Software Development Teams

JD Kilgallin

CPSC:480

08/31/22

Pressman Ch 5, 24

The "i" in "team"

Learning Objectives

- Participants in software development and their roles
- Organization of software development teams
- Career tracks in software engineering
- Relationships to other parts of an organization
- Applying to software engineering roles

- Project 1 assigned.

Job titles in software engineering teams

- Software engineer/developer (build software as primary focus)
  - Game/web/mobile developer (design and build software in specific target domain; particular flavors of software engineer)
  - IoT/Embedded software engineer (software for connected devices)
- Product/project manager, UX engineer (design software elements)
- Quality assurance engineer (test software, build test infrastructure)
- Technical writer (Develop software documentation and guides)
- Integration engineer (writing code to interface technologies)
- Software/systems/solutions architect (typically a higher-level role responsible for design and maintenance of program structure)

Other titles for computer science graduates

- Data scientist (Using software to produce relevant business insights)
- Systems analyst (computer & software support and troubleshooting)
- Solutions engineer (develop means to solve business problems using software; technical sales role)
- Systems engineer (holistically managing computers and software)
- Database engineer/administrator (develop and maintain large dbs)
- Cloud architect, DevOps engineer, Site Reliability Engineer (Build and maintain cloud-native applications and related components)
- Computer engineer (developing hardware and system-level software)

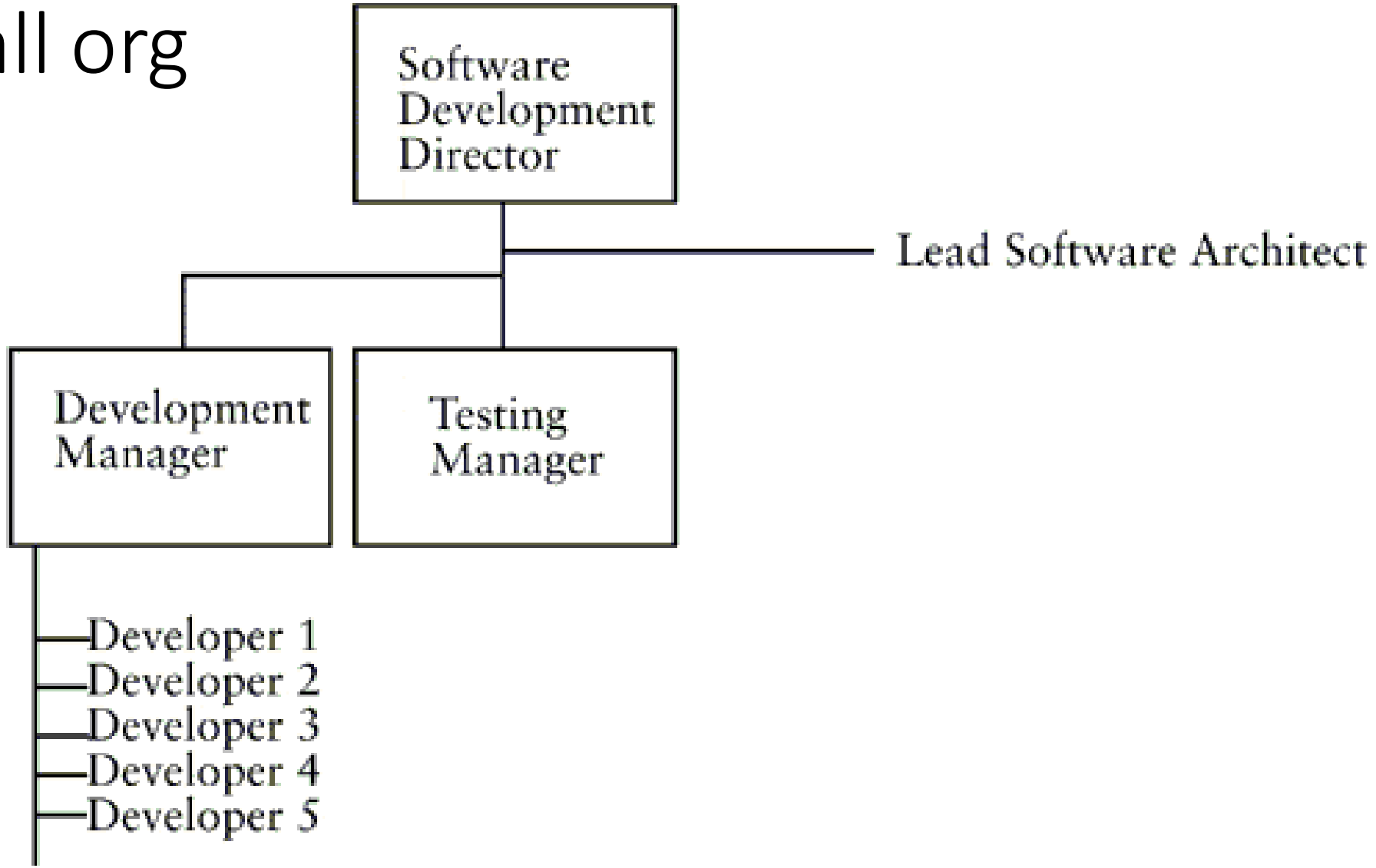
Software engineering team structure

- A functional *software development team* typically consists of a team lead, 2-10 SWE, 1-4 QA engineers, 1-3 product managers, and possibly a technical writer, integration engineer, and/or UX engineer.
- A *software development organization* is responsible for a suite of products and typically contains 1-50 development teams.
- Team leads report to a director or VP of engineering, or directly to CTO in smaller organizations. Conversely, larger organizations may have intermediate engineering managers for additional hierarchy.
- Higher-level and ancillary roles, such as software architect, product manager/owner, integration engineer, technical writer may report directly to organization leader and fall outside individual teams.

