CSEET 2016

Software Engineering Core in an Age of Specialization

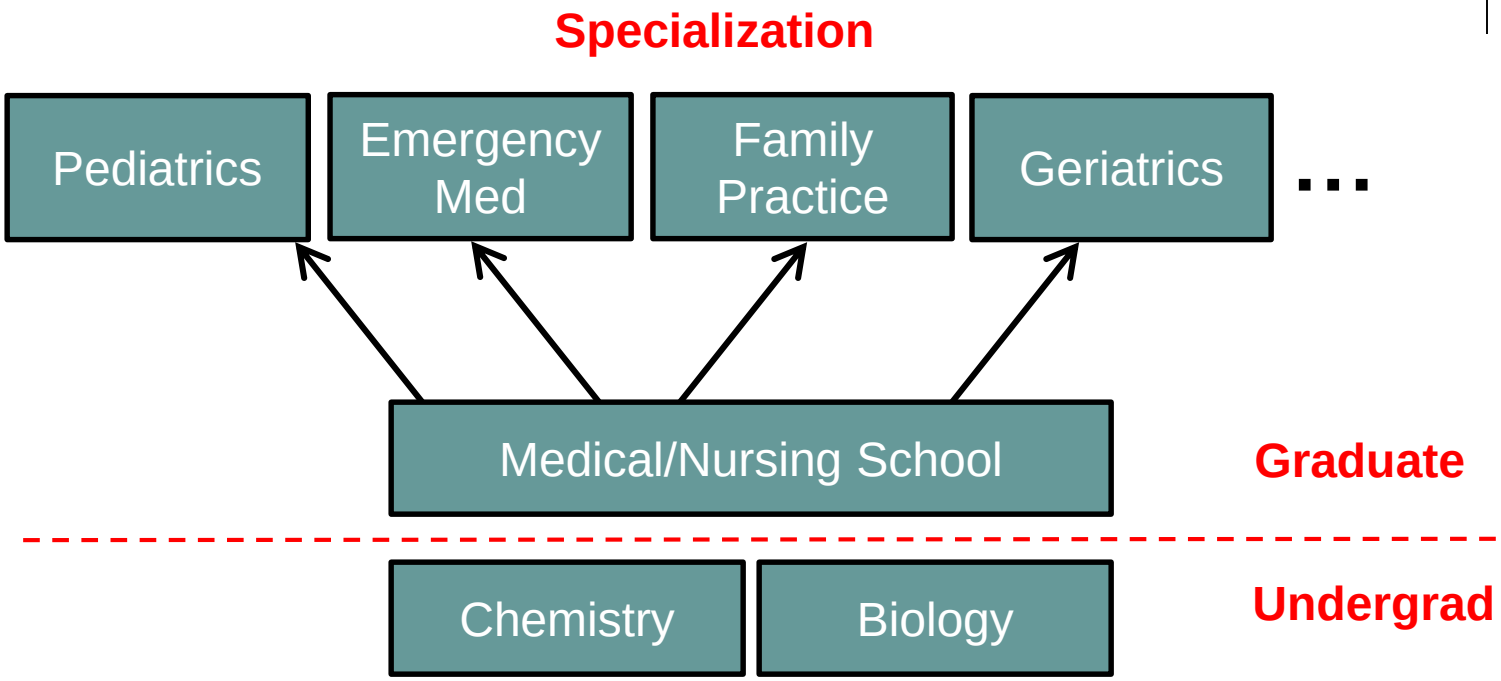
Anthony J. Lattanze

Director, Professional Software Engineering Programs.

Carnegie Mellon University, Pittsburgh PA, USA

"10000 hours of deliberate practice to become an expert" – Malcom Gladwell, Outliers.

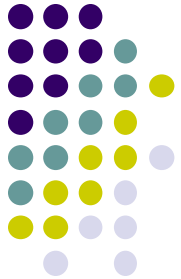
Introduction



Is it time for a similar formalized specialization strategy in Software Engineering?

We think so… this is the topic of my talk.

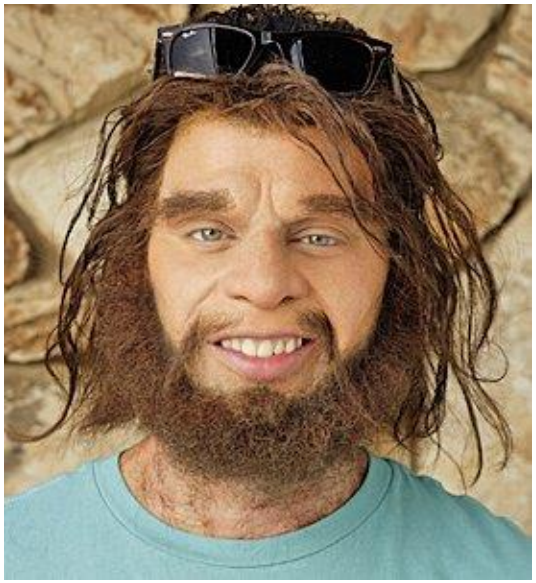
Topics



- Overview and Evolution of our Programs
- The Need for Specialization
- Our Approach to Specialization
- Challenges

CMU Masters of Software Engineering (MSE) Evolution

In 1991, in a little "cave," deep in the basement of the SEI, 6 MSE students huddled together and studied software engineering…



Today, in a third floor studio space in Pittsburgh, 75-100 students from numerous nations, various collaborating universities, in 4 programs huddle together and study software engineering…

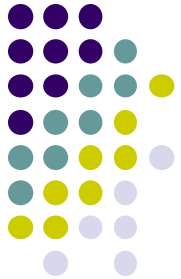
General Program Goals: Filling the Tool Bag

- In our program, students fill their “tool bags” with enduring principles to prepare them for a career as a software engineer.

Tool bag then,…

Tool bag now,…

General Program Goals: Creating Agents of Change



- All of our programs are graduate programs where are students have some level of industrial experience.
- Our goal is to *develop the future leaders of industrial software engineering*… serve as Agents of Change and SWE Evangelists:
  - influence organizations and (more broadly) the state of the practice
  - firm basis in theory, principle, and best practices, and know how to best apply it in practice
  - balance of people, process, and technology

What is an Agent of Change?

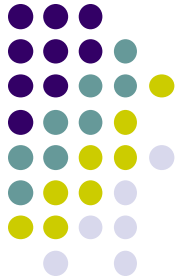
"César reflects upon his experience as an MSE student,… he will reveal how two essential technologies, PSP/TSP and Software Architecture, proved to be foundational concepts that differentiated Quarksoft from its software competitors and enabled it to grow into the successful, profitable IT company it is today."

CÉSAR MONTES DE OCA VÁZQUEZ, CO-FOUNDER / QUARKSOFT

With greater than 20 years of software development experience, César Montes de Oca Vázquez is co-founder of Quarksoft and its current CEO. He graduated from Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM) in 1993 with a B.S. in Computer Science. He completed Carnegie Mellon's Master of Software Engineering (MSE) in 1991, with a specialization in processes and business. In 2000, he earned his MBA from Carnegie Mellon. He serves as a professor in the Master in Information Technology program at ITAM in Mexico.

