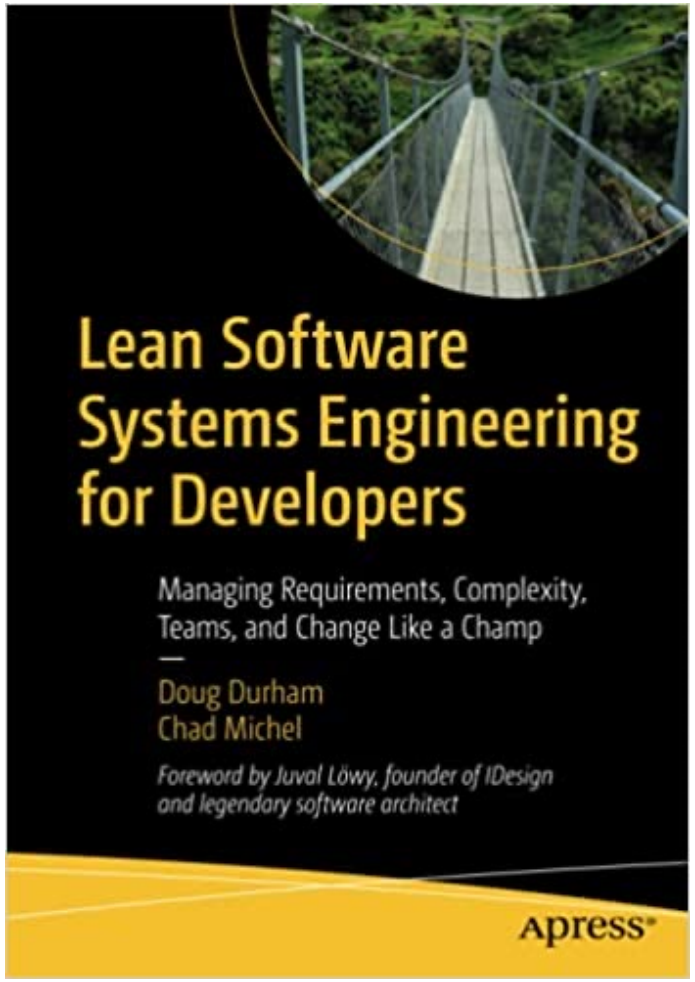
Musings on developer maturity and growth

Why HDC is important to me

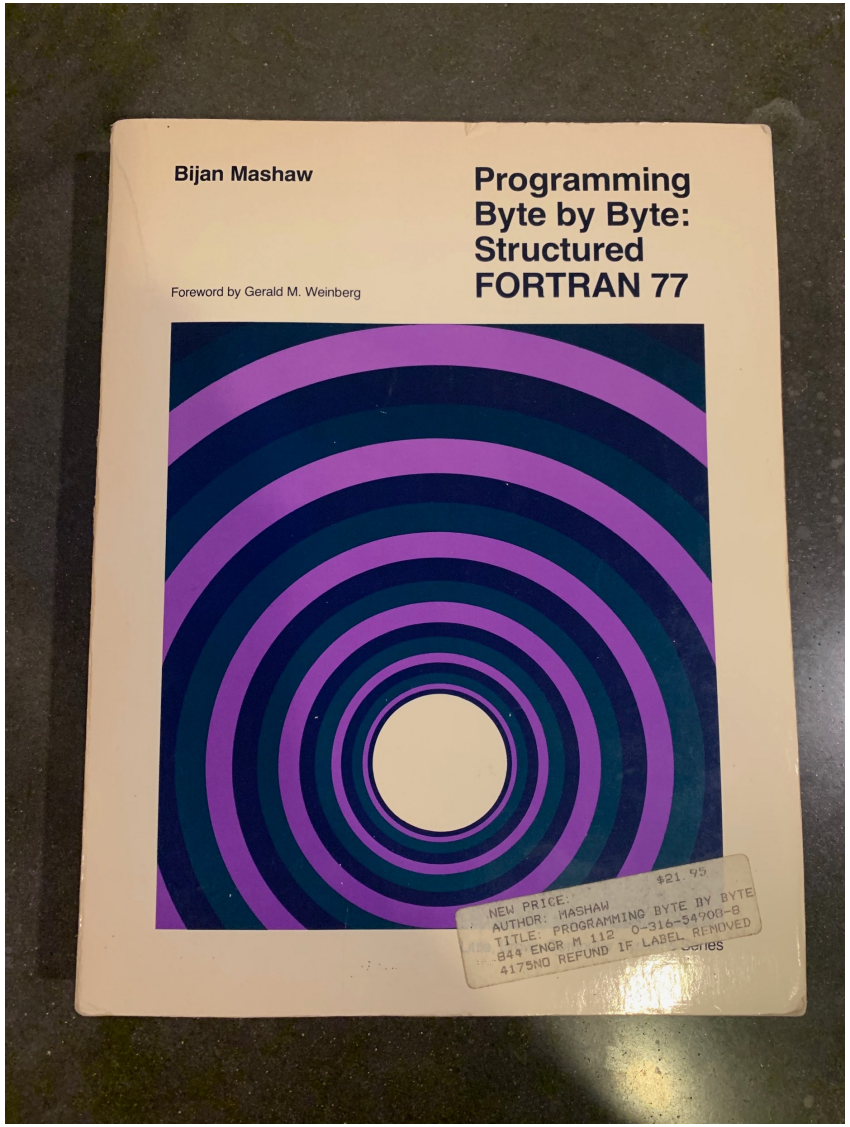
- Opportunities to learn
- Opportunities to connect
- Opportunities to meet new people from a broad region
- Opportunities to do something I have not done before…

- We need HDC



DON'T PANIC LABS

My software education…



What I have learned

- I have…
  - … actively identified the gaps in my knowledge and skills
  - … collected and studied a number of resources to fill those gaps
  - … been able to apply what I have learned
  - … grown to understand the breadth of Software Engineering and how it relates to other forms of engineering (especially systems engineering)

- Feeling like an impostor early on made me a better learner

- Strong desire to practice software development as an engineering discipline
- People's growth and career path is impacted by their "grasp" of what we are trying to do

Musings

- What are the stages of developer maturation?
- What path should they follow in this process?

Attributes that differentiate the best engineers I have met

The best developers I have worked with…

- … profoundly learn from experience
  - They have suffered pain and take active steps to avoid it
  - They recognize that a simpler solution now will benefit them down the road
  - They are often the best at “seeing around corners”
  - The recognize that tradeoffs are the norm and absolutes are rare

DON'T PANIC LABS

The best developers I have worked with…

- … invest in themselves
  - They recognize limits to their knowledge and continuously work to overcome them
  - Their skills and knowledge transcend technologies, platforms, and programming languages

The best developers I have worked with…

- … understand the importance of repeatable processes that produce results
  - Are more focused on the how than the what, or what with
  - They are not dogmatic, but they do have strong opinions that are loosely held
  - They understand the importance of process and discipline but won't let ineffective process get in the way of progress

DON'T PANIC LABS

The best developers I have worked with…

- … are obsessed with quality
  - They are often the best testers
  - They understand the long-term value and importance of good design
  - The are more strategic than tactical in their thinking

DON'T PANIC LABS

The best developers I have worked with…

- … understand that software development is a team sport
  - Software teams require strong leadership just as a sports team requires a head coach

So, what might a path look like for each of us?

Software Engineering

- We need to have a shared definition
- We need to have a shared vision for what the end game looks like
- We need a knowledge/skills map

Software engineering is…

… “the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.”