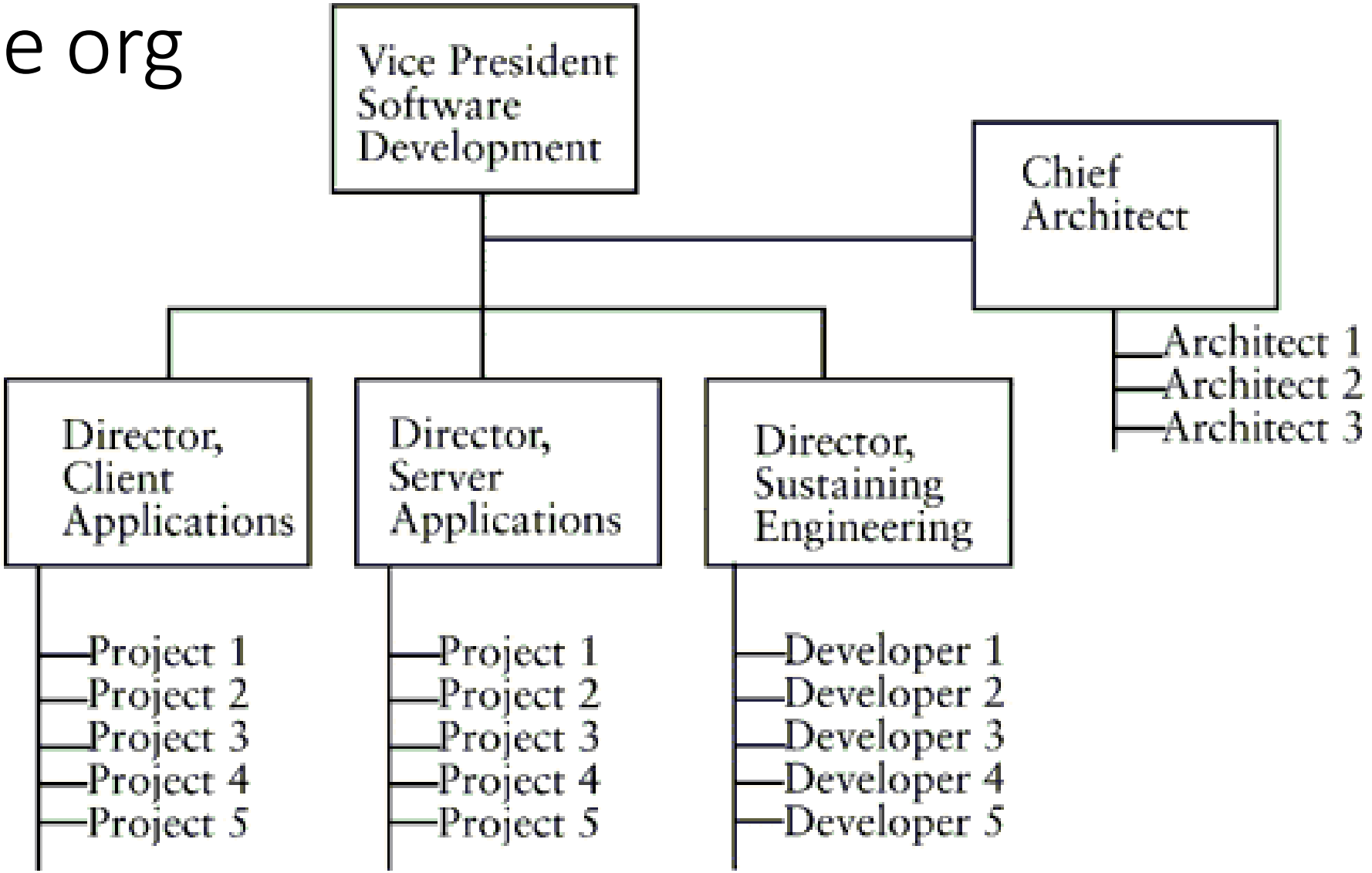
Cross-functional relationships

- Software engineers respond to escalated technical questions from sales staff, support staff, partner organizations, and end users that require deep knowledge of the product area and source code.
- Software engineers work with marketing to create blog posts, whitepapers, conference presentations, and other materials for technical audiences that require subject matter expertise and support business objectives.
- Software engineers work with technical staff at other organizations to develop software integrations using APIs and extensibility points.
- Centers of Excellence and virtual teams - Official or unofficial reporting structures to address cross-team concerns (e.g. GUI design may involve a Software Engineer, UX engineer, Product Owner, and sales and support staff with first-hand experience on common customer criticisms).