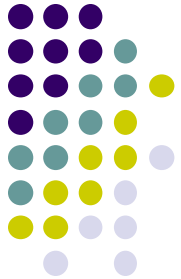
MONDAY, OCTOBER 28
NOON – 1:30 P.M.
300 S. CRAIG STREET, ROOM 265

SWE Industrial Landscape (Circa 1989)



- The technological landscape was very different at the time our programs were created:
  - Google and Amazon did not exist and Java hadn't been invented.
  - The first web browser (Mosaic) was 4 years from release.
  - The i486 was the top-end PC processor.
  - IBM was restructuring from a HW to a service company.
  - The cell phone was not widely adopted and smart phones hadn't been invented… pagers were state-of-the art.
- The state of the practice of SWE was relatively immature… demand was high for any trained software engineers

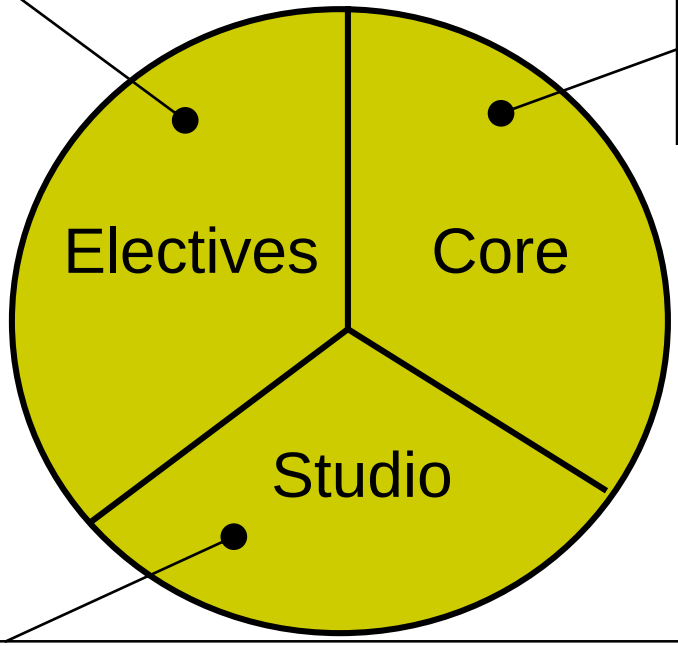
MSE Students 1989–1999



- Our average students were practicing software engineers at, or near the mid-career point with 5+ years of experience:
  - most had experienced many classic SWE problems before they came to the program
  - most were preselected and sponsored by their organizations
- On return to their organizations, most would assume senior leadership positions
  - agents of change within their organizations
  - SWE evangelists

MSE 1.0 Program Structure

- Opportunity for study in areas of personal or professional interest.

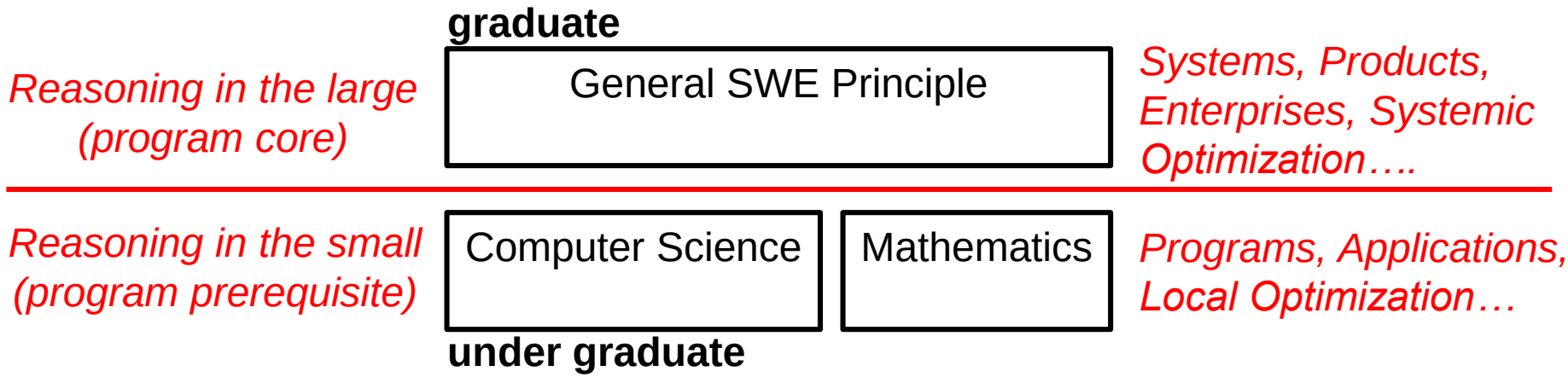


- Academic backbone of the program.

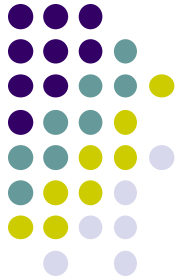
- Provides opportunities to apply core material in mentored environment within a realistic context.

Importance of the SWE Core

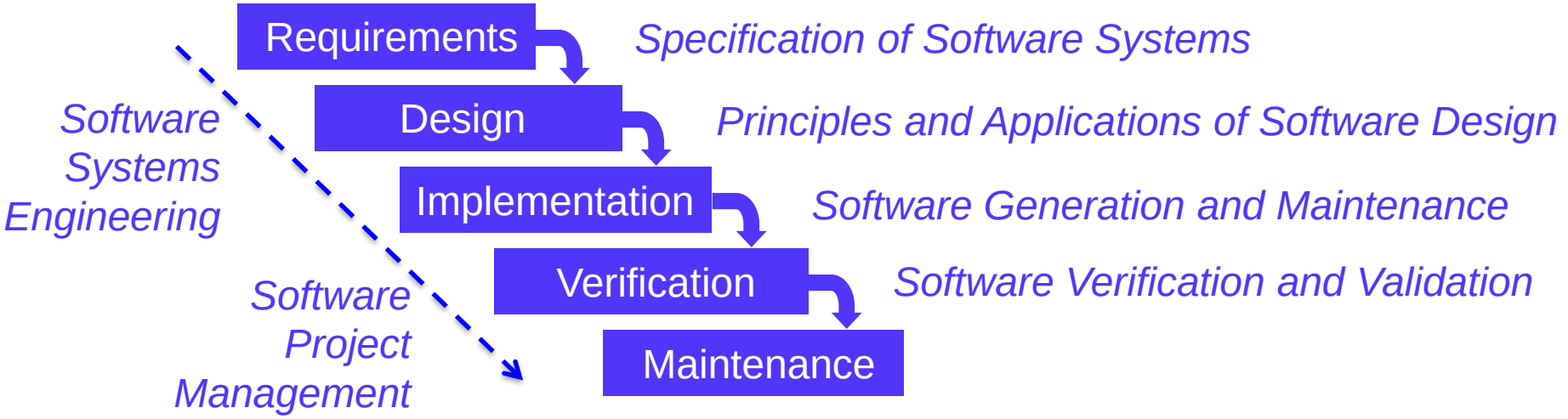
- The core courses represent the soul of our programs – it is the *common, enduring* principles of software engineering
  - **common** – applies to all domains
  - **enduring** – lasts beyond passing fads and hype



Waterfall Lifecycle Oriented Core Curriculum – MSE 1.0

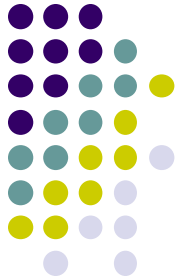


- The first core mirrored the traditional waterfall model as described by Royce[3] (made famous by 2167A)



[3]Royce, Winston, "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26, August 1970

