- Part 5 will be through commits to the team GitHub repository. Part 6 and the survey will be done individually and submitted through Brightspace. Absolutely all work is due by **Friday, Dec 2, 11:59 PM.** You may submit everything 24 hours early for 2% extra credit, or 6 hours early for 1%.

## Project 4 – Grading

- Grades will be 20% individual and 80% shared, broken down evenly as follows:
- 10% – Part 1 meeting notes and artifacts
- 10% – Part 2 implementation
- 10% – Checkpoint 1 requirements met
- 10% – Part 3 materials
- 10% – Checkpoint 2 requirements met
- 10% – Part 4 project presentation
- 10% – Part 4 prototype demonstration
- 10% – Part 5 final submission
- 10% – Part 6 report (individual)
- 10% – Effective defense of your project and contributions (individual)
- 0-2% Extra credit for early submission
- 0-2% Extra credit for top 3 teams convincing me of the project and contributors' value.
- Additional penalties & awards on an individual basis per project survey results as always.

## Learning objectives

- Purpose of automation
- Types of automation
- Automatic execution of software development tasks
- Automating other processes

## Software automation

- The application of software technologies to minimize human input required for a task.
- Three main goals:
  - Efficiency – Reduce workload by eliminating extraneous steps so developers can focus on other tasks.
  - Reliability – Reduce human error by allowing computers to execute defined procedures.
  - Productivity – Provide self-documenting, distributable solutions that can be used across a team and beyond.
- Widely used by many IT disciplines to automate a broad range of business processes and tasks across the whole organization.
- Software engineers are uniquely positioned in an organization with the skillset to implement advanced, high-impact automation.

# Automation Types Accessible to Non-engineers

- Formulas – Automated calculations
  - e.g. spreadsheets (Excel), Customer Relations (CRM) systems (Salesforce).
  - Used by revenue, finance, payroll, project management (e.g. velocity), etc.
- Macro ("Macroinstruction") – Transforming predefined input into predefined output
  - e.g. hotkeys and shortcuts, autocomplete, spreadsheet manipulation, browser or file bookmarks. C pre-processor directives are macros for software engineers.
- Rules – Handling logic for event and data management (e.g. deleting emails from a spam sender without ever going to the inbox)
- Reports – Data analysis, data mining, and "number-crunching" to distill large data sources into actionable insights.

## Automation Types Needing Engineering Skills

- Script – A list of commands in a specific platform to accomplish a task
  - Distinctions between a "script" and a "program" vary but a common feature is "interpreted" input (computer operates on the source directly) vs "compiled".
  - PowerShell, bash, and Python are popular scripting environments.
  - Some true programs are written just to automate development of other programs, or other tasks.
- Plugin – Many business tools and software platforms support user-defined extensions to augment out-of-the-box functionality.
- Scheduled action – Script, program, or function that executes automatically at a given time, interval, or event ("cron job" in Linux).
- Software Development Kit (SDK) – Library to automate integrations.

## Applications in Software Development Cycle

- Build – Script run from IDE or command line to build software locally, and running automatically on a build server for smoke & nightly tests.
- Test – Framework to run entire or partial suite of automated unit, integration, and scenario tests manually, on specific events, and on a regular schedule, and report results.
- Checkin – Script to perform all actions needed to check in code (e.g. git add, git commit, git pull, build, smoke test, git push, send email)
- Environment setup – Configure and launch containerized application, create test machine and install software pre-requisites
- API and documentation generation – HTTP request structure can be automatically inferred from server request listening code. This can be used to generate examples and explanations for request parameters and entire API client SDKs in a variety of programming languages.

## Other Applications as a Software Engineer

- Data transformation and interoperability – Convert the output of one program or operation into the input for another (e.g. convert json returned in a response from a web request into xml in a file)
- File and environment management – Quickly access programs you need, copy build output to test machine, delete/archive log files.
- Project management, work tracking, and systems access – Browser extension or command-line tool to quickly open a page to a given bug number or search for a term in work item backlog.
- Communication and reporting – Generate data for employee timesheets by quickly entering a category just before logging off for what you were doing (coding, zoom call, project or bug #, etc)

## Example: Exercise 6 checkout in powershell

```
# Call with e.g. ".\Score.ps1 -branch main"
param(
    [Parameter()]
    [String]$branch
)
# Undo local changes to whatever branch the repo was on
git restore Exercise6.cpp
# Clean previous build so suppressed build failures below don't misreport
  results
# Suppress the output by capturing it in a variable instead of printing to
  console
$mute = msbuild -target:Clean
# Download the new branch
git checkout "origin/$branch"
# Build Exercise6.cpp with visual studio
$mute = msbuild
```

## Example: Exercise 6 Testing in PowerShell

```
# Run the built code on the author's set of tests to output pass rate and
  coverage
echo "Original test run"
.\x64\Debug\SWEF22-Exercise-6.exe
# Load the instructor set of test cases
$tests = Get-Content jdkTests.cpp
# Append instructor test cases to student submission so they will be run
Add-Content Exercise6.cpp $tests -Encoding Ascii
# Clean previous build so build failures with added tests don't misreport
  results
$mute = msbuild -target:Clean
# Build the project with the added tests
$mute = msbuild
# Run the build with instructor tests added.
echo "With JDK tests"
.\x64\Debug\SWEF22-Exercise-6.exe
```

# Exercise 6: Output for Grading

```
PS C:\Users\jdk\github\SWEF22-Exercise-6>
 .\Score.ps1 -branch main
HEAD is now at aa1d201 Clean and comment build
 script
Original test run
Pass rate: 100%
Coverage: 64.7059%
With JDK tests
Pass rate: 100%
Coverage: 100%
PS C:\Users\jdk\github\SWEF22-Exercise-6>
```

## Advice on Automation

- Automate everything you can. If you type the same 50+ characters more than one a week – even if a few of them are different each time – find a way to do it in fewer. Again: efficiency, reliability, productivity.
- Learn Python, bash scripting for Linux, and AutoHotKey for Windows.
- Share the automated tools you write – This can be a huge productivity boost to the team, provide resources for new members, and increase your value to the organization. Consider an "automation" repository.
- In a similar vein, if you find yourself writing the same function in multiple programs, create or add it to a shared utilities library or SDK.
- Find out what processes colleagues in other departments might be using that could be automated and help them automate that work.