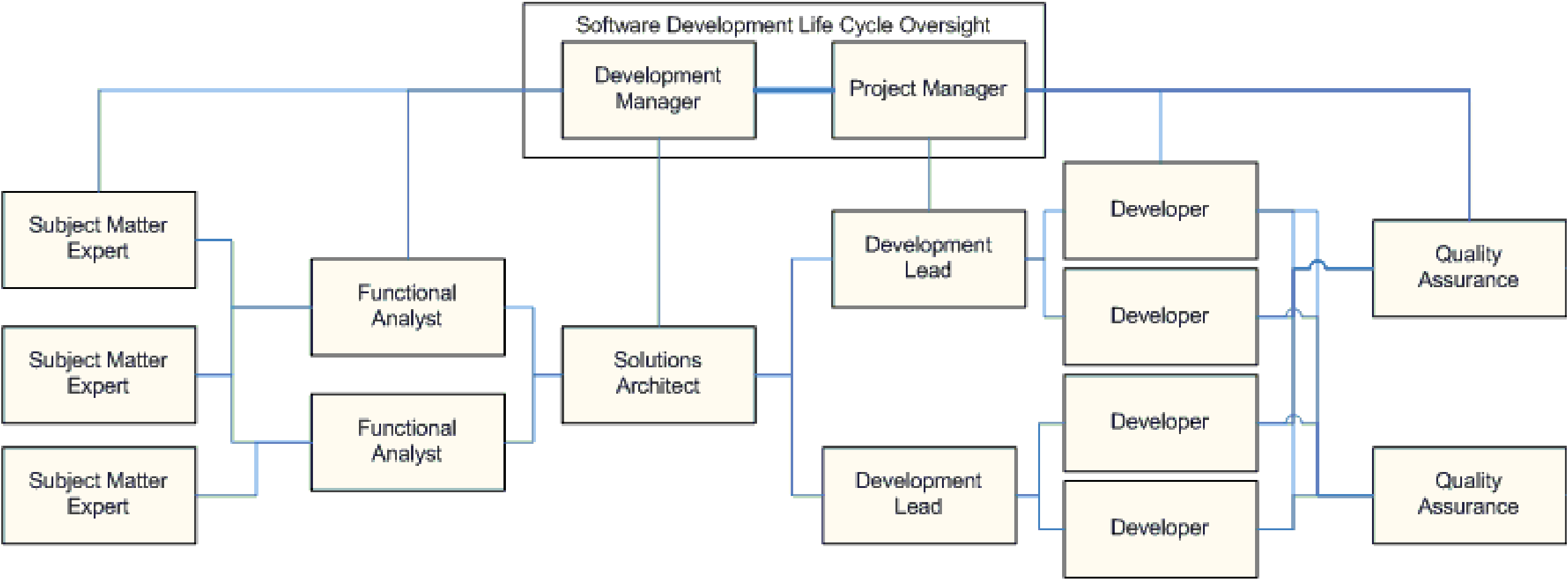
Small org


Software Development Director
Lead Software Architect
Development Manager
Testing Manager
Developer 1
Developer 2
Developer 3
Developer 4
Developer 5