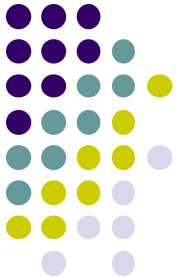
Core Curriculum Revision

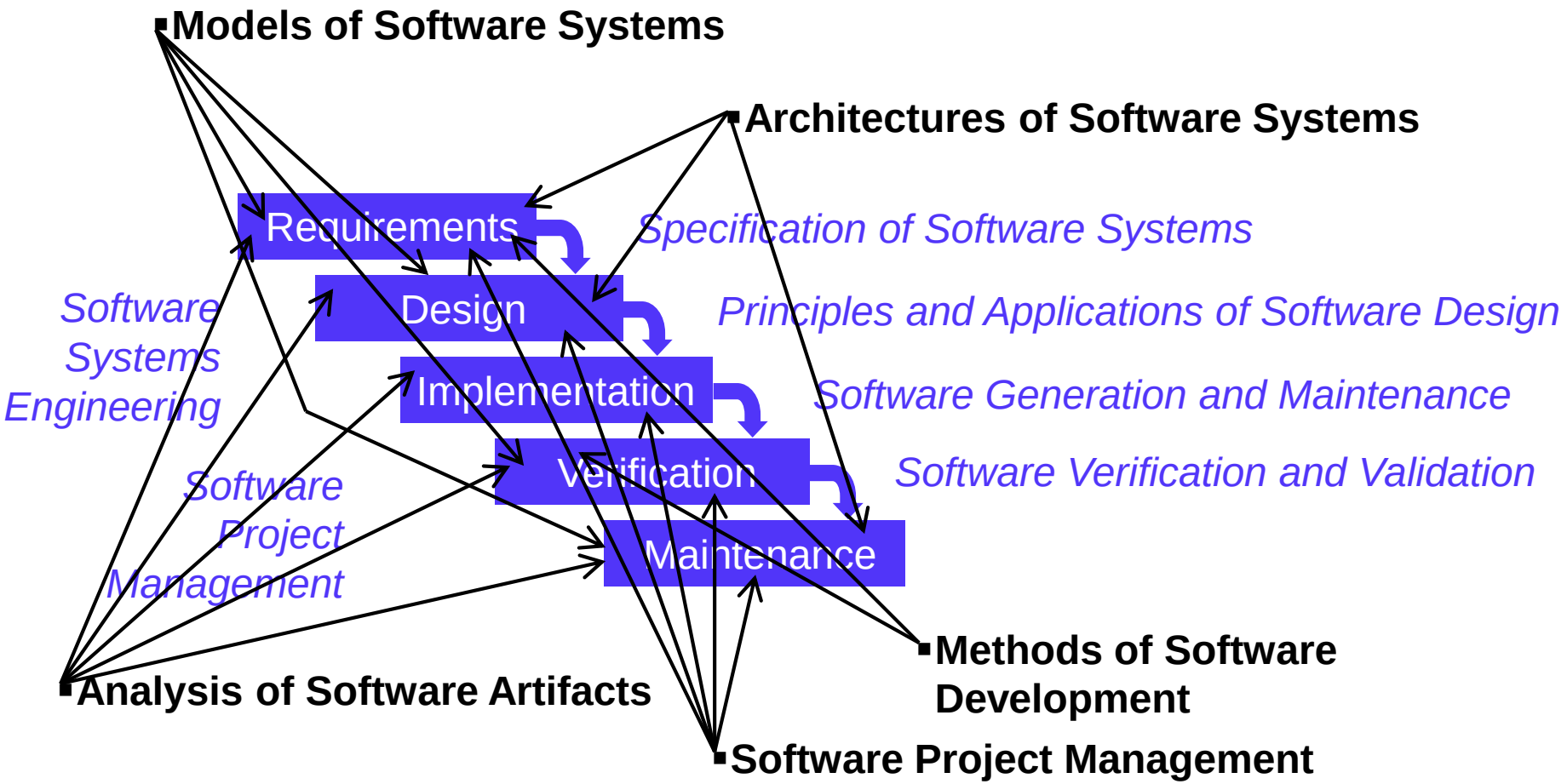


- The MSE program was revised in 1993 and was moved from the SEI to the School of Computer Science (SCS).
- The program core was restructured:
  - David Garlan, Mary Shaw, Jim Tomayko, Daniel Jackson, Jeannette Wing,… and others
- The 6 core courses were consolidated into 5 applicable at various phases of the lifecycle.

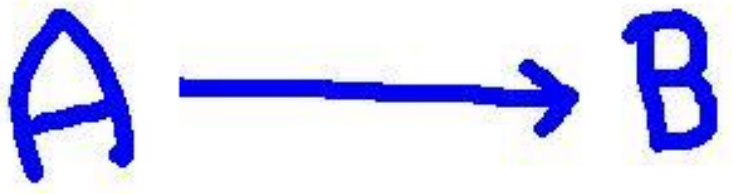
[5] David Garlan, David Gulch, James Tomayko, *Agents of Change: Educating Software Engineering Leaders,* IEEE Computer, November 1997

Waterfall Lifecycle Oriented Core Curriculum – MSE 2.0





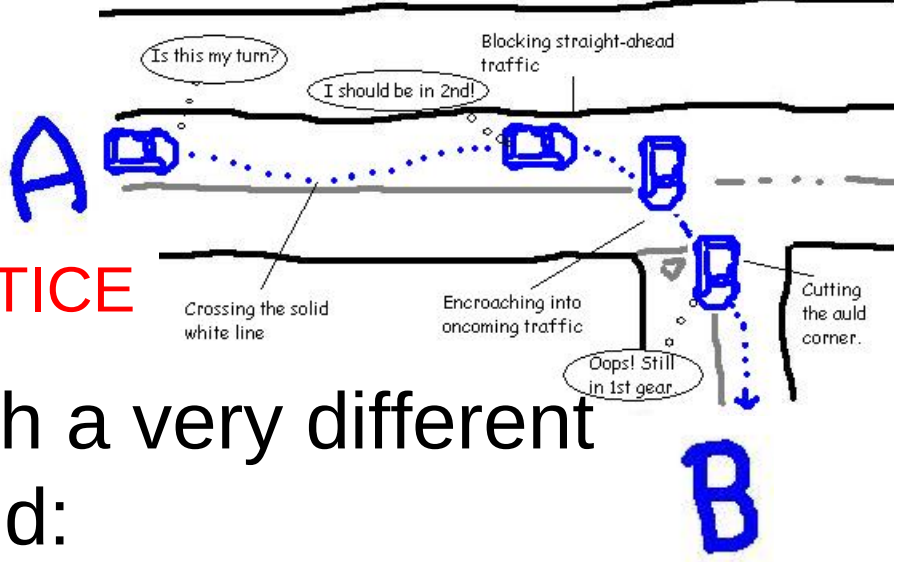
Traditional Approaches



THEORY

- Many university computer science and software engineering programs focus on:
  - software engineering and computer science theory
  - individual assignments, thesis, limited or no team work
  - contrived problems, textbook examples, problem-solution recipes
  - testing how well students remember concepts

But In The Real World…



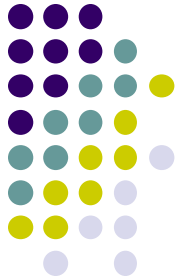
PRACTICE

- Engineers are faced with a very different situation in the real world:
  - recipes don't work in practice
  - systems in practice are larger and more complex
  - products and systems are built in large teams
  - there is often no single "right answer" and engineers ***must apply judgment***
- The project course is an opportunity to apply core course principles in a real-world context.

Project Courses

- The projects are the hallmark of our programs…
  - 16 month, industrial customers, ~5 students teams
  - compliments the program core – principle without practice leaves student unprepared
- Student teams are required to…
  - get the requirements from the customers
  - analyze the requirements
  - design the solution
  - implement the system
  - manage the project
- Provides opportunities for students *to practice engineering judgment in a realistic context.*

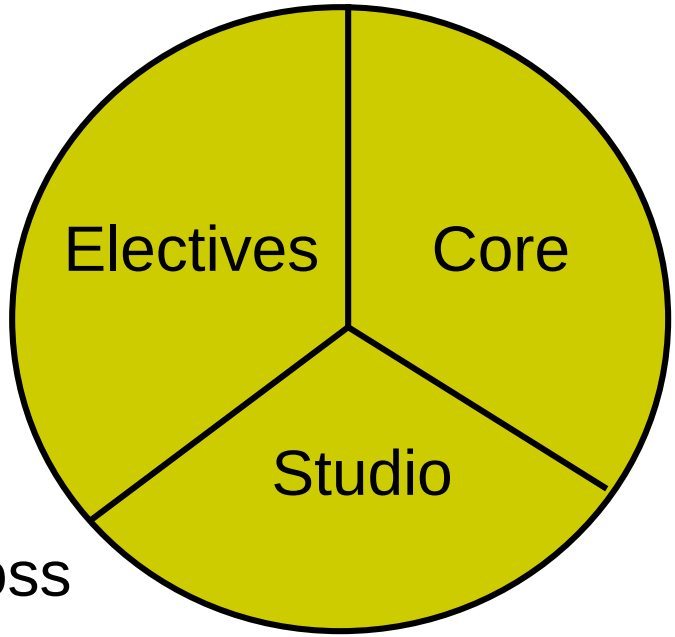
Scaling the Program – 1



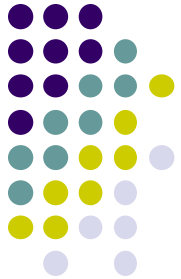
- 1999: MS in Information Technology (MSIT)
  - same core as MSE program
  - shorter in duration (3 semesters)
  - MSIT projects of smaller scope than MSE
- 2006: Embedded Software Engineering (ESE)
  - projects include HW and SW
  - core is variant of existing MSE core - expanded to include systems engineering and ECE courses
- In these programs students…
  - have less experience (1-2 years of experience)
  - are more often self-funded

