Why We Will Never Have Enough Software Developers

```
import laborsupply

developers = laborsupply.load(skill="computer science")

developers.do_work(job="software development")

# ResourceWarning: Insufficient developers allocated
# Reason: Developer dropout
```

We will never have enough software developers.

Developers are dropping out of the profession in large numbers despite efforts to grow the number of computer science graduates and software engineers.

Here's why.

Developer dropout is real

Software development has a *serious* retention problem:

- At age 26, 59% of engineering and computer science grads work in occupations *related* to their field of study. By age 50, only 41% work in the same domain, meaning a full **~30% drop out of the field by mid-career**
- In contrast, engineering and computer science majors who join *unrelated* fields upon graduation retain at much higher rates, with only 10-15% switching out after the age of 26:

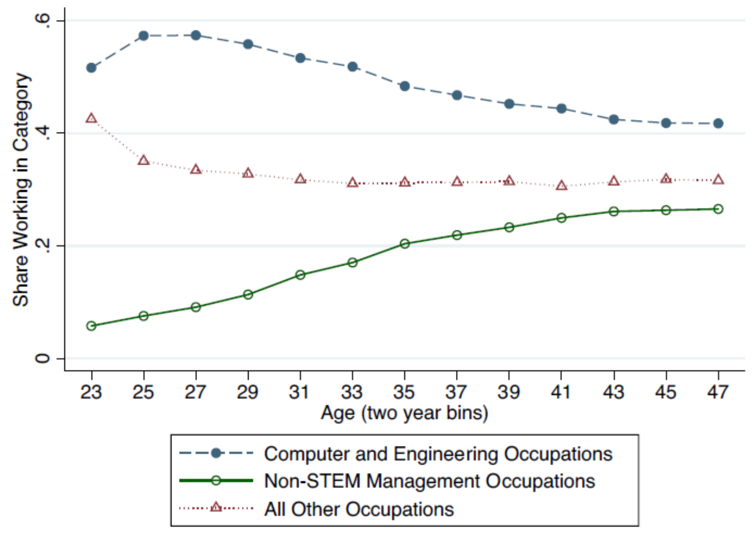


FIGURE VIII

Occupational Sorting by Age for Engineering/CS Majors

The figure plots coefficients from three separate regressions of indicators for working in the labeled occupation category on two-year age bins plus controls for sex-by-age indicators, year fixed effects, race and ethnicity, citizenship, veteran status, and an indicator for having any graduate school education. The sample is all full-time working four-year college graduates aged 23–50 in the 2009–2017 American Community Survey, who also majored in computer science or engineering. Computer and engineering occupations are two-digit SOC codes 15 and 17. Non-STEM management is two-digit SOC code 11, except a small number of codes indicating management in computer or engineering fields. See the text for details.

Engineers often leave engineering for non-STEM management roles. Graduation into management is not surprising. What's surprising is that these are **non-STEM** positions. Engineers swap technical roles for *non-technical* roles over time.

This phenomenon, which I'll call "**developer dropout**," is a real problem. What's behind it?

Out with the old skills, in with the new skills

Programming-related jobs have high rates of skill turnover. Over time, the types of skills required by companies hiring software developers change more rapidly than any other profession.

To demonstrate this, researchers analyzed job postings on more than 40,000 online job boards and company websites between 2007 and 2019, controlling for employer, location, and occupation. They defined "new" skills as those that were rare or non-existent in 2007 but prevalent in 2019 and "old" skills as those that were prevalent in 2007 but rare or extinct in 2019.