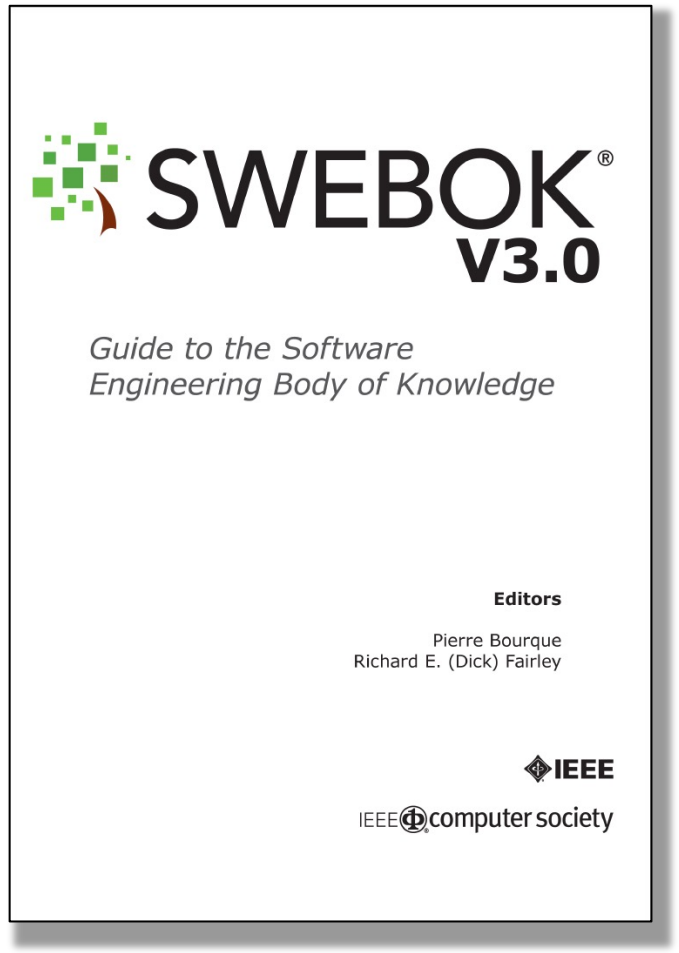
Source: ISO/IEC/IEEE Systems and Software Engineering Vocabulary and SWEBOK

What It Will Mean for Software Development to Be an Engineering Discipline

- Today:
  - We have a long a history of relying on virtuoso performance and/or death marches to achieve success, even when dealing with complex systems that are not necessarily novel or innovative.
- The Future(?):
  - We are guided by sufficiently broad and accepted knowledge, patterns, and practices that have become established and widely known.
  - Solutions to common problems are shared in a way that they can be reused to solve similar problems.
  - This collective knowledge and the ability to effectively apply it allow the average engineer to successfully solve complex problems routinely and predictably.
  - The need for exceptional engineers and talent is necessary only for those new problems that require innovation.

Knowledge/Skills Map

- Guide to the Software Engineering Body of Knowledge (SWEBOK)
- https://www.computer.org/education/bodies-of-knowledge/software-engineering



SWEBOK Objectives

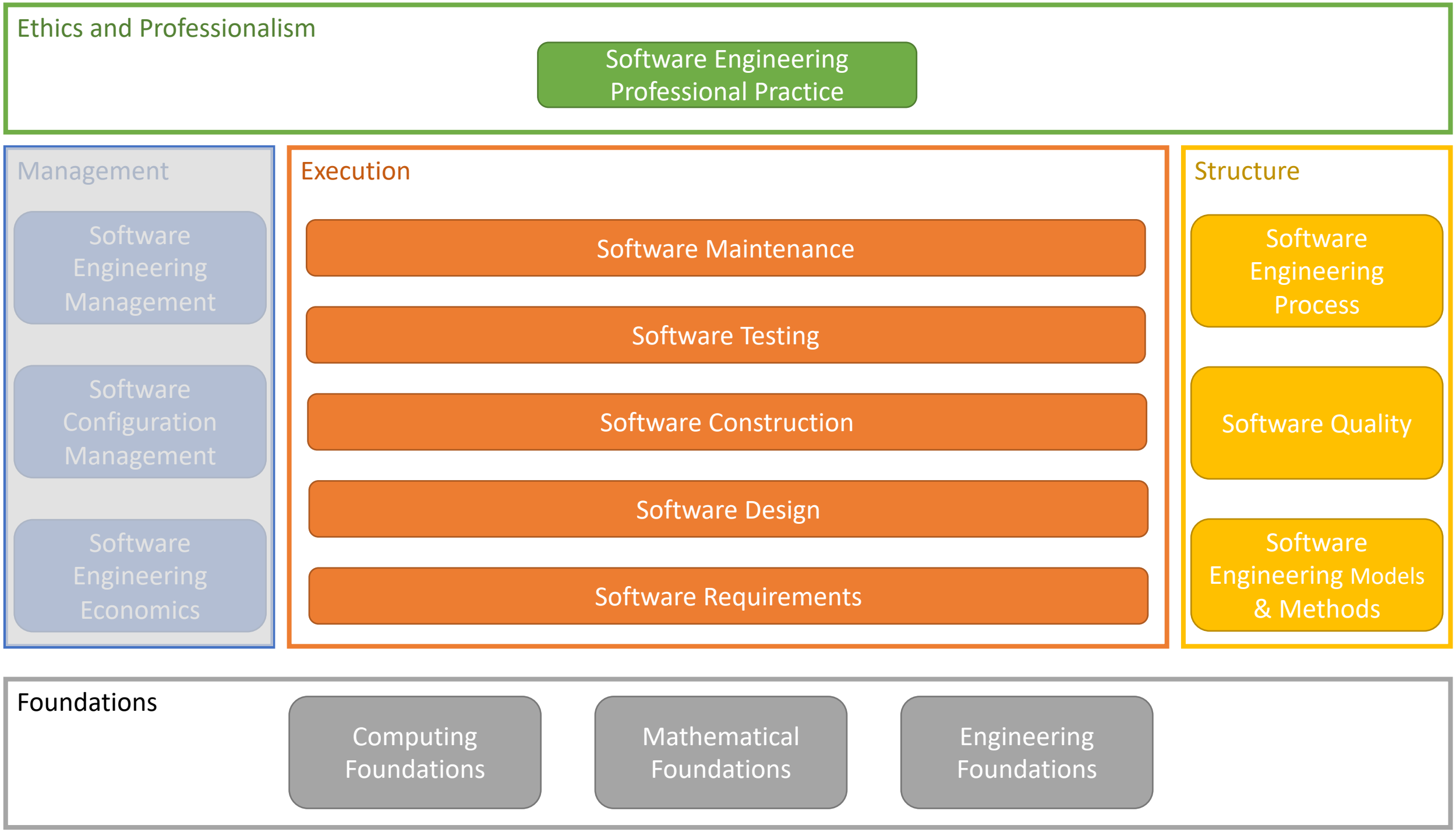
- To promote a consistent view of software engineering worldwide
- To characterize the contents of the software engineering discipline
- To provide a topical access to the Software Engineering Body of Knowledge
- To provide a foundation for curriculum development and for individual certification and licensing material

SWEBOK v3.0 Knowledge Areas

- Software Requirements
  - Defining and managing the “what”
- Software Design
  - Translating requirements to a complete design of system
- Software Construction
  - The use of coding, verification, testing, and debugging to create working software
- Software Testing
  - Structured methods for program verification
- Software Maintenance
  - Activities to effectively manage defects, enhancements, and environment changes
- Software Configuration Management
  - Traceability and management of system configuration over time
- S/W Engineering Management
  - Project management
- S/W Engineering Process
  - Work activities related to SDLC and process assessment and improvement

SWEBOK v3.0 Knowledge Areas

- S/W Engineering Models & Methods
  - Systematic and structured models and methods for developing S/W
- Software Quality
  - Defining and measuring quality
- S/W Engineering Professional Practice
  - How to practice as a true professional
- S/W Engineering Economics
  - Connecting software engineering decisions to business decisions
- Computing Foundations
  - Computer Science BoK
- Mathematical Foundations
  - Math concepts related to logic and reasoning
- Engineering Foundations
  - Foundational skills and techniques common to all engineering disciplines


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

DON'T PANIC LABS

Software Engineering Maturity “Personas”

- Caveat:
  - For illustrative purposes only
  - Reflects my experience
  - Boiling down a continuous developer maturity spectrum into a small group of discrete personas may make it difficult to identify where one fits