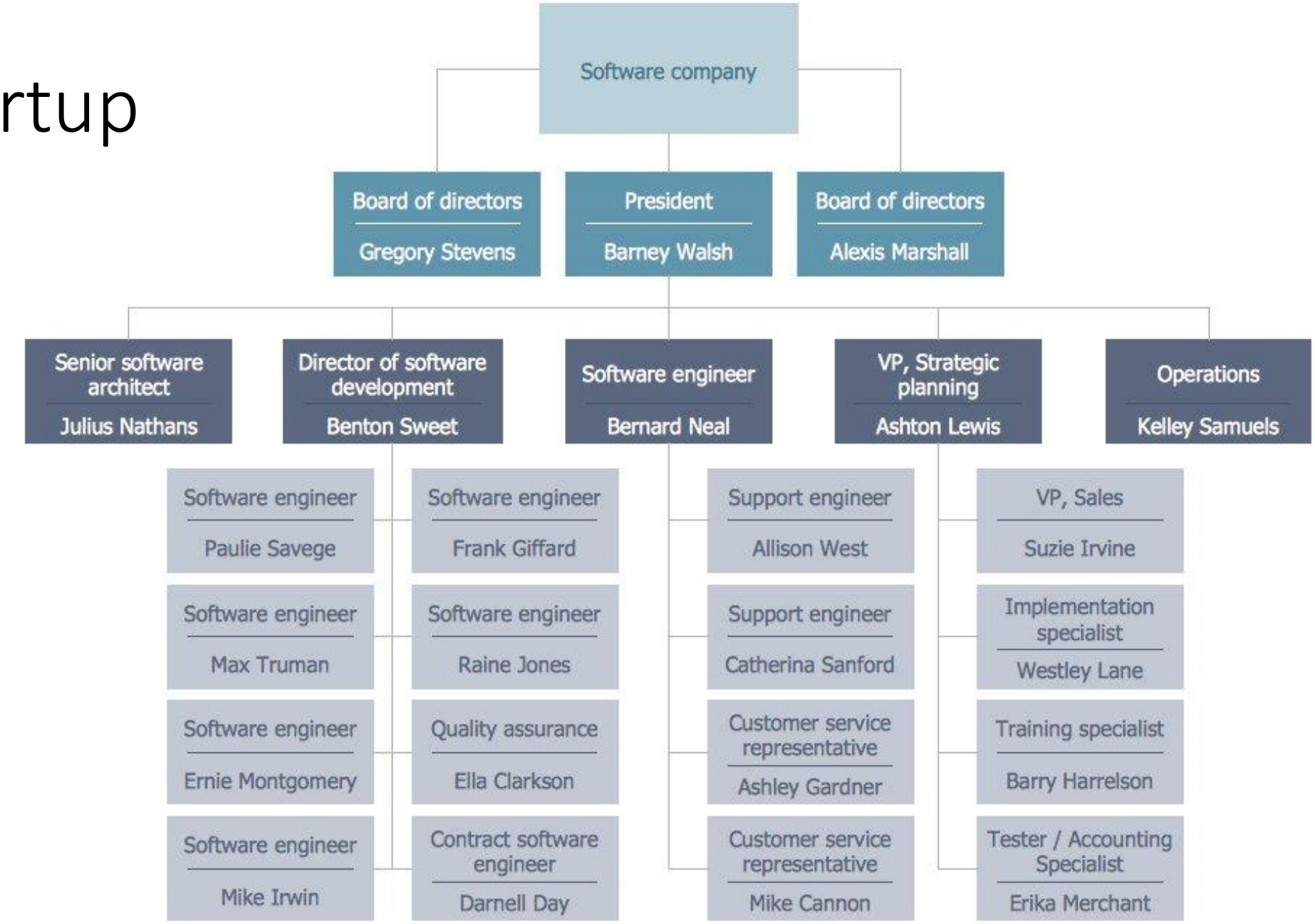
Large org


Vice President
Software
Development
Chief
Architect
Architect 1
Architect 2
Architect 3
Director,
Client
Applications
Director,
Server
Applications
Director,
Sustaining
Engineering
Project 1
Project 2
Project 3
Project 4
Project 5
Project 1
Project 2
Project 3
Project 4
Project 5
Developer 1
Developer 2
Developer 3
Developer 4
Developer 5

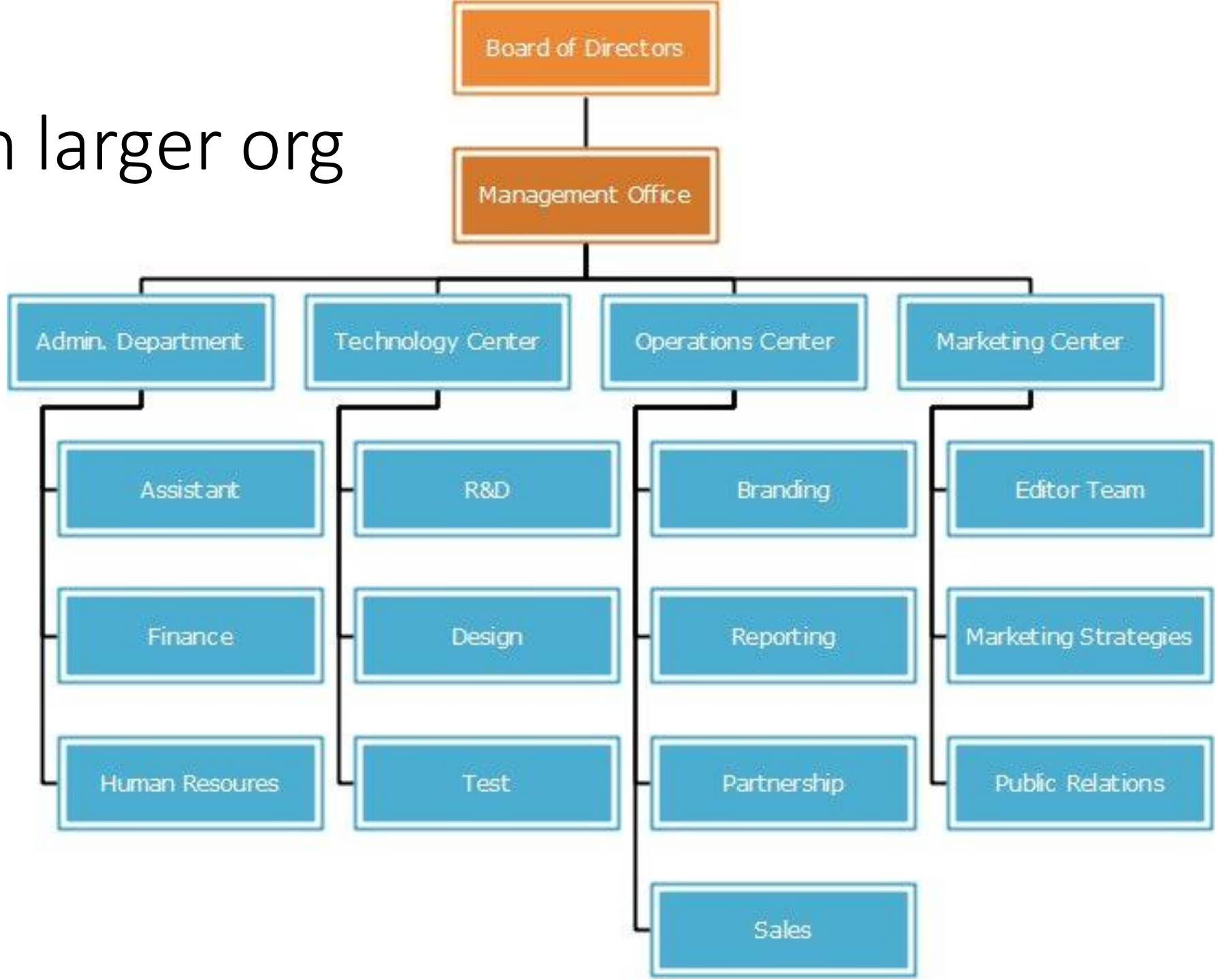
Relationships within engineering organization