- While only 30% of all job vacancies required at least one new skill by 2019, **47% of computer and mathematical jobs required at least one new skill** (i.e. a skill that was not common back in 2007)
- This compares to *less than 20%* of jobs in fields like education, law, and community and social services

- In addition, **16% of jobs in computer and mathematical fields in 2007 required a skill that was obsolete by 2019** (i.e. a skill that was common in 2007 but relatively rare in 2019), more than double any other job category:

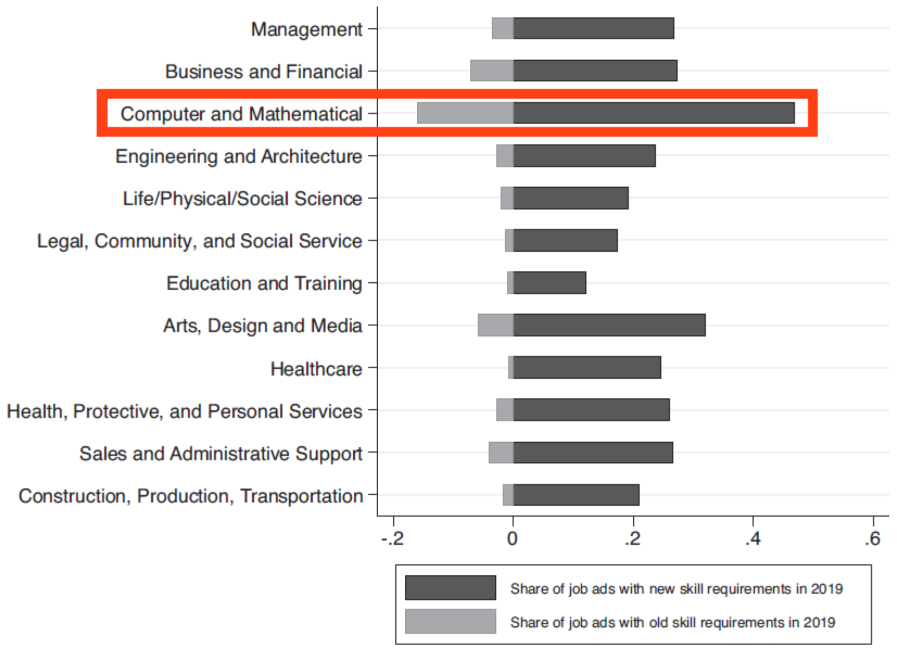


FIGURE I

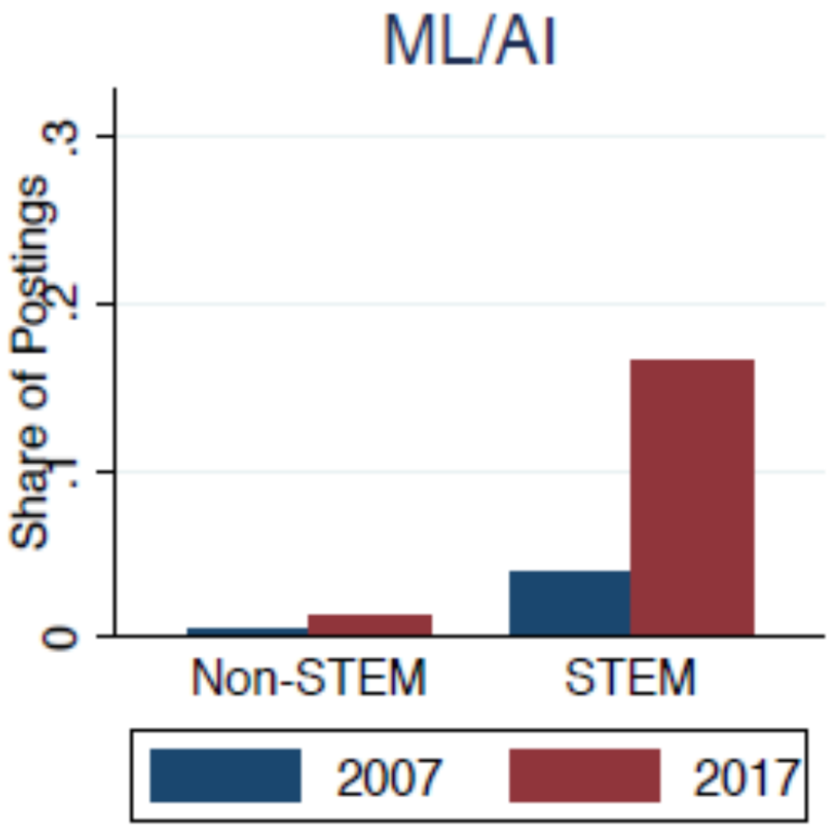
Turnover of Skill Requirements by Occupation Category