- Skills, Blind Spots, Differentiators, Areas of Focus

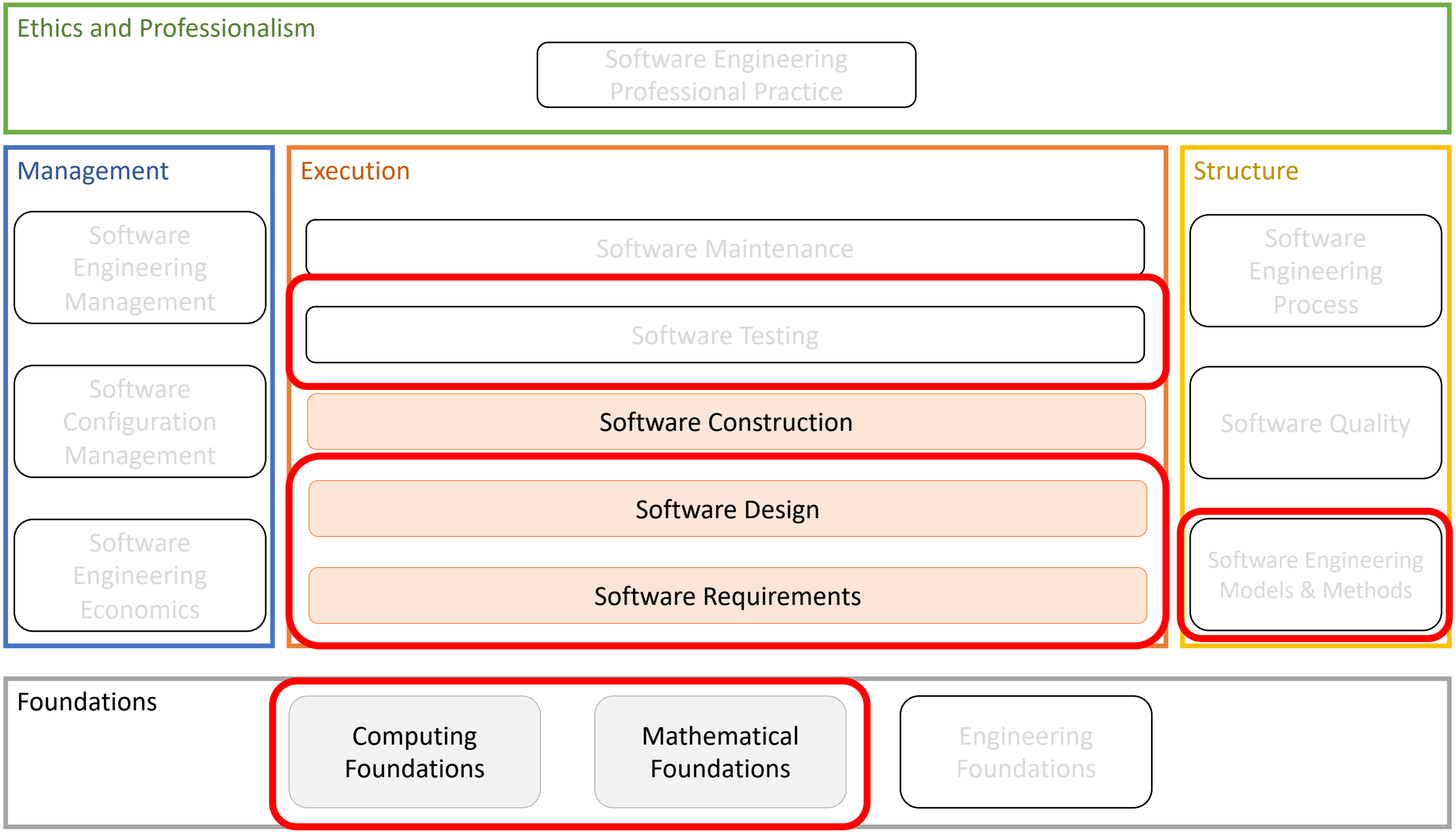
“Entry-Level Developer”

- Typical education level
  - Code school and online/books
  - CS or CE bachelors/associates
- Typical skills
  - Website development with some basic back-end
  - Familiarity with modern development environments
  - Some exposure to agile methods

- Typical blind spots
  - Not much experience building actual software applications
  - Prone to errors in judgment
  - Little/no real software design skills/knowledge
  - Little testing experience
  - Gaps in CS, Math, and Engineering foundations

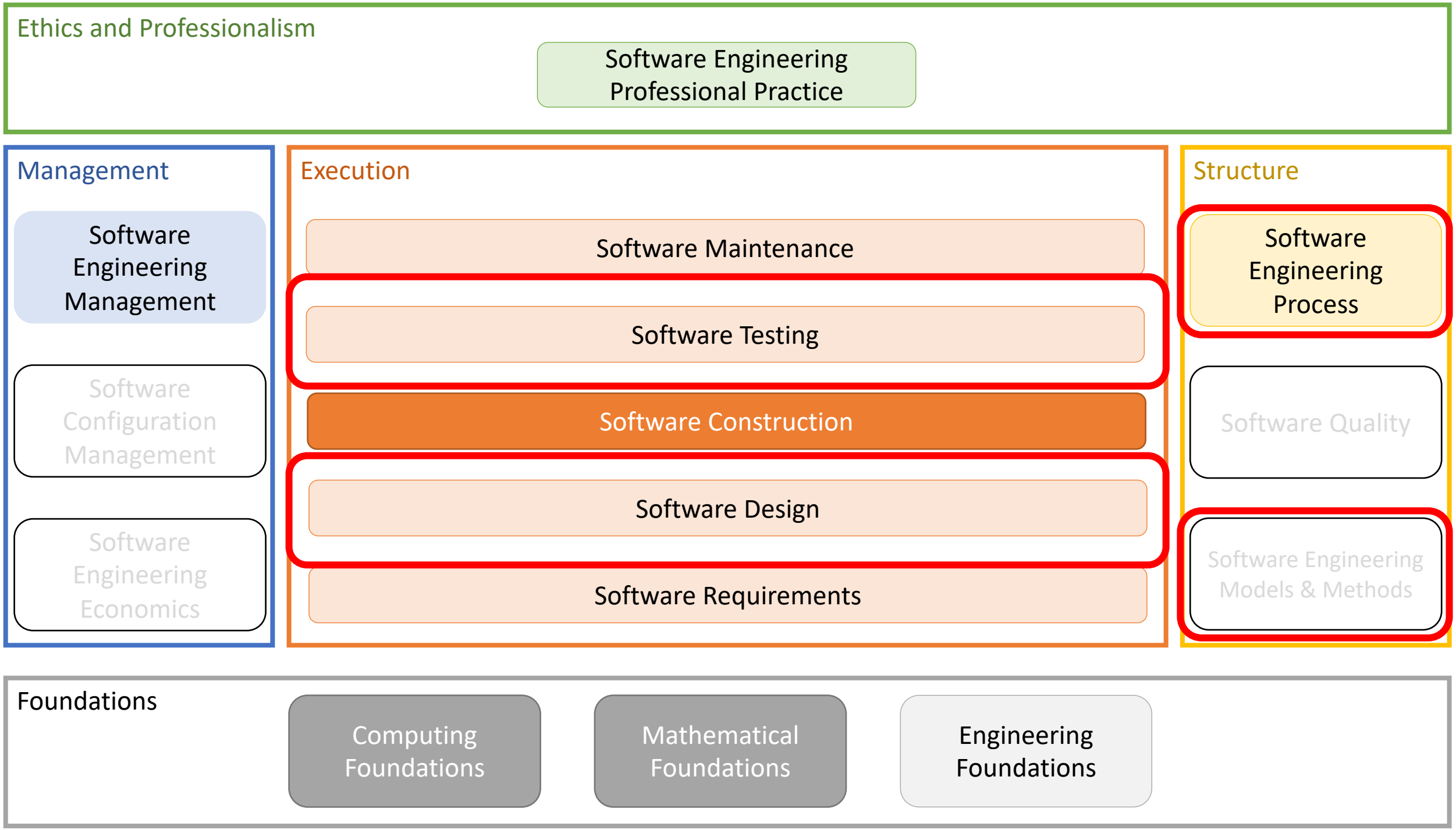
- Attributes that differentiate the top performers
  - Internships with relevant real-world experience
  - Strong desire to expand the breadth and depth of their knowledge
  - Aren’t afraid to ask for help

- Suggested areas of focus
  - Shore up CS and Math foundational knowledge
  - Software Design
  - Software Testing
  - Software Requirements
  - Software Engineering Models & Methods


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

“Experienced Developer”