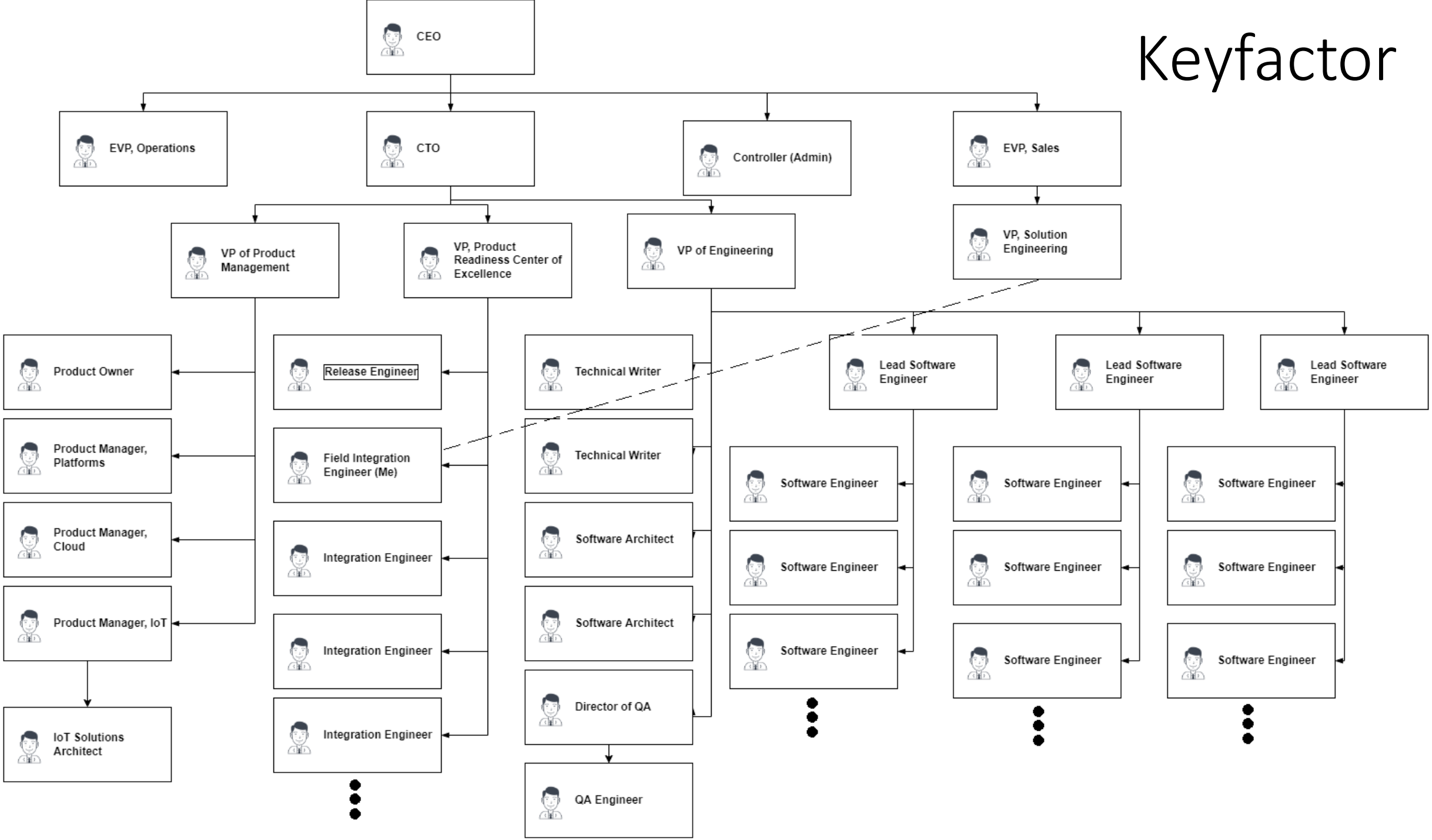
Startup


Software company
Board of directors
Gregory Stevens
President
Barney Walsh
Board of directors
Alexis Marshall
Senior software architect
Julius Nathans
Director of software development
Benton Sweet
Software engineer
Bernard Neal
VP, Strategic planning
Ashton Lewis
Operations
Kelley Samuels
Software engineer
Paulie Savege
Software engineer
Frank Giffard
Software engineer
Max Truman
Software engineer
Raine Jones
Software engineer
Ernie Montgomery
Quality assurance
Ella Clarkson
Software engineer
Mike Irwin
Contract software engineer
Darnell Day
Support engineer
Allison West
Support engineer
Catherina Sanford
Customer service representative
Ashley Gardner
Customer service representative
Mike Cannon
VP, Sales
Suzie Irvine
Implementation specialist
Westley Lane
Training specialist
Barry Harrelson
Tester / Accounting Specialist
Erika Merchant