Scaling the Program – 2

- The current model of common core, studio and electives has served us well.
- However, the ideal of an immutable core is at its limits given needs of…
  - expectations of hiring managers
  - numerous SWE domains
  - variation in SWE approaches across domains and business context

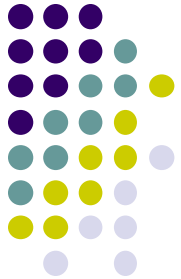


Topics



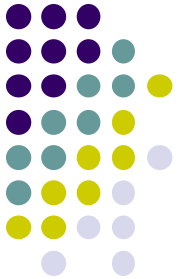
- Overview and Evolution of our Programs
- The Need for Specialization
- Our Approach to Specialization
- Challenges

Today's Industrial Landscape



- Software is pervasive and cuts across all domains and aspects of modern life:
  - Internet/Web
  - Cloud centric
  - Data intensive organizations
  - Internet of things
  - Embedded/Cyber Physical systems (IoT, medical, automotive, aerospace, commercial electronics,…)
  - Gaming
  - Government and military
  - Mobile computing

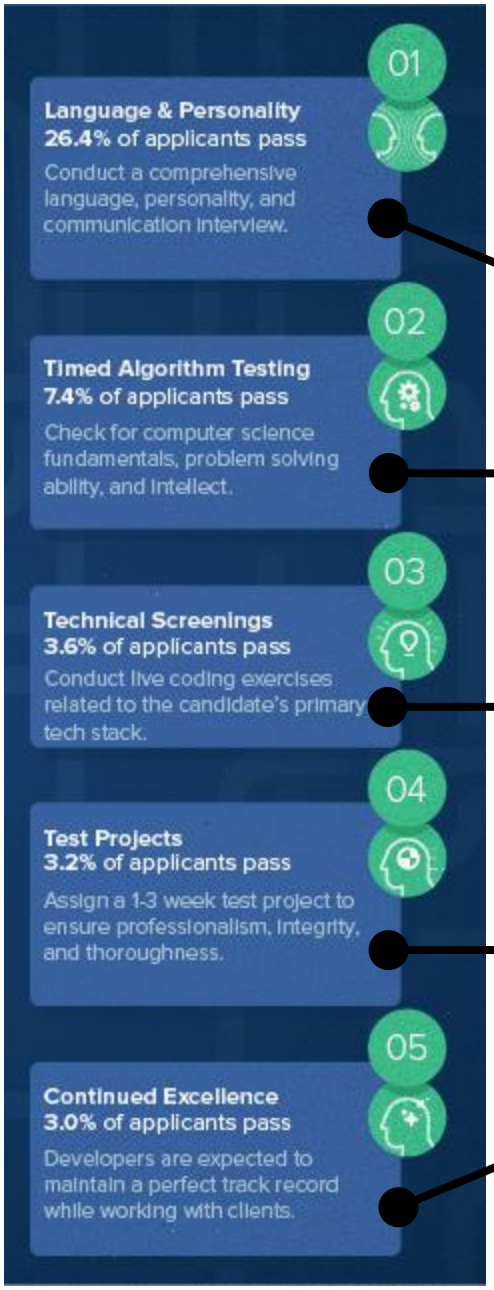
Organizational Expectations



- Recruiters and hiring managers expect students have technical knowledge consistent with the needs of their organizations - *especially true for students will little industrial experience.*
- Given the diversity of software domains today, there is often little in common between what different organizations are looking for in candidates.

Top 5 Qualities Sought

(for new hires with little experience)



Soft Skills, Cultural Match

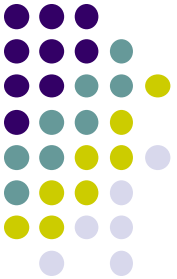
Fundamental Technical Skills

Domain Specific Knowledge and Technical Skills…
(#2 for more experience hires)

Relevant experience

https://www.toptal.com/freelance/in-search-of-the-elite-few-finding-and-hiring-the-best-developers-in-the-industry

Examples – 1



Applications Programmer

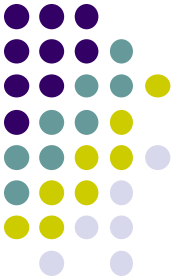
(BANKING DOMAIN)

a rapidly growing company is looking for a Great applications programmer. We need programmers who will work closely with Customer Service to identify and resolve issues while helping to build a world class suite of applications for the mortgage industry.

Experience: 2 years+ with a wide background working with:

- Databases: SQL Server, stored procedures, linq
- Development Environment: Visual Studio, C# for now, mobile apps soon
- Standards: Bootstrap, JQuery, SignalR, SSIS, SSRS, SVN
- Communications: API's, XML, Fiddler, ARC

Examples – 2



Programmer

(Gaming Domain)

We are looking for talented programmers to help create and further the technology that runs the top grossing game

Requirements: Ideally candidate should be able to check a few of these boxes:

C++

Objective C

Parse

Real-Time, Schedule driven programming

Mobile App Development

Game Engine Experience

Android, iOS Experience

Operating systems internals

Examples – 3

Description

TECHNOLOGY SERVICES GROUP (TSG)
(CLOUD DOMAIN)

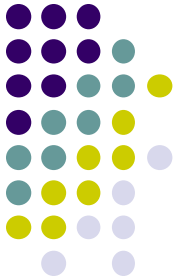
Key Job Duties to Include:

- Develop and deliver services associated with our Cloud Lifecycle Management, Automation and Orchestration platform.
- Support Hypervisor Engineering, e.g. VMWare, Hyper-V, KVM
- S/he designs, implements, integrates and supports Cloud Infrastructure Components; grow the delivery of internal cloud offerings.
- Design, deliver and support go forward Cloud technology roadmap that includes, but is not limited to, OpenStack, Cloud Foundry and Puppet.

Qualifications

- Expert proficiency in Hypervisor Engineering, e.g. VMWare, Hyper-V, KVM
- System Engineering skills in both Unix/Linux and Windows along with understanding of Veritas Clustering, SSH, Telnet, SNMP
- Strong experience with scripting in Linux/Windows environments (Such as Powershell, nsh, java script, perl, python, ruby-on-rails)
- Experience with RHEL LINUX
- Experience with OpenStack, Cloud Foundry a plus
- Experience with SOAP / REST APIs a plus
- Hands on experience in an IT Engineering role and established IT operations methodology experience (e.g. ITSM/ITIL)
- Experience with Storage Products from EMC, NetApp and IBM
- S/he must have expertise in developing detail-level functional requirement documentation.

Examples – 4



Embedded Software Engineer

(Embedded/Space Domain)

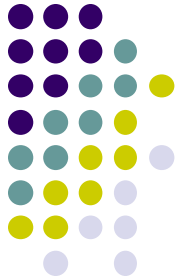
Required Skills

- Bachelor's degree in Computer Science, Electrical Engineering, and/or other relevant engineering field degrees
- 5 years of experience in Embedded Software development with at least 2 years in a mission critical environment
- Development of Real Time embedded software, particularly with FreeRTOS and/or RTEMS
- Experience with developing device drivers & BSPs with an emphasis in PowerPC architectures and Ethernet, Serial I/O, and USB peripherals.
- Experienced using bench equipment; oscilloscopes, Logic Analyzers, In-Circuit Emulators, J-TAG debuggers
- A very high level of skill with C/C++
- Designing and documenting software test procedures
- Experience with Hardware/Software integration and debugging

Required Experience

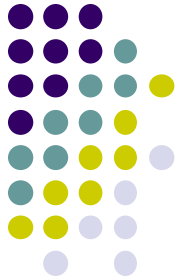
- Experience capturing requirements and designs in Requirements Management tools such as DOORS
- Experience in a Scrum agile development environment.
- Experience with other RTOS (VxWorks, Integrity, Xenomai, etc)
- Experience with Python and other scripting languages
- Experience with data collection and analysis
- Experience with software testing (V&V)
- Strong Analytic and problem solving skills
- Strong team leadership and communication skills
- 10+ years of experience in avionics electronics software development
- Experience with multiple cross-platform development environments including Windows, MacOS, and Linux.