The bars show the share of jobs in each occupation category that required an "old" skill in 2007 (the light gray bars) and a "new" skill in 2019 (the black bars). Old skills are defined as those with at least 1,000 appearances in 2007 but are either five times less frequent or do not exist in 2019. New skills are defined as those with at least 1,000 appearances in 2019 that either did not exist in 2007 or are 20 times more frequent in 2019 than 2007. The values of each bar are coefficients from a vacancy-level regression of the frequency of old and new skill requirements on an indicator for 2019, the total number of skills listed in each vacancy, education and experience requirements, and occupation-city-employer fixed effects. Occupations are grouped according to two-digit Standard Occupation Classification (SOC) codes. Some two-digit SOC codes are grouped together to conserve space (see text for details).

About a third of the change in required skills in computer-related occupations is due to specific new software:

- The fastest-growing software skills between 2007 and 2019 include **Python, R, and Apache Hadoop**
- Software that was popular in 2007 but effectively obsolete by 2019 includes QuarkXpress, ActionScript, Solaris, IBM Websphere, and Adobe Flash (ah, finally a name I recognize)

Data science, machine learning, and AI saw big increases among technology-intensive jobs as well. For example, the number of STEM-related jobs requiring skills in machine learning and AI grew more than 4x from 2007-2017, touching more than 15% of STEM jobs:



To better compare rates of skill change across occupations, the researchers came up with a measure of skill change that tracks the absolute growth or decline of various skills within each profession from 2007 to 2019. Occupations whose required skills change rapidly in prevalence among job postings receive a high score, while jobs whose skills do not change much receive a lower score:

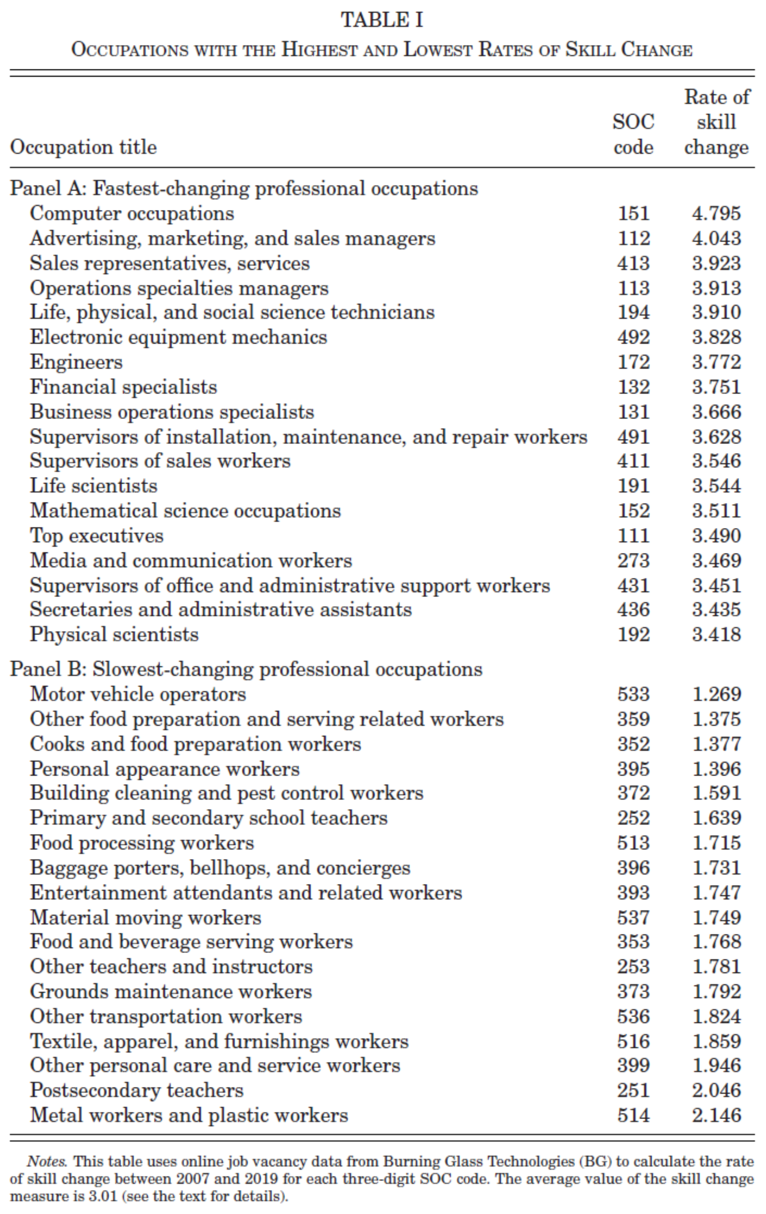


TABLE I

OCCUPATIONS WITH THE HIGHEST AND LOWEST RATES OF SKILL CHANGE

Occupation title	SOC code	Rate of skill change
Panel A: Fastest-changing professional occupations		
Computer occupations	151	4.795
Advertising, marketing, and sales managers	112	4.043
Sales representatives, services	413	3.923
Operations specialties managers	113	3.913
Life, physical, and social science technicians	194	3.910
Electronic equipment mechanics	492	3.828
Engineers	172	3.772
Financial specialists	132	3.751
Business operations specialists	131	3.666
Supervisors of installation, maintenance, and repair workers	491	3.628
Supervisors of sales workers	411	3.546
Life scientists	191	3.544
Mathematical science occupations	152	3.511
Top executives	111	3.490
Media and communication workers	273	3.469
Supervisors of office and administrative support workers	431	3.451
Secretaries and administrative assistants	436	3.435
Physical scientists	192	3.418
Panel B: Slowest-changing professional occupations		
Motor vehicle operators	533	1.269
Other food preparation and serving related workers	359	1.375
Cooks and food preparation workers	352	1.377
Personal appearance workers	395	1.396
Building cleaning and pest control workers	372	1.591
Primary and secondary school teachers	252	1.639
Food processing workers	513	1.715
Baggage porters, bellhops, and concierges	396	1.731
Entertainment attendants and related workers	393	1.747
Material moving workers	537	1.749
Food and beverage serving workers	353	1.768
Other teachers and instructors	253	1.781
Grounds maintenance workers	373	1.792
Other transportation workers	536	1.824
Textile, apparel, and furnishings workers	516	1.859
Other personal care and service workers	399	1.946
Postsecondary teachers	251	2.046
Metal workers and plastic workers	514	2.146

Notes. This table uses online job vacancy data from Burning Glass Technologies (BG) to calculate the rate of skill change between 2007 and 2019 for each three-digit SOC code. The average value of the skill change measure is 3.01 (see the text for details).

- **Computer-related occupations receive the highest score by far, 4.8**. Note that the mean and standard deviation of this measure are ~3 and ~1 respectively, so computer-related jobs are **nearly two standard deviations away from the typical job in America**
- Meanwhile, jobs in education and and those involving manual labor have very low skill change scores, typically less than 2.