Within larger org



Keyfactor


CEO
EVP, Operations
CTO
Controller (Admin)
EVP, Sales
VP of Product Management
VP, Product Readiness Center of Excellence
VP of Engineering
VP, Solution Engineering
Product Owner
Product Manager, Platforms
Product Manager, Cloud
Product Manager, IoT
IoT Solutions Architect
Release Engineer
Field Integration Engineer (Me)
Integration Engineer
Integration Engineer
Integration Engineer
Technical Writer
Technical Writer
Software Architect
Software Architect
Director of QA
QA Engineer
Lead Software Engineer
Lead Software Engineer
Lead Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer
Software Engineer

Advancement

- Promotion solely as a software engineer
  - Associate->[SWE]->Senior->Lead*->Principal->Staff Software Engineer
  - *"Lead Software Engineer" does not always refer to team leadership
  - "Staff SWE" – role akin to tenure as professor, title akin to "Staff Seargent". Critical projects to support the business rather than product line engineering.
- Promotion as other individual contributor role in software org
  - Software Engineer->Software Architect
  - Software Engineer->Product Manager
  - Software Engineer->Integration Engineer
- Promotion to management in software org
  - SWE->Lead->Engineering Manager->Engineering Director->VP of Engineering
- Promotion to other orgs
  - Sales/solution engineering or implementation engineering
  - Management in operations, partner enablement, or other departments

Joining a software organization

- Officially starts with submitting a résumé ("resume")
  - Building a project portfolio & well-composed resume come even before that.
- First screening likely done by automated systems and recruiters.
- Initial, informal interviews usually done by phone with 1-4 current engineers. Team may interview 5-10 candidates from a pool of 20 or so resumes provided to the team by a recruiter.
- Formal technical interviews usually done on-site and involve a few rounds of coding exercises and other skill and team-fit assessments. Likely includes 3-5 applicants from top candidates out of phone screen.
- Hiring manager likely to make an offer contingent on a background check.
- Next preferred candidate will receive an offer if the first one falls through

Resume suggestions

- Stick to conventional format *unless you're interviewing for a design role.*
- Put basic info first – name, contact, citizenship, web link
- Academic info - college, degree, dates, GPA, key courses, key projects.
- Work info - most recent first. Company, title, dates, accomplishments.
  - List non-technical work if you can afford the space, and/or if it includes a promotion or rehire (e.g. worked for Old Navy two summers in a row).
- Technical skills – programming languages, frameworks, tools, areas of focus or specialization (e.g. cryptography, databases, Android. *Not "PowerPoint"*).
- Awards, memberships, certifications (e.g. First Aid, community theater).
- Include key words that an automated resume screener will catch.
- Avoid "soft" verbs – "assisted with...", "participated in...", "worked to..."
- Stick to 1 page, and try not to make it too busy. It will get 1 minute or less.

Application suggestions

- Work with a recruiting agency like Robert Half.
- Get referrals from colleagues where you can.
- Put your resume in the University directory.
- Tailor your resume to jobs you're interested in.
- Learn about the company before you interview.
- Look up individual interviewers on LinkedIn if you can.
- Make sure your phone/email information is correct, and check them.
- Be on LinkedIn, & keep other searchable social accounts professional.
- Apply to places just for practice if you need.

General interview suggestions

- Be on time, well-groomed but in attire you'd be comfortable in daily.
- Bring copies of your resume, pen and paper. Review your resume and be prepared to talk about anything on it.
- Be prepared with questions about the job/company (two-way street).
  - "What does a typical work week look like for someone in my position?"
  - "What languages and technologies do you work with on a regular basis?"
  - "What does a typical career advancement timeline look like here?"
  - "What are the company's objectives, core values, and growth expectations?"
  - "What led you to choose this company and its culture over others?"
- If you don't know something you're asked, explain how you'd learn.
- Ask for business cards from the interviewers.
- Smile, and remember that everyone gets nervous interviewing.

Technical interview suggestions

- Do a couple of practice problems (Leetcode) in your preferred language a day or two before you interview, on PC *and* whiteboard.
- Describe your thought process as you solve the problem. Anything that would go in a thoroughly-commented file, and alternatives you reject. Use descriptive method and variable names.
- Brush up on hashing and sorting one-dimensional data structures, and look for places they can improve your solution.
- Be prepared to give a runtime (big-O) analysis of your solution, and expect follow-up questions to modify or extend the original problem.
- Ensure you fully understand the assignment; repeat it and ask questions.
- Don't be afraid to use pseudo-code when you're not using an editor. Gloss over any details you need to. You can always go back to refine, optimize, and fill in details on a second pass. Leave space between lines to do this.