- Typical new/emerging skills
  - Strong software development skills across technologies
  - Emerging analysis and critical thinking skills
  - Beginning recognition of the role of professionalism
  - Understanding of the value of engineering management processes, models, and methods
  - Effective in production system support
- Typical blind spots
  - Can follow the patterns but might not be able to explain the why
  - Errors in judgment still a concern
  - Understanding of what makes a design good vs bad
- Attributes that differentiate the top performers
  - Just enough impostor syndrome to consistently seek validation and feedback on decisions
  - Organized and deliberate about continuous learning
  - Increased focus on quality and testing
- Suggested areas of focus
  - Software Design
  - Software Testing
  - Software Engineering Models & Methods
  - Software Engineering Process


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

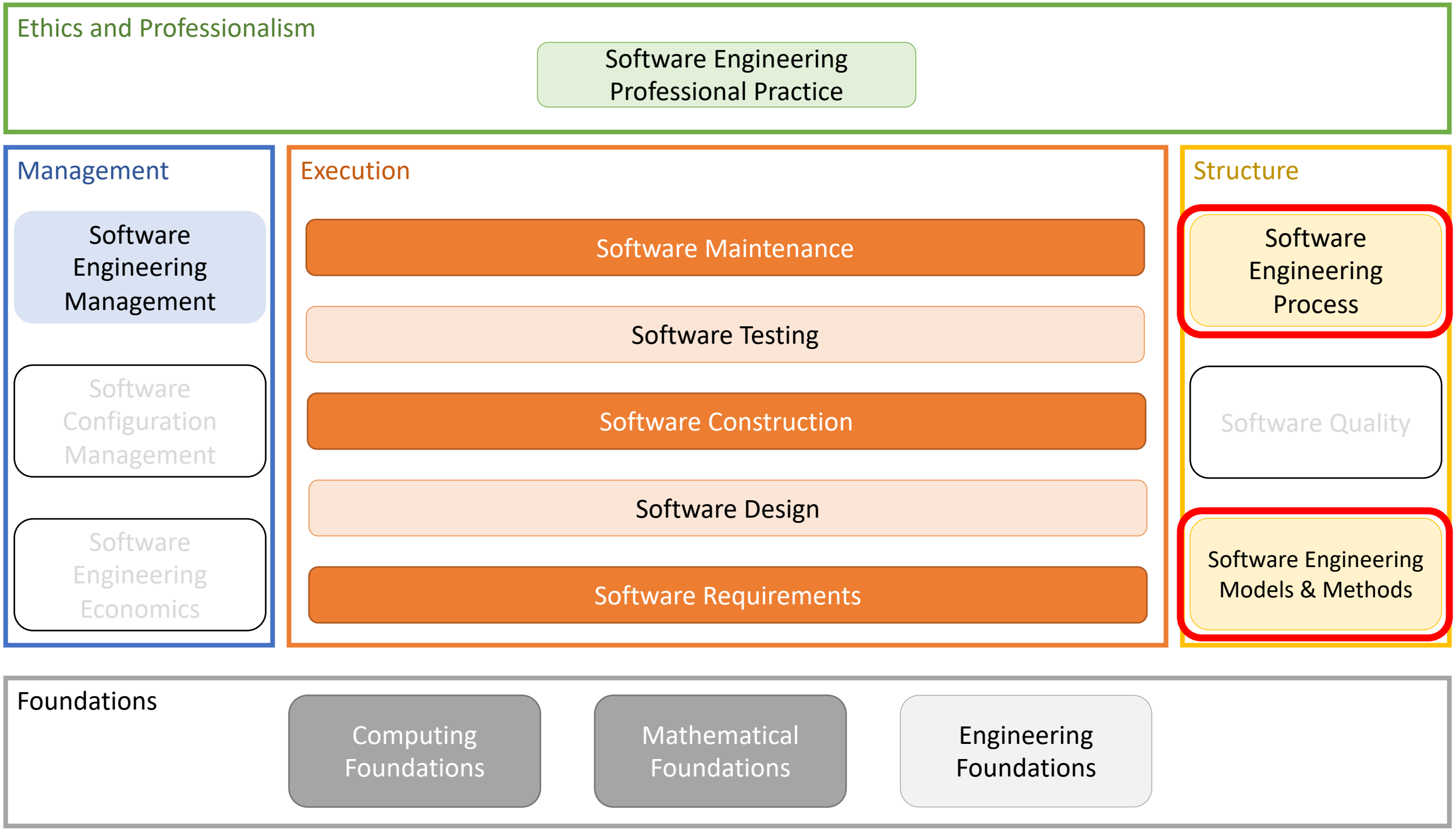
“Emerging Leader”

- Typical new/emerging skills
  - Ability to effectively advocate patterns and best practices
  - More pronounced ownership over the quality of requirements
  - Recognition of the need for “the application of a systematic, disciplined, quantifiable approach…”
  - Able to participate and contribute to architectural and detailed design
  - Increased empathy for the customer
  - Can effectively apply patterns and best practices to code reviews
  - Errors in judgment are minimal

- Typical blind spots
  - Testing expertise is still emerging
  - Design expertise still emerging
  - Often prioritizes responsiveness over “leave nothing to chance”

- Attributes that differentiate the top performers
  - Effective at coaching others
  - Cares deeply about developing a deep understanding of software design
  - Consistently prioritizes effective testing
  - They seek out mentors to help them on their journey

- Suggested areas of focus
  - Software Engineering Models & Methods
  - Software Engineering Process


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

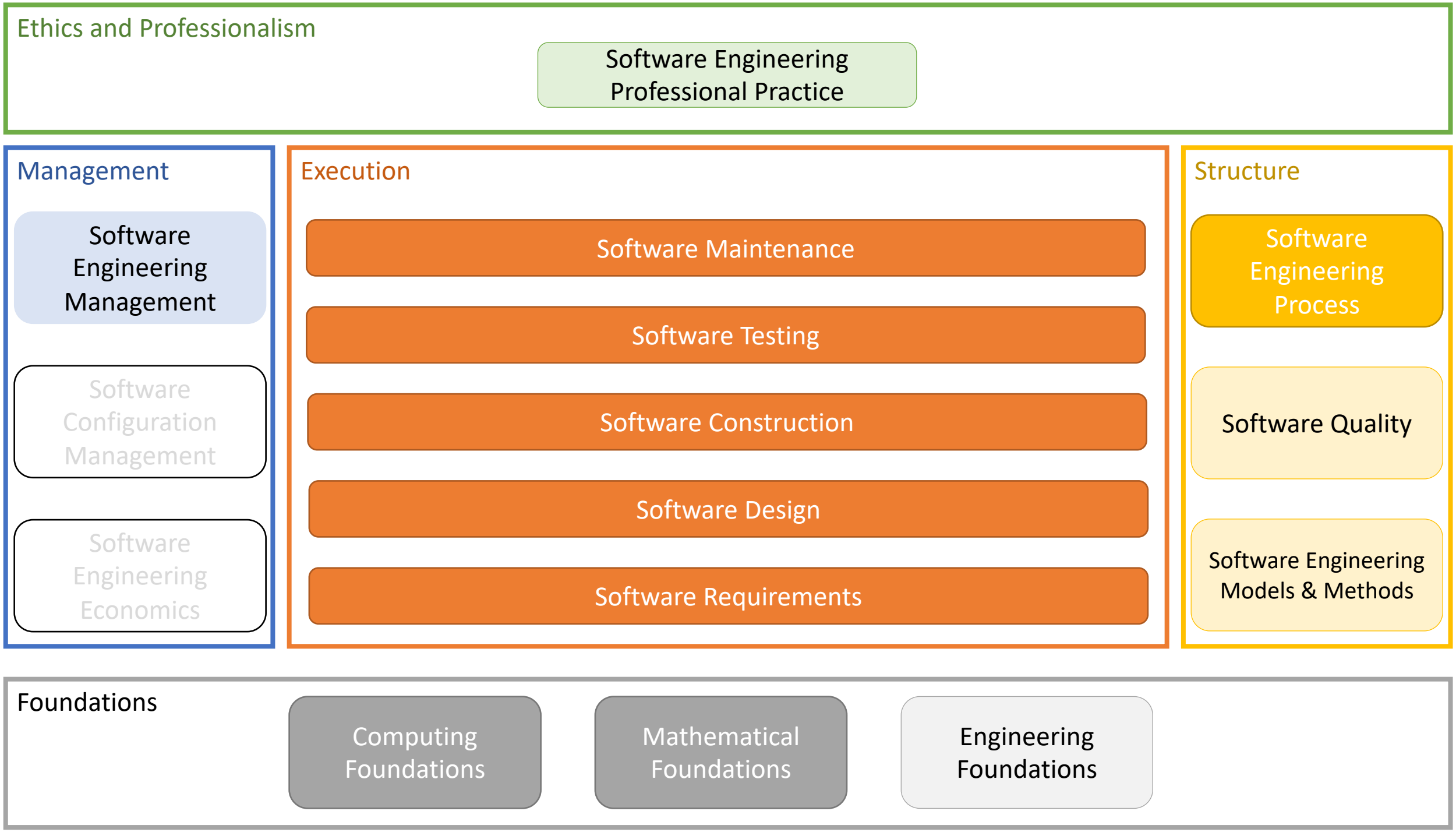
“Software Engineer”