We can get even more granular and look at specific job roles. This level of detail makes the difference even more stark (only showing the fastest changing roles):

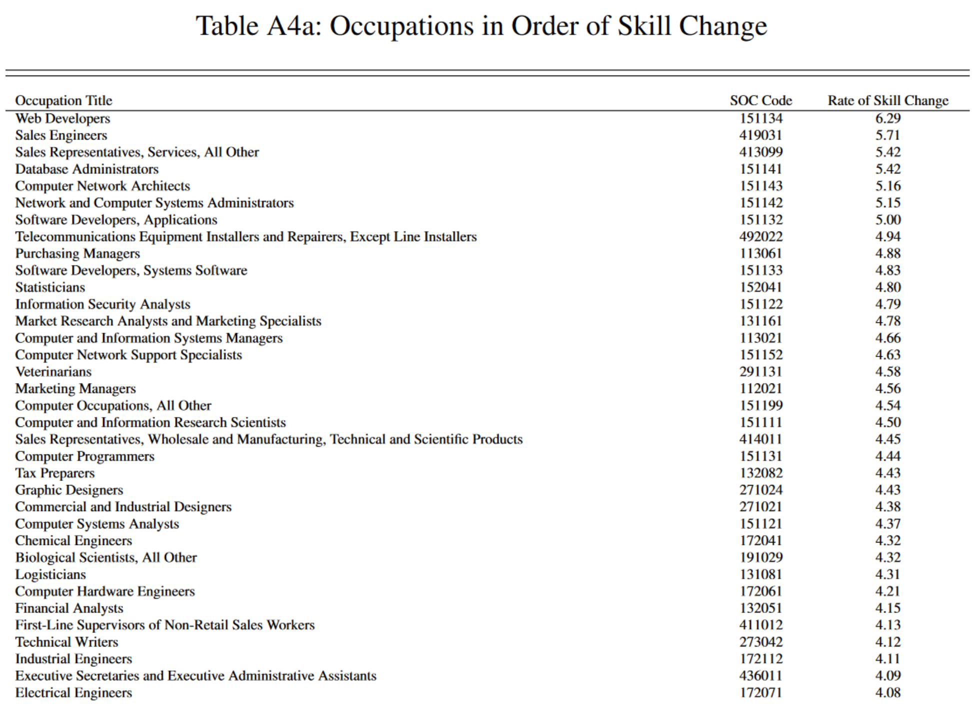


Table A4a: Occupations in Order of Skill Change

Occupation Title	SOC Code	Rate of Skill Change
Web Developers	151134	6.29
Sales Engineers	419031	5.71
Sales Representatives, Services, All Other	413099	5.42
Database Administrators	151141	5.42
Computer Network Architects	151143	5.16
Network and Computer Systems Administrators	151142	5.15
Software Developers, Applications	151132	5.00
Telecommunications Equipment Installers and Repairers, Except Line Installers	492022	4.94
Purchasing Managers	113061	4.88
Software Developers, Systems Software	151133	4.83
Statisticians	152041	4.80
Information Security Analysts	151122	4.79
Market Research Analysts and Marketing Specialists	131161	4.78
Computer and Information Systems Managers	113021	4.66
Computer Network Support Specialists	151152	4.63
Veterinarians	291131	4.58
Marketing Managers	112021	4.56
Computer Occupations, All Other	151199	4.54
Computer and Information Research Scientists	151111	4.50
Sales Representatives, Wholesale and Manufacturing, Technical and Scientific Products	414011	4.45
Computer Programmers	151131	4.44
Tax Preparers	132082	4.43
Graphic Designers	271024	4.43
Commercial and Industrial Designers	271021	4.38
Computer Systems Analysts	151121	4.37
Chemical Engineers	172041	4.32
Biological Scientists, All Other	191029	4.32
Logisticians	131081	4.31
Computer Hardware Engineers	172061	4.21
Financial Analysts	132051	4.15
First-Line Supervisors of Non-Retail Sales Workers	411012	4.13
Technical Writers	273042	4.12
Industrial Engineers	172112	4.11
Executive Secretaries and Executive Administrative Assistants	436011	4.09
Electrical Engineers	172071	4.08

Web development has the highest rate of skill change *among all jobs in the country*. Next up are sales engineers, another often technical role. Database administrators, computer network architects, sysadmins, and application developers all make the top 10, and we see many other technical roles among the top 30. The mean and standard deviation are similar here, placing web development **more than 3 standard deviations away from the typical job in America** in terms of skill change over time.