Take-home coding exercises

- Some companies will give a longer coding project to be completed at home. May use an online tool or ask you to submit code by email.
- Usually language is not dictated unless it's a specific job requirement. Unlike whiteboard exercises though, it probably needs to be real code at a minimum. It's possible it will require more than one language.
- Make sure your code is *very* well-commented and follows good coding style – proper variable names, indentation, everything.
- It's okay if the solution isn't "production-grade", but note where improvements should be made (security, performance, exception handling)
- It's usually fine/encouraged to solve a slightly more general or advanced problem explaining how it solves the original case. You can also stub out or put comments explaining where functionality could be extended.

Post-interview suggestions

- Follow up thanking the technical interviewers for the time and listing a couple reasons why you feel like the *position* is a good fit for *you* (e.g. "I was very interested in what you said about opportunities to work directly with customers" or "I was happy to learn that your company offers gym memberships, as my health is important to me.")
- If there were any questions you deferred on or if you discover errors in your problem solutions, share what you found.
- Be patient (remember you may eventually get an offer even if you were the 2nd or 3rd best candidate).
- If your salary offer seems low, ask something like "Could I expect a raise within a year after proving my value to the company?" Working with a recruiting agency is likely to get you a better, more accurate offer.

Onboarding suggestions

- You'll start with standard HR and IT setup and introduction to the company and team (trainings on security, workplace harassment, etc)
- Your manager will meet with you to review expectations and provide resources for getting up to speed.
- If your manager doesn't assign a specific member of the team as a new-hire mentor, ask for one. This is extraordinarily helpful.
- Setting up an initial development environment will take a day or two. Document *everything* as you go. Ideally the process is documented already, but there will always be gaps or things out of date. This lets you provide immediate value to the team.
- It will take 1-3 months to fully get up to speed with the software.

Example technical interview question

- Suppose an interviewer asks "Given an array of integers and a target sum, determine if the array contains two integers that add up to the target sum"
- Ask a few questions to clarify the problem: (here we'll assume "Yes" for all)
  - Exactly two of the integers in the array that sum to the target?
  - Can I assume that adding any two of the integers doesn't cause overflow?
  - Do the numbers in the array need to be distinct? If one element in the array is half the target sum, adding it to itself doesn't satisfy the condition?
  - No assumptions about the array; It can contain duplicates, negatives, etc?
- Start with method signature "bool canSumTo(int[] addends, int sum)" and a one-line comment above "returns TRUE if there are distinct i, j such that addends[i] + addends[j] == sum, and FALSE if not"

Example technical interview v1

- Say "The easy solution is a double-loop through the array, checking each pair of numbers to see if they add up to the target. If we find a pair that does, stop and return true. If we get through all pairs and none of them do, return false at the end. That runs in $O(n^2)$; let me see if we can do better."
- Code would start something like this (may not actually write it):

```
bool canSumTo(int[] addends, int sum) {
    for(int i = 0; i < addends.length()-1; i++)
        for(int j = i+1; j < addends.length(); j++)
            if (addends[i] + addends[j] == sum) return true;
    return false;
}
```

Example technical interview v2

- "What happens if we sort the array first? Maybe we can improve the inner loop. First that would let us stop the loop early."

```
bool canSumTo(int[] addends, int sum) {
    addends = Array.sort(addends);
    for(int i = 0; i < addends.length()-1; i++)
        for(int j = i+1; j < addends.length(); j++) {
            if (addends[i] + addends[j] == sum) return true;
            if (addends[i] + addends[j] > sum) break;
        }
    return false;
}
```

Example technical interview v2.1

- "Actually if the array is sorted, we can use binary search on the inner loop to see if it has the other number we need, and that brings it down to O(nlog(n))"

```
bool canSumTo(int[] addends, int sum) {
    addends = Array.sort(addends);
    for(int i = 0; i < addends.length()-1; i++) {
        int diff = sum - addends[i];
        if (binarySearch(addends.sub(i+1), diff) >= 0) return true;
    }
    return false;
}
```

Example technical interview v3

- "If we could do that inner lookup in O(1), then the whole thing would run in O(n), and it wouldn't be possible to do it faster than that. Putting the array into a hash table would let us do that."

```
bool canSumTo(int[] addends, int sum) {
    HashTable<int> addendMap = Array.hash(addends);
    for(int i = 0; i < addends.length()-1; i++) {
        if(addendMap.contains(sum - addends[i]) return true;
    }
    return false;
}
```

Example technical interview v3.1

- "Oh but now it will return true if a number added to itself equals the sum, so we need to track whether there's a *different* copy of that number in the array. Really all we need to know is if there are 2 or more copies of the number when it's half the sum."

```
HashMap<int, int> hashCount(int[] input) {
    HashMap<int, int> result = new HashMap<int, int>();
    for (int i = 0; i < input.length(); i++) {
        if (result.ContainsKey(input[i]) result[input[i]] = result[input[i]] + 1;
        else result[input[i]] = 1;
    }
    return result;
}
```