Extreme Examples of Industry Specialization – 1



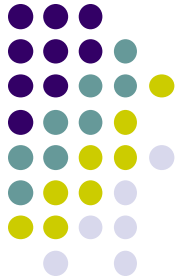
- In IT/data/Internet domains,...
  - software is decoupled from compute hardware, operating systems, networks, peripherals
  - system design and development is decoupled from infrastructure and operating systems
  - hardware, infrastructure, OSs, and so on are commodities can be designed and/or procured with minimal dependence on application design
  - a great deal of commonality between systems providing a rich set of patterns and applications to support system design and development

Extreme Examples of Industry Specialization – 2



- In the embedded domain,…
  - Systems and software interact with a physical environment.
  - Applications often must be deterministic.
  - HW, OSs, networks, and sensors are often special purpose to meet physical constraints.
  - Software, CPUs, networks, sensors, peripherals, etc. are tightly coupled and drives the design of systems.
  - Embedded systems vary widely.

From a SWE Perspective What is Different?

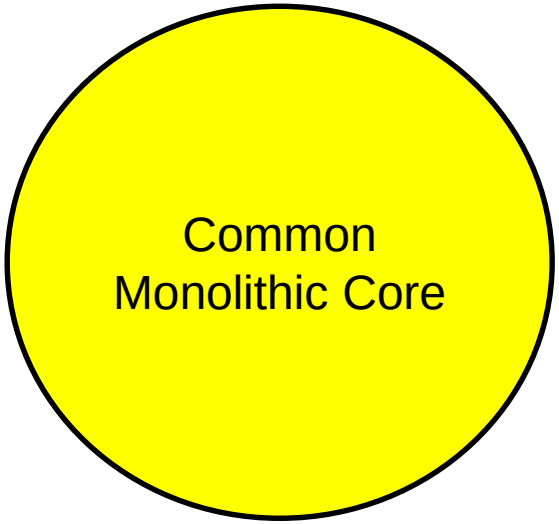


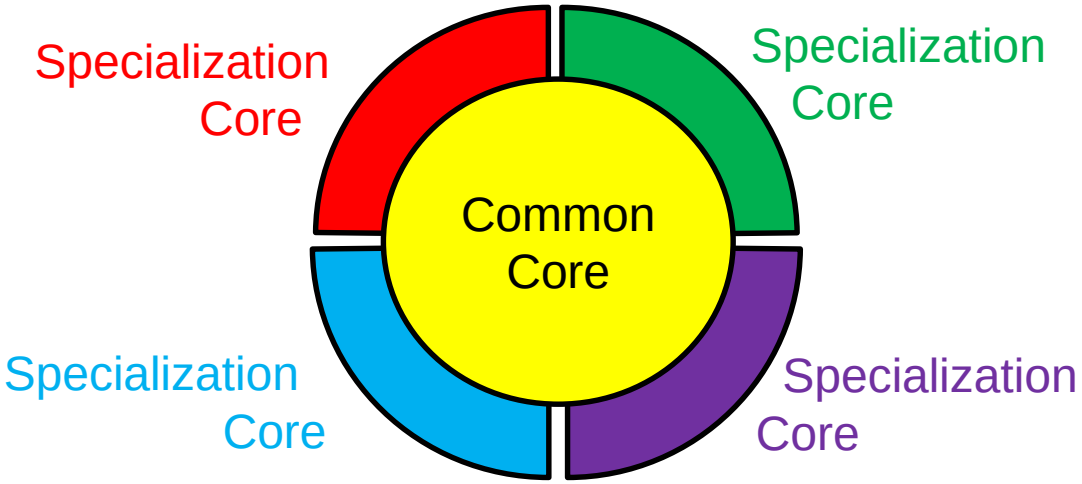
- Everything:
  - Lifecycle and Detailed Processes
  - Customer interaction, detailed requirements elicitation, and management
  - Design and the design process
  - How we go about the process of construction
  - Verification and Validation
  - Maintenance
- These differences shrink the amount of commonality in SWE core.

What is the Same

- A smaller set of basic concepts and principles.
  - the need for disciplined SWE processes
  - the role and importance of requirements engineering
  - fundamental design principles
  - basic configuration management principle
  - principles in testing and test planning
  - teaming, communications, and organizational skills
  - project planning, tracking, and risk management skills
- While there is more to know about SWE, that which is common across domains is probably smaller.

Issues With a Monolithic Core





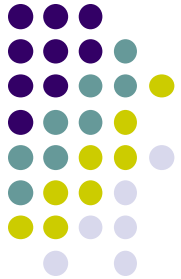
5 full semester Core Courses:

- focus on software engineering generalists – one size fits all
- many topics are not relevant to all specializations
- some topics not covered in depth

Smaller Lightweight Common Core:

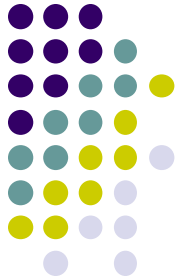
- secondary core courses built on common core to focus specifically on specializations
- provide better depth of topic coverage

Our Vernacular: Specialization Verses Tracks



- Tracks…
  - a common approach to tracks is to throw in a couple of electives of a particular type into an existing program typically with the same core
- Specialization…
  - we think “specialization” is deeper than this approach in that the core courses are tailored for the needs of a particular specialization
- To implement specializations, we will need to rethink and restructure our monolithic approach to the common core courses.

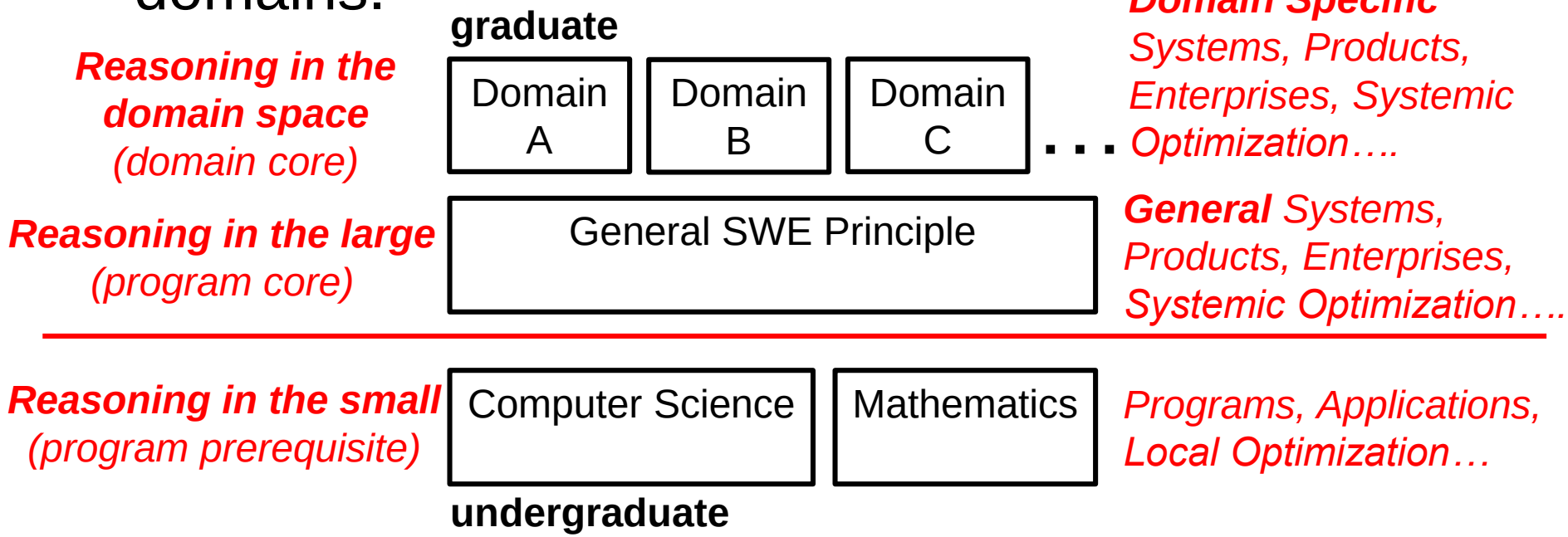
Topics



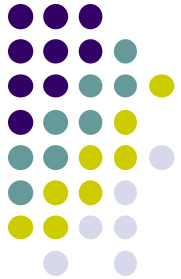
- Overview and Evolution of our Programs
- The Need for Specialization
- Our Approach to Specialization
- Challenges

Goals

- Create a suite of programs with a lighter-weight core that addresses general SWE principle, but provides depth of knowledge in specific domains.