- Typical skills
  - Can consistently lead projects to a successful outcome
  - Tends towards “leaving nothing to chance”
  - Effective at identifying the most effective ways to test a system
  - Always looking for ways to improve outcomes, processes, and methods
  - Thinks in terms of “engineering” software

- Typical blind spots
  - Knowledge of other processes, models, and methods
  - The importance and value of metrics

- Attributes that differentiate the top performers
  - Ability to identify and incorporate proven patterns, methods, and processes
  - Ability to teach as well as do

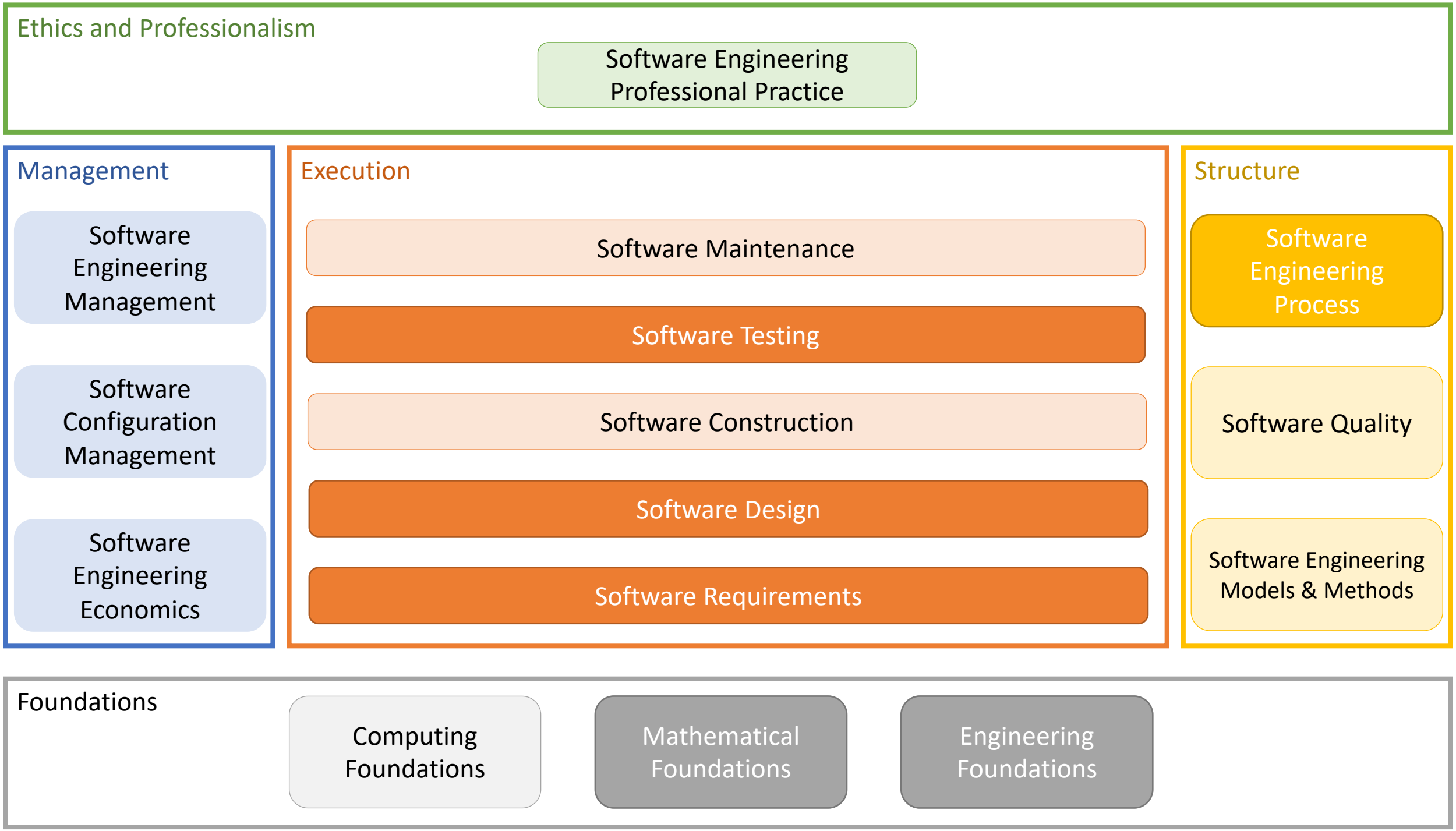
- Suggested areas of focus
  - Study systems engineering and other engineering disciplines


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

So, what should I take from this?

Do a self-assessment

- Use this knowledge map (and the SWEBOK) to make an honest assessment of where you are in your journey
- Be deliberate about your growth and education and track your journey


Ethics and Professionalism
Software Engineering Professional Practice
Management
Software Engineering Management
Software Configuration Management
Software Engineering Economics
Execution
Software Maintenance
Software Testing
Software Construction
Software Design
Software Requirements
Structure
Software Engineering Process
Software Quality
Software Engineering Models & Methods
Foundations
Computing Foundations
Mathematical Foundations
Engineering Foundations

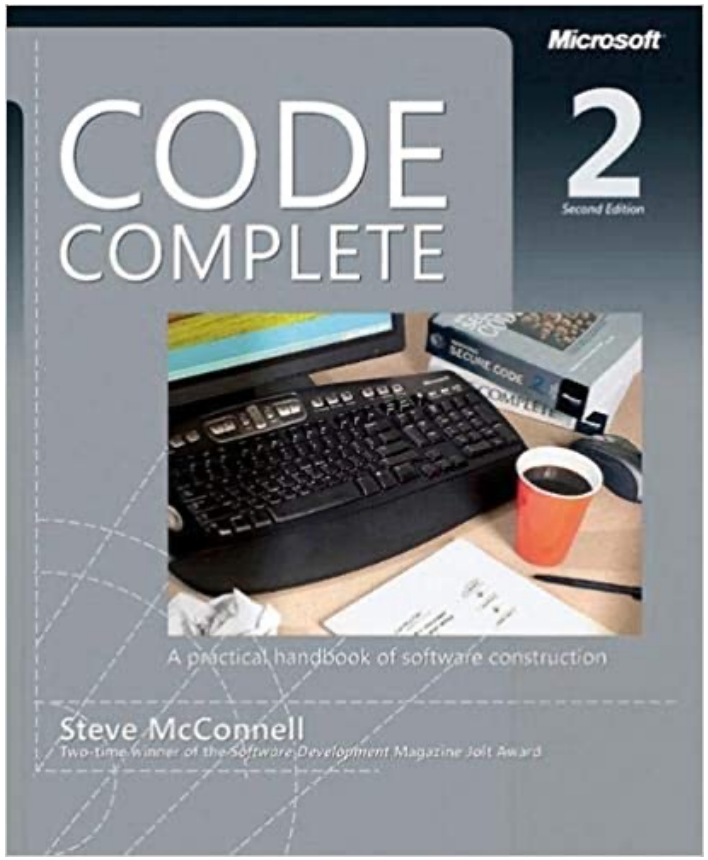
If you aspire to continue to mature and grow, you should recognize…