Example technical interview v3.1 pt 2

- "Then we can use this map and handle that case separately, and this still runs in O(n)."

```
bool canSumTo(int[] addends, int sum) {
    HashMap<int, int> addendMap = hashCount(addends);
    for(int i = 0; i < addends.length()-1; i++) {
        if(addends[i] * 2 == sum) {
            if (addendMap[addends[i]] >= 2) return true;
        }
        else if(addendMap.containsKey(sum - addends[i]) return true;
    }
    return false;
}
```

Example technical interview II

- What if we change the problem now and ask for the indexes in the array of the numbers that add up to the target sum?
- Ask a couple more clarification questions (again assume "yes")
  - How should it return the indexes? Is a two-element array okay?
  - What should it return if there aren't two elements? Is null okay?
- "I think we can extend this solution easily by tracking the actual indexes in the hash table and handling the case where one number is half the sum still as a special case"

Example technical interview II solution

- "So we map each element in the array to a list of indexes where that element is found"

```
HashMap<int, List<int>> hashIndexes(int[] input) {
    HashMap<int, List<int>> result = new HashMap<int, List<int>>();
    for (int i = 0; i < input.length(); i++) {
        if (!result.ContainsKey(input[i]) result[input[i]] = new List<int>();
        result[input[i]].Append(i);
    }
    return result;
}
```

Example technical interview II solution pt 2

```
int[] sumsTo(int[] addends, int sum) {
    HashMap<int, int> indexMap = hashIndexes(addends);
    for(int i = 0; i < addends.length()-1; i++) {
        if(addends[i] * 2 == sum) {
            if (indexMap[addends[i]].length() >= 2) {
                int otherIndex = indexMap[addends[i]].Remove(i)[0];
                return int[] {i, otherIndex};
            }
        }
        else if(addendMap.containsKey(sum - addends[i])
            return int[] {i, addendMap[sum - addends[i]][0]};
    }
    return null;
}
```

Example technical interview III

- What if we change the problem now and ask if there are exactly *three* numbers in the array that add up to the target sum?
- "I think we can extend this solution by putting all the *pairs* of numbers in the hashMap, where the key is the sum of the two numbers in the pair and the value is a list of the index-pairs that can be used to add up to the key value."
- "Then we can loop through in the main function and look in the hashMap to see if it contains the other two numbers we need, checking that neither of them are the element we're looking at. Then we can return an array of the three indexes that were used."
- What if we want to find if two *or* three numbers add up?
  - Have the hashMap contain the individual elements and the pairs both.

References

- Software Development: Building Reliable Systems. Marc Hamilton. Apr 1999. Prentice Hall.
- Tech Company Organizational Structure and You. Apr 2020. New York City Voices
- Success Stories of Software Engineer [sic]. Rich A. Boss. Apr 2016.
- 25 Sample Orgcharts. Concept Draw.
- LeetCode - The World's Leading Online Programming Learning Platform
- Software Engineering Interviews. JD Kilgallin. Apr 2014. University of Akron.
- Binary search, hash table, asymptotic complexity (big-O)
- *Reading for next lecture:*
  - Hello World - GitHub Docs
  - Set up Git - GitHub Docs

Project 1 requirements

- Part 1
  - Create a resume in Word, LaTeX, etc. My template is available as an option.
  - Curate a project portfolio as applicable. Do not include common course projects, but if you chose and solved a problem yourself, you may use that.
- Part 2
  - Identify 2 employers' open positions and report on how/why you'd apply.
  - Include the job description and a brief summary of the company's software.
- Part 3
  - Create a GitHub account (if not already created) and a personal repository.
  - Upload resume and portfolio to GitHub (single "mono-repo" or 1/project).
  - Create a README in markdown for personal repo with a brief introduction.
  - Create a release in personal repository with a pdf copy of your resume.

Project 1 details

- Submit part 2 through BrightSpace.
- Submit part 3 (covers part 1) by making the repository(ies) public or granting instructor (@Kilgallin) permissions to view repository.
  Put link in BrightSpace and a brief summary of structure/contents.
- Privacy mechanisms – You do not need to share or publish personal details that you do not want to (GPA, phone number, etc). You may create a GitHub account just for this, keep the repository private, redact info or even fake it, but it is encouraged to be transparent and authentic as you would with a recruiter. The hope is that you will be able to actually use this in a job search.
- If you do not have any suitable portfolio projects, contact instructor.

Project 1 grading

- 10% Resume – Appropriateness and formatting. Content not graded.
- 30% Portfolio – Existence of at least one intelligible project.
- 20% Report – Completeness and suitability for target positions.
  - 2% Extra credit – Include suitable cover letters for applications to these roles.
- 10% GitHub – Account and one or more reasonable repositories.
- 10% README – Suitable markdown presentation and formatting.
- 20% Instructions followed – Permissions, PDF release, BrightSpace
- Due Sunday, Sept 18, 11:59 PM: Repository commits + BrightSpace
- Maximum grade decreases by 1%/10 minutes until 4:39 PM Monday. No unexcused submissions will be accepted beyond 4:40 PM Monday.
- 1% extra credit if last submission by Saturday, Sept 17, 11:59 PM.