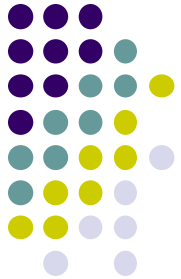


Student Demographics and Their Needs



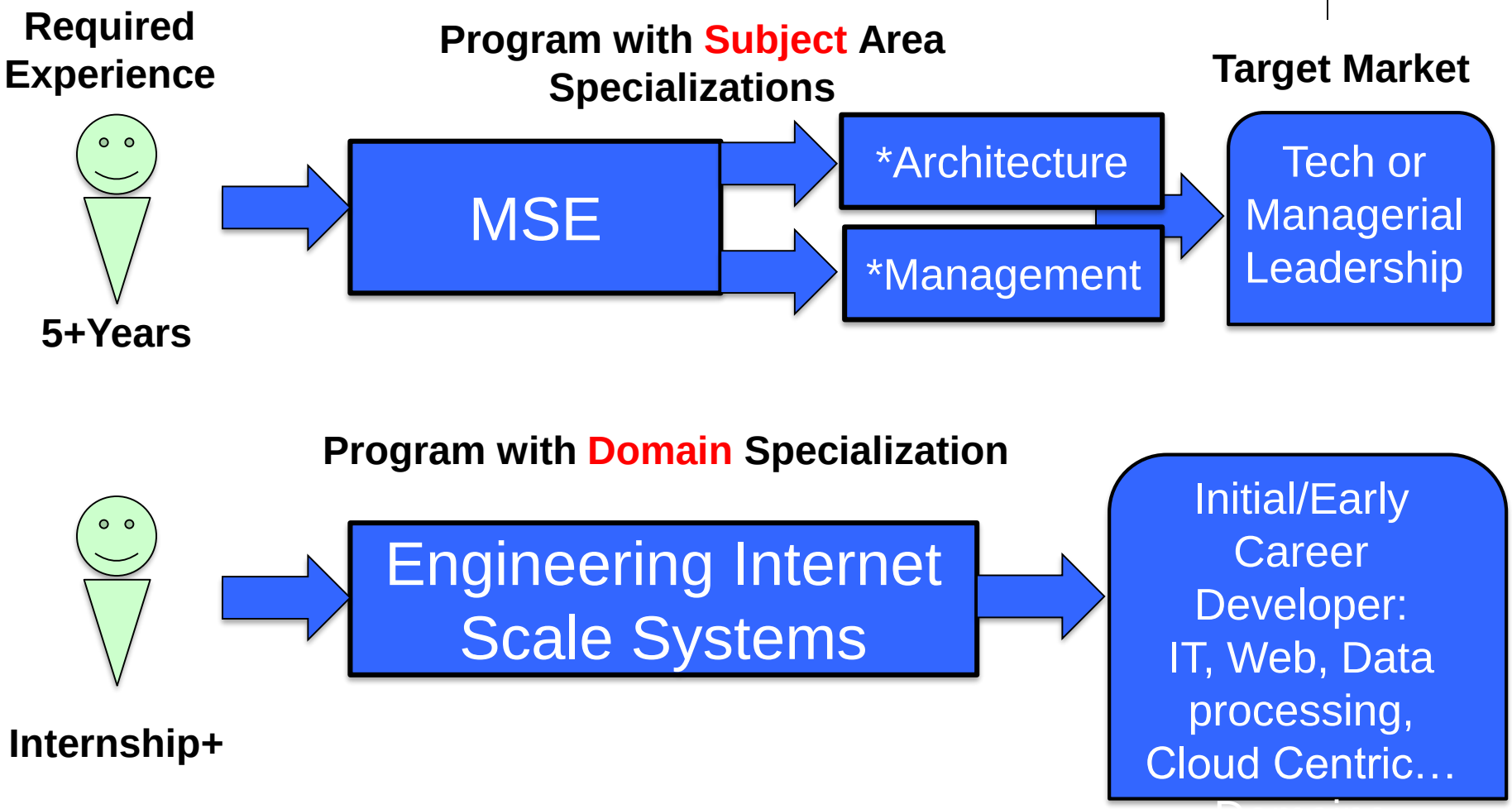
- Students with little or no industrial experience are often looking…
  - for their first professional position
  - to be more competitive in a specific domain
- Students with 5 or more years of experience are often…
  - looking to refresh general software engineering skills
  - preparing for more senior technical or managerial leadership positions
  - change agents and more effective leaders

Domain Verses Subject Area Specialization – 1



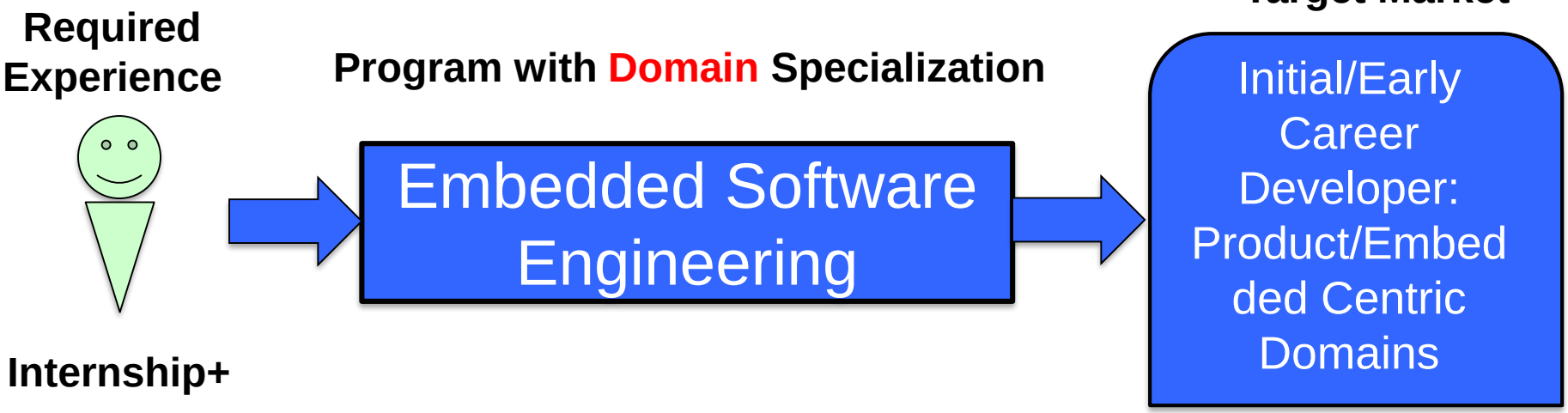
- Subject area specialization: focus on subject areas such as *artificial intelligence, robotics, security, analytics, etc.* This is common.
- Domain are specialization: focus on the needs of an industrial domains such as *enterprises, cloud centric, product centric,* etc. This is not so common.
- We are considering a combination of both for different student demographics.

Domain Verses Subject Area Specialization – 2



*These are proposed subject area specializations, not definite.