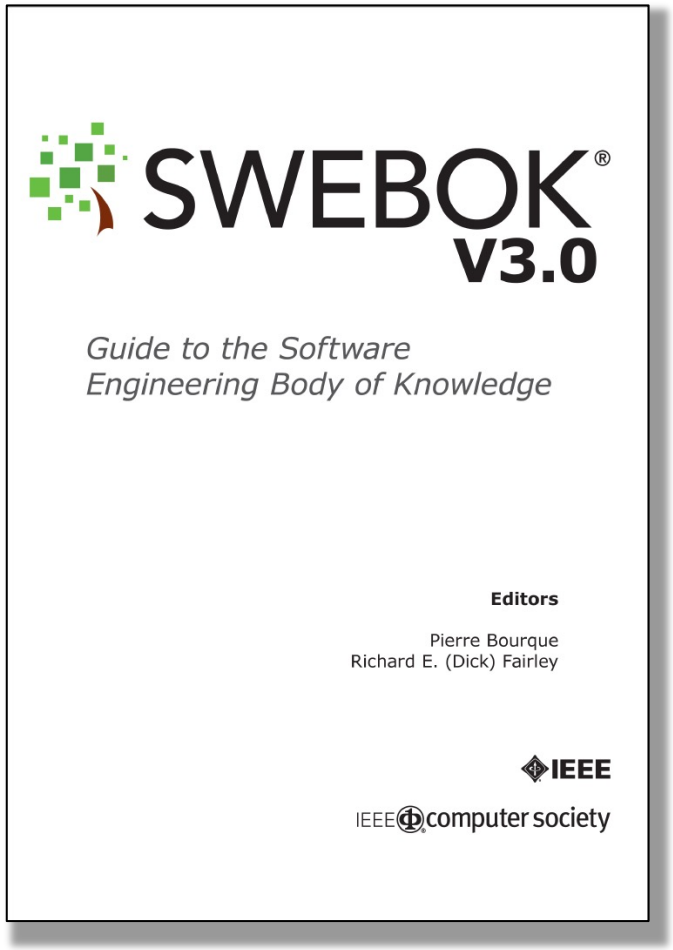
- … traditional education systems will not give you everything you need
- … experience in your current role will not give you everything you need
- … being the best programmer does not make you an engineer
- … your progress will be dictated by your efforts and where you choose to work
- … it is important for organizations to have a system for software development that enables putting this learning into practice

I highly recommend you…

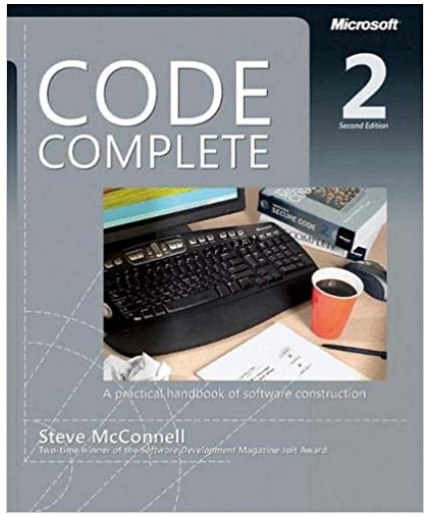
- … identify “mentors” who can help provide guidance and coaching
- … leverage the abundance of resources out there to help with your gaps

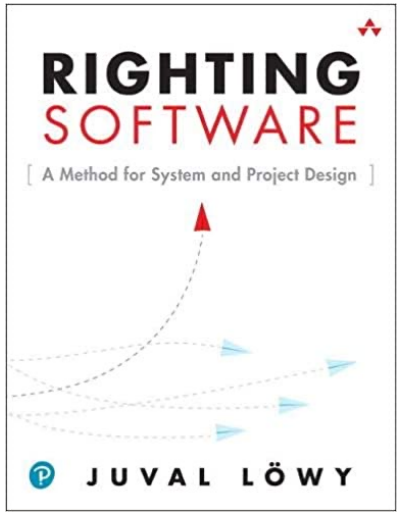
Software Engineering “Index”