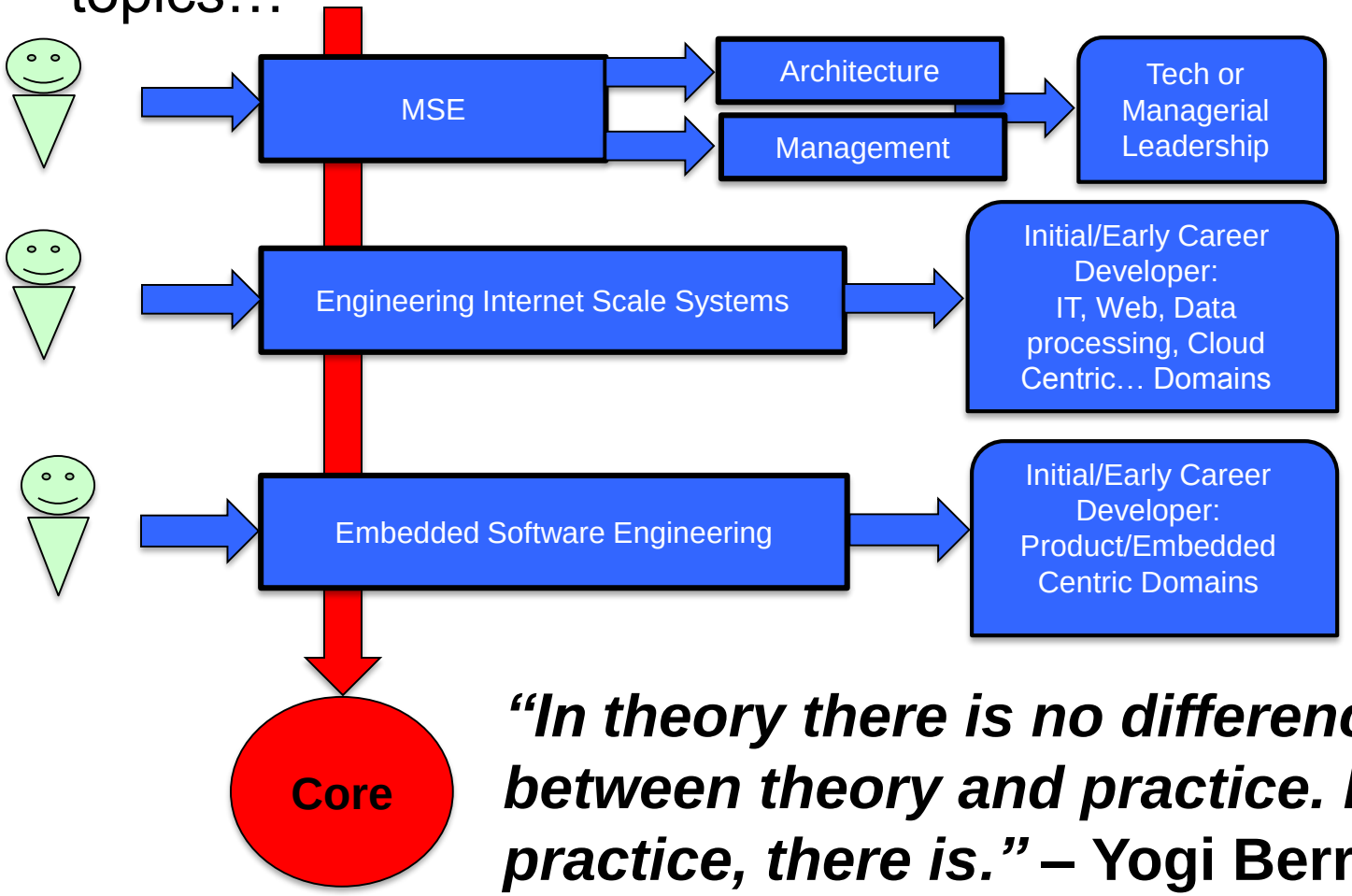
Domain Verses Subject Area Specialization – 3



- Our goals for students in programs targeting initial/early career positions is that they…
  - possess immediate technical competency
  - are ready for leadership positions: possess excellent teaming, project, and communication skills

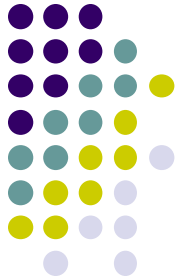
What About the Core?

In theory it should be easy right? Identify common topics…



"In theory there is no difference between theory and practice. In practice, there is." **– Yogi Berra**

Topics

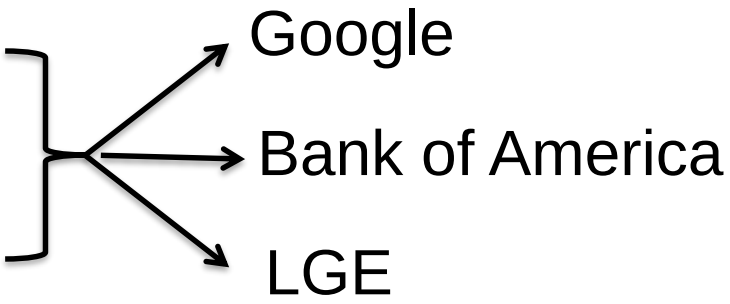


- Overview and Evolution of our Programs
- The Need for Specialization
- Our Approach to Specialization
- Challenges

Challenges: Defining the Core

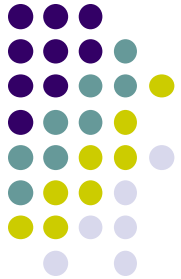
- SWEBOK and similar frameworks can help generally, the specifics of what is taught still must be instantiated.
- This is specially important when defining specialization specific core.
- Example (using SWEBOK version 3.0)

3. Requirements Elicitation
3.1. Requirements Sources
3.2. Elicitation Techniques



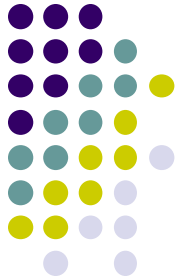
The way that "requirements elicitation" is done in each of these organizations is radically different and is a key motivation for specialization…

Challenges: Operationalizing the Core – 1



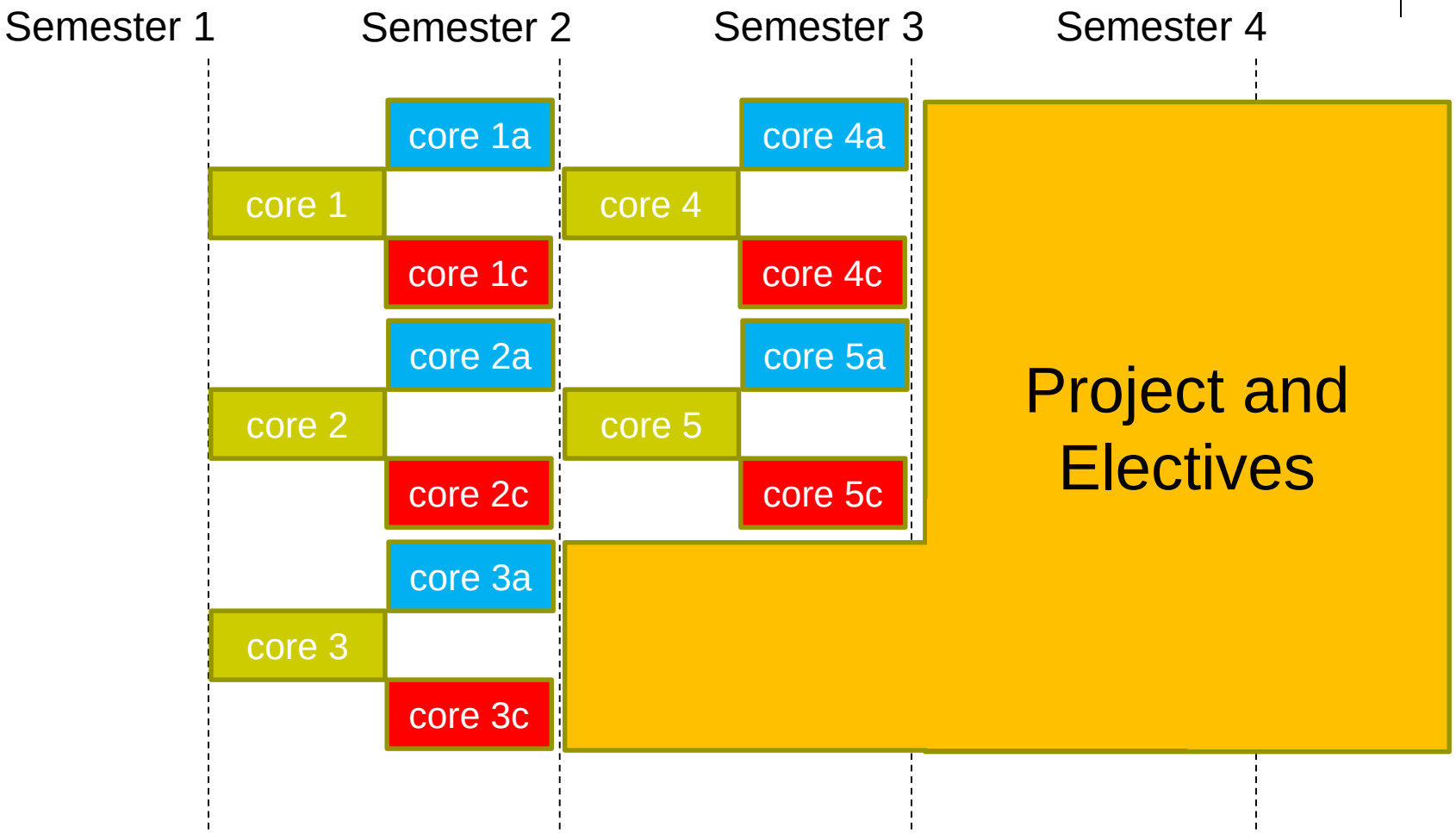
- Two key approaches we are considering:
  - "Thin," single common core course model – reduced set of courses that all students in all specializations take early in the program before they embark on their specialization core.
  - Common core framework mode– a common framework that we use to develop the core courses for all specializations ensuring that key core concepts are embodied in the specialization core.