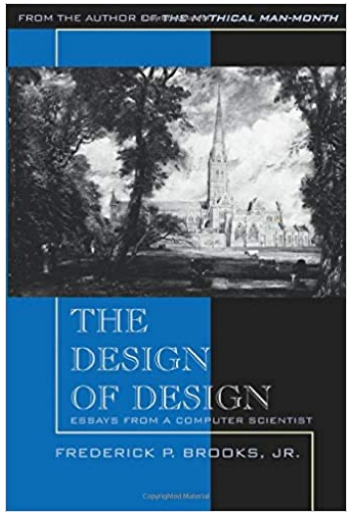


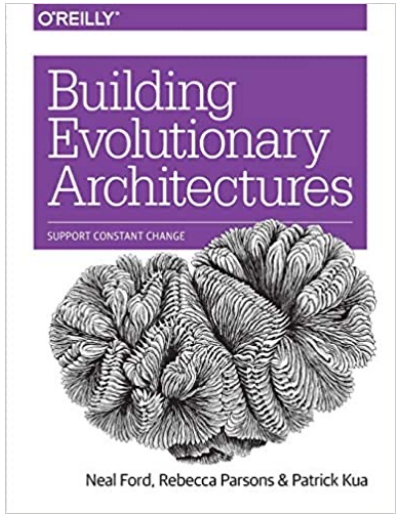


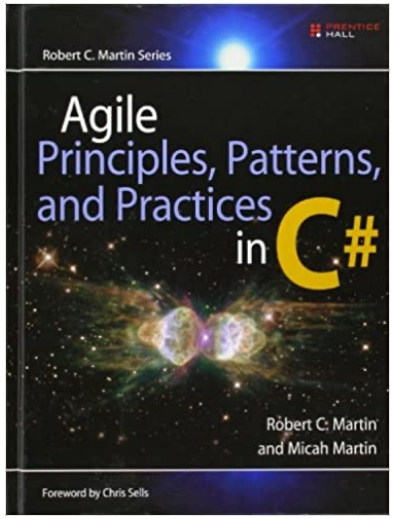
Software Design

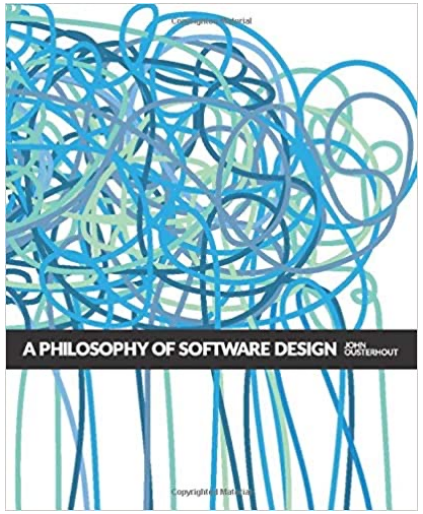






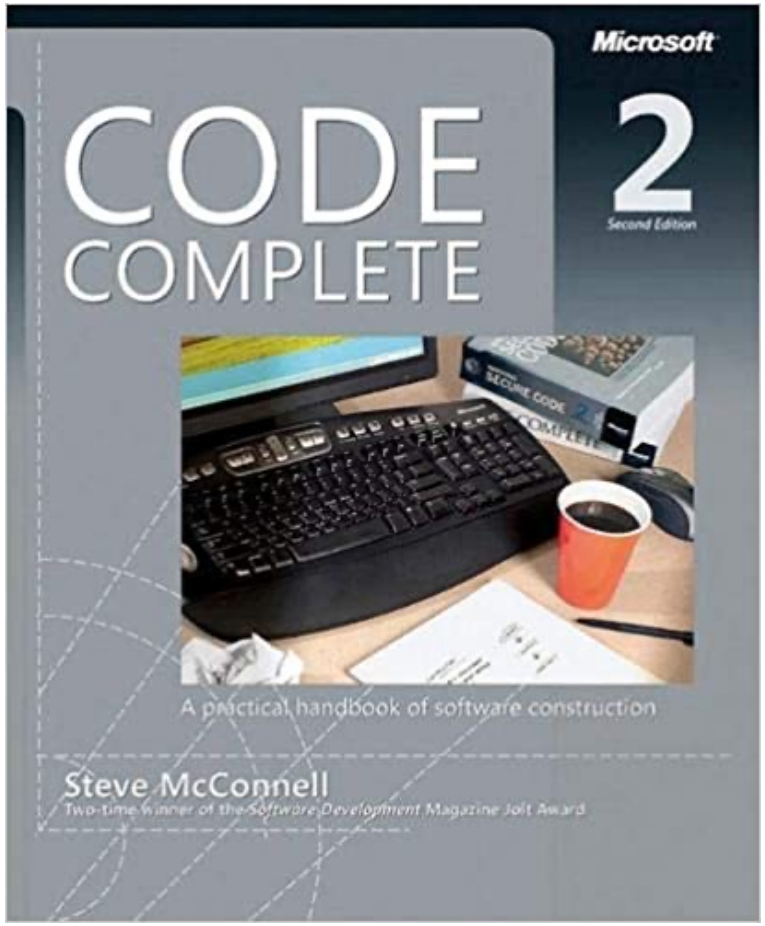




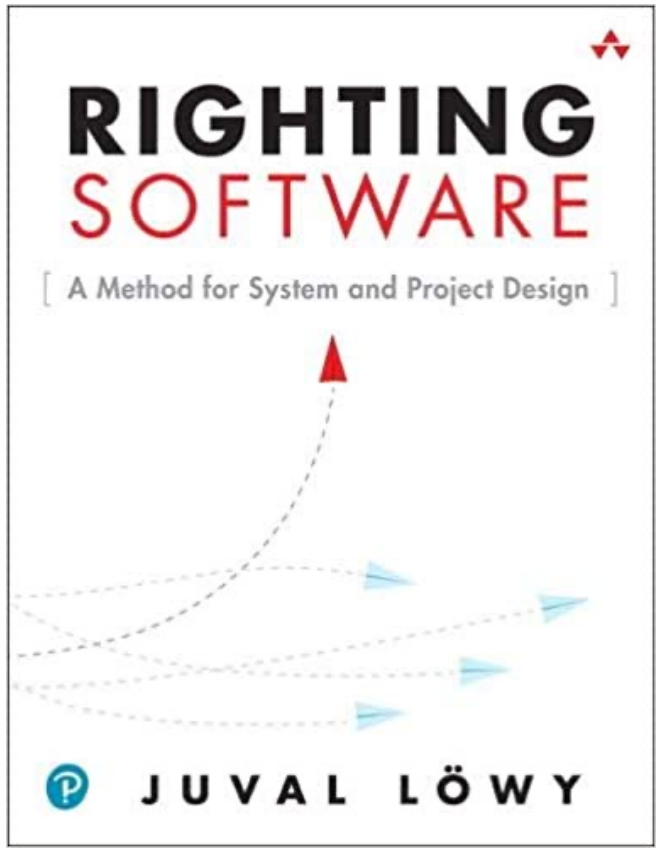


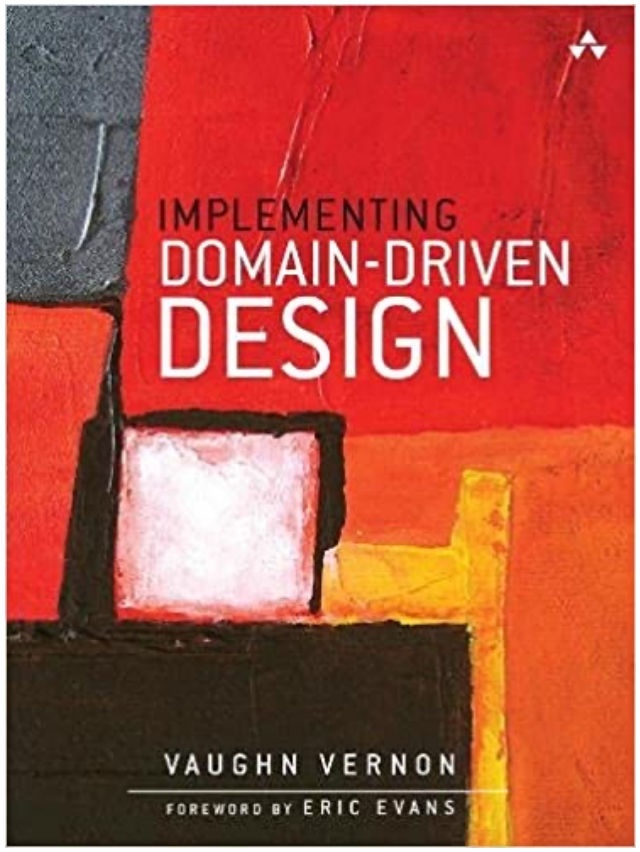
DON'T PANIC LABS

Software Testing

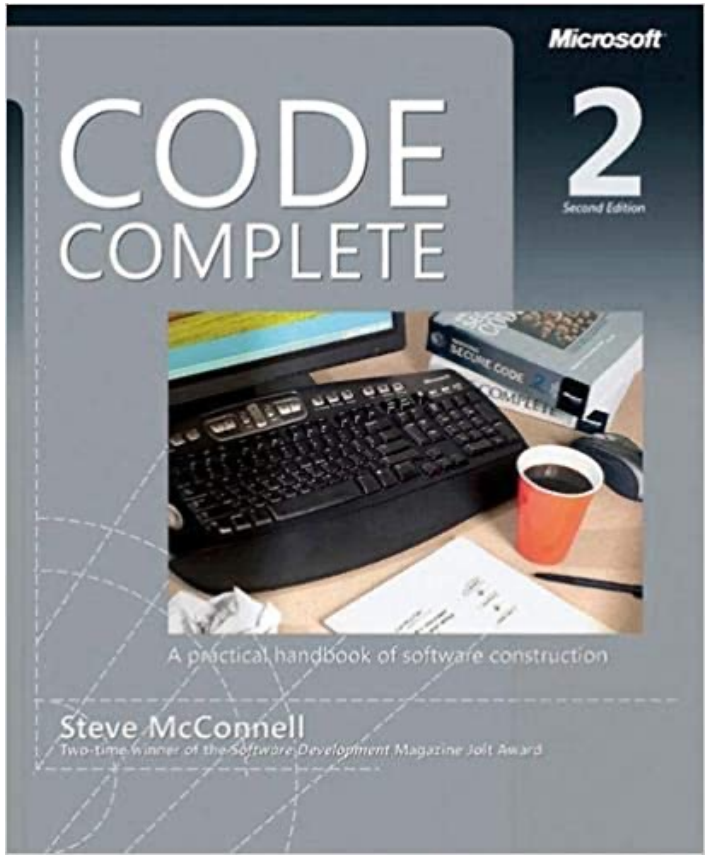


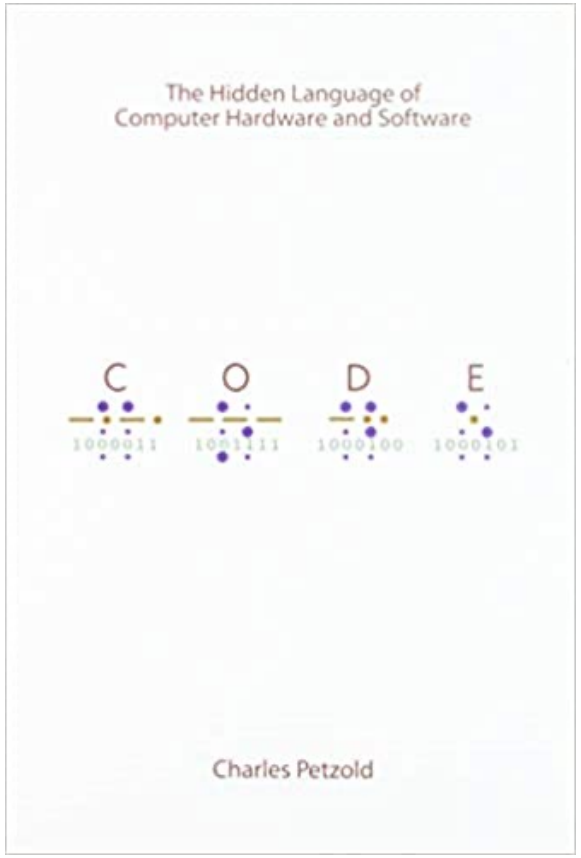
Software Engineering Models and Methods

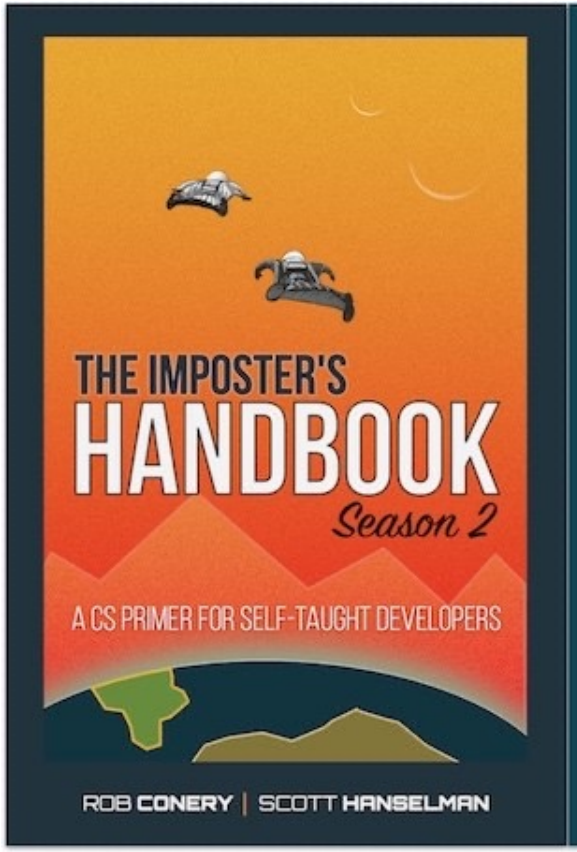


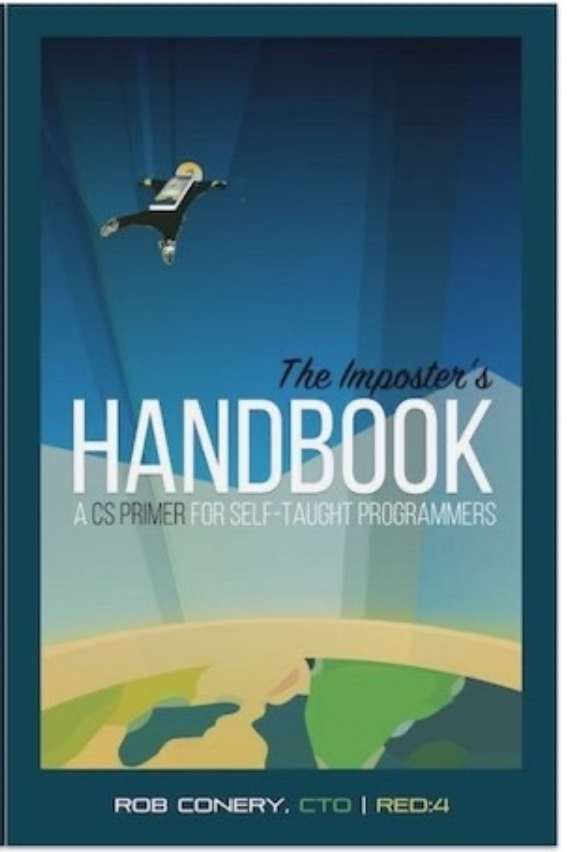


Computer Science Fundamentals



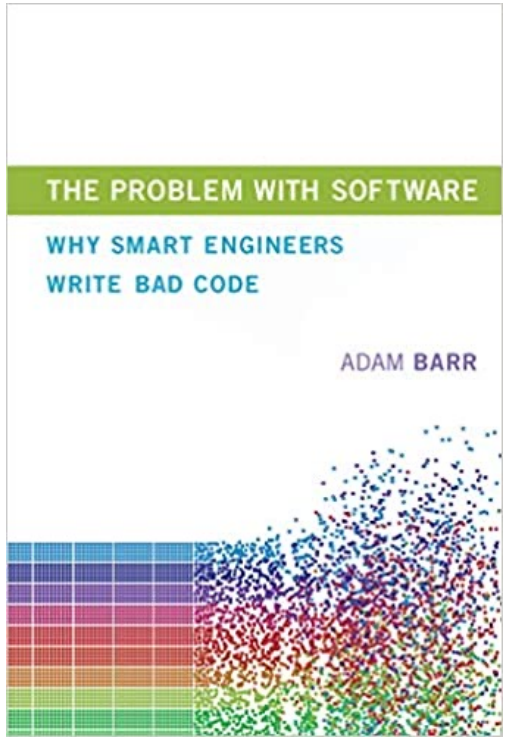


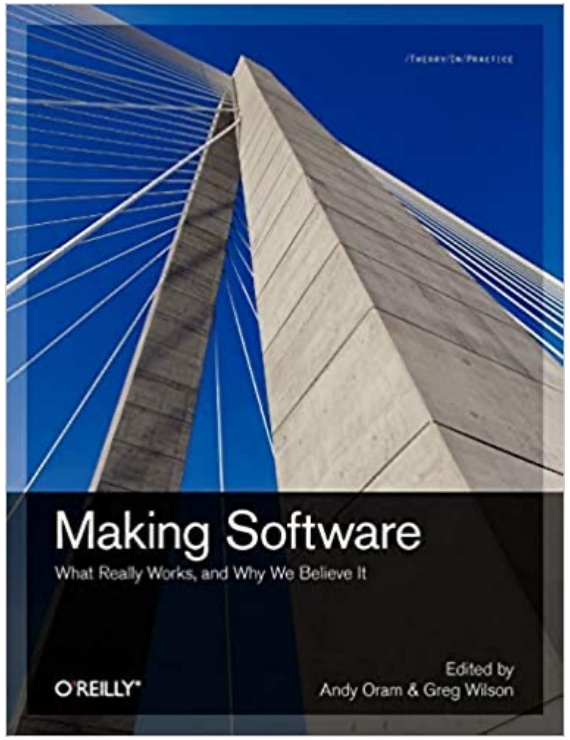


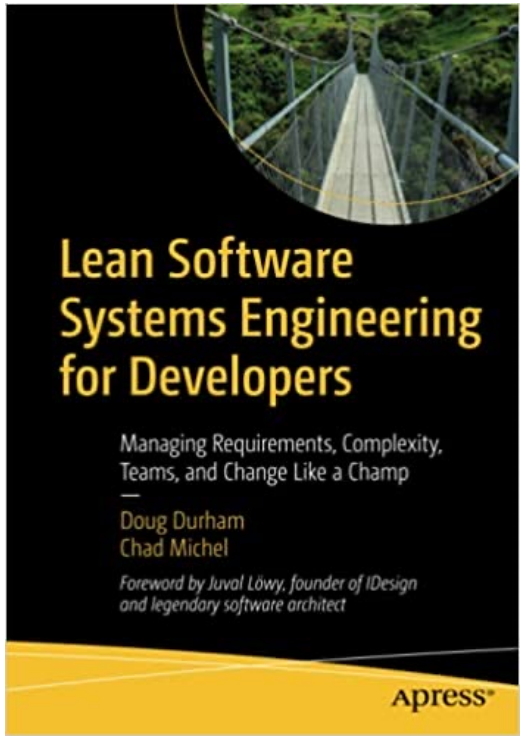


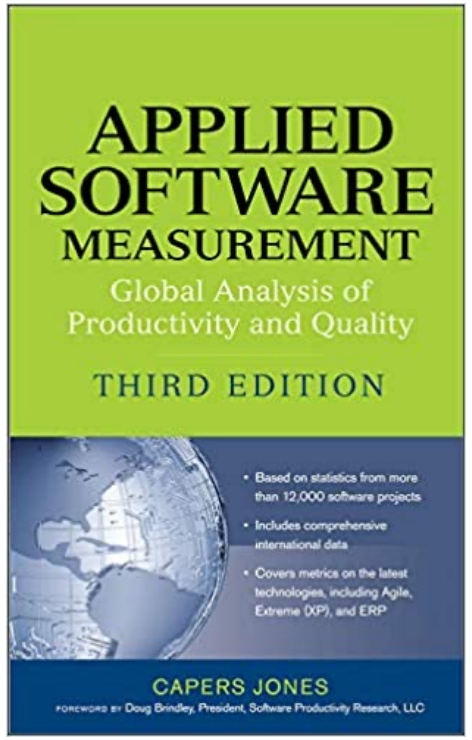
DON'T PANIC LABS

Other Interesting/Useful Sources









Thanks, and let me know what you think!

"If builders built buildings the way programmers wrote programs,
then the first woodpecker that came along
would destroy civilization."

Gerald Weinberg

- ddurham@dontpaniclabs.com
- dougdurham.com
- @dnsdurham

DON'T PANIC LABS