Common Core Course Model

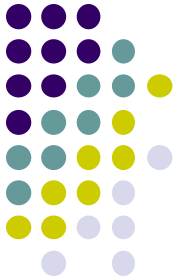


- Full Semester Core Courses won't work:
  - no way to break out individual specialties
  - we need to reduce core material that does not address the specializations
- Half semester core courses are an option
  - students take common core in one mini
  - then students take the relevant specialization core in a second mini

Example: Single Common Core Course Model

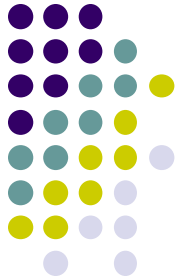


Tradeoffs with Single Core Course Model



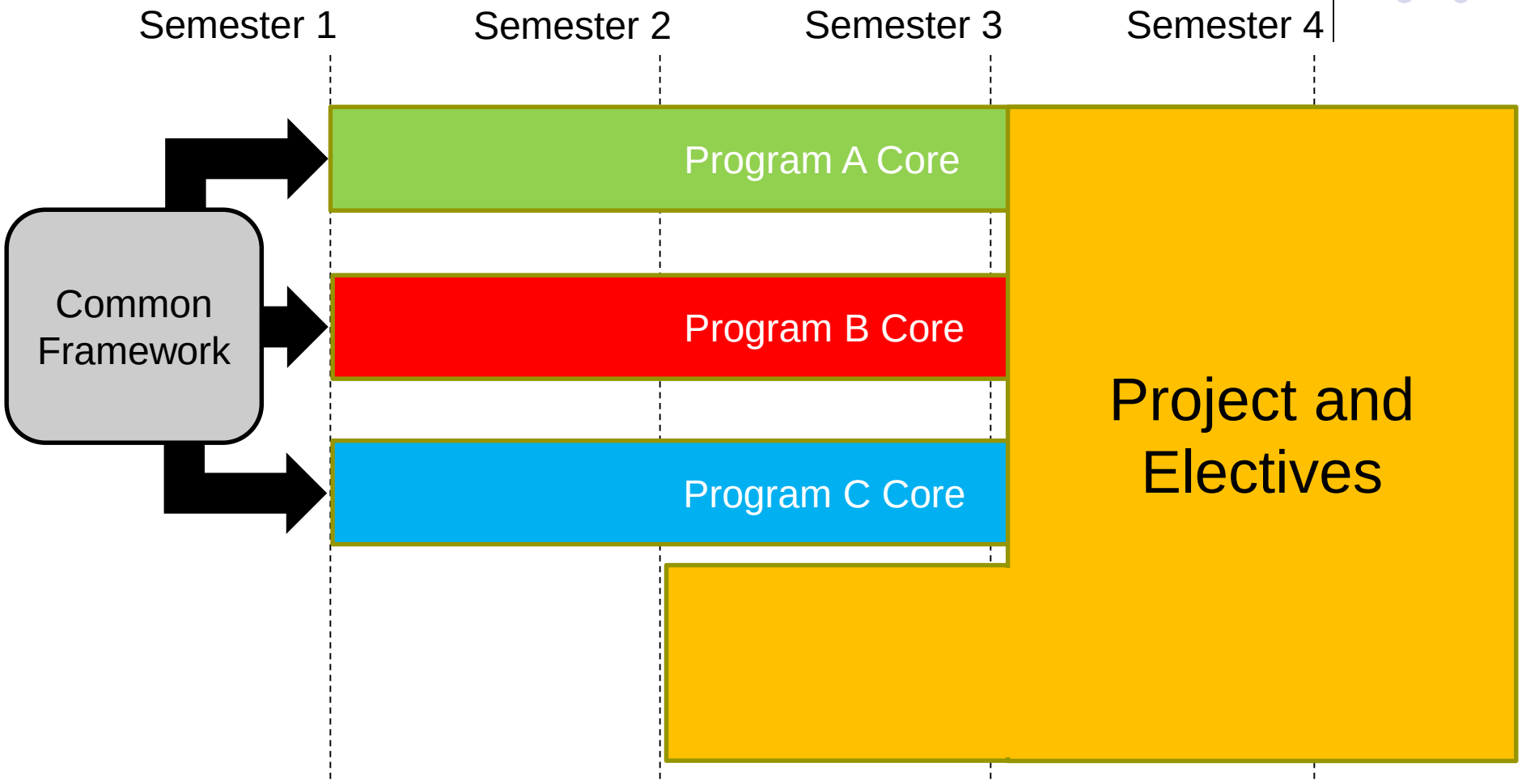
- Conceptually simple…
  - common core minis instantiated from existing 12 unit courses
  - specialty core are extensions of the common core
- Logistically difficult…
  - results in state explosion of mini courses
  - less flexible: binds each program to the same course and project schedule
  - not sure if a single common set of core courses is feasible

Common Core Framework Model

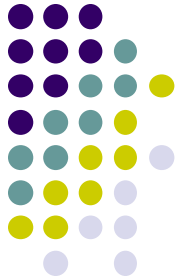


- Rather than common core courses, we design a common core framework *which defines the common learning objectives and general topic areas for all programs.*
  - each program would be responsible for implementing the core courses for their program consistent with their specialization
  - each program would follow their own course and project schedule
  - oversight would be provided to ensure that each of the programs are meeting the framework's learning objectives

Example: Common Core Framework Model



Tradeoffs with Common Core Framework Model

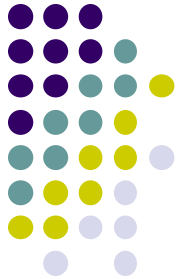


- Logistically easier…
  - each program develops and maintains their own set of core courses
  - flexible: each program follows their own course and project schedules
- Hard to develop and enforce…
  - not sure what such a framework might look like
  - not sure how we would ensure compliance with the framework and how much effort this would evolve
  - not sure it embodies the conceptual integrity of a set of common core courses

Other Challenges

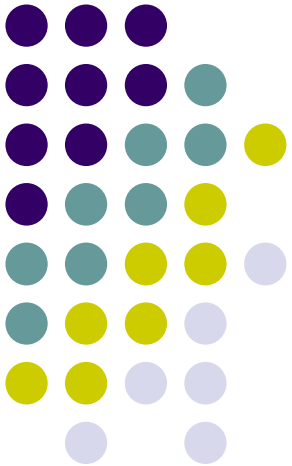
- Treading a fine-line between training and education.
- Not all common and specialization core courses fit traditional notions of semester boundaries and grading deadlines.
- Implementing project courses in specializations.
- Transitioning and continuity of operations.
- Fixed resources.

Finally



- While challenging, we plan to move forward with specializations.
- We think that industry needs software engineering specialists today, in the same way it needed software engineers 25 years ago.

CSEET 2016



Thank you!

Questions/Comments