Suffice to say, **software development is a rapidly changing profession**.

You might think, however, that skill change would eventually settle down as one becomes more experienced.

You'd be wrong. The skills for software engineering jobs change rapidly throughout the entire career lifecycle:

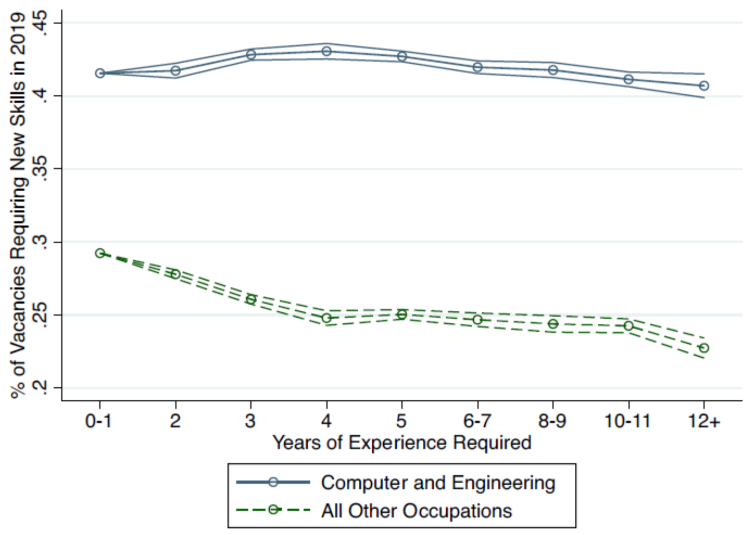


FIGURE II

New Skills Required by Job Experience

This figure shows how new skill requirements change along with required years of experience in computer and engineering occupations (defined as SOC two-digit codes 15 and 17), compared with all other occupations. Each point in the figure is the coefficient (and associated 95% confidence interval) on the relevant experience category from a vacancy-level regression of the frequency of new skill requirements on experience categories, the total number of skills listed in the vacancy, education requirements, and employer-by-MSA fixed effects. New skills are defined as those with at least 1,000 appearances in 2019 that either did not exist in 2007 or are 20 times more frequent in 2019 than 2007.

- In entry-level roles in computer and engineering occupations all the way through those requiring 12+ years of experience, **the proportion of job postings requiring at least one new skill in 2019 was effectively the same, 40-45%**
- In contrast, **29% of entry-level non-computing and engineering roles in 2019 required at least one new skill, but this proportion declines to 24%** for jobs requiring more than four years of experience

> *This means that experienced STEM workers seeking employment in 2019 are often required to possess skills that **were not required** when they entered the labor market in 2007 or earlier.*

Software engineers **never** escape the skill-change vortex, even many years into their careers. Experienced engineers must learn and adopt technologies that didn't even exist when they started out. Developers must constantly retool themselves, even well after their formal education ends.

Nothing's changed but my change

College majors associated with faster changing jobs pay more early on.

- In professions with one standard deviation increased skill change, pay is **~30%** higher in the first few years of one's career
- If we exclude both the fastest and slowest-changing fields (Engineering/Computer Science at the high end, Health/Education at the low end), the early earnings premium for faster-changing roles increases to **~60%**:

Fast-changing fields pay better.

Notice however that the pay advantage declines over time. By the time one approaches the age of 50, the pay premium for working in rapidly changing fields falls dramatically to only 20-30% vs slower changing professions.

Here's another way to see the eroding pay advantage. The below chart simulates the earnings of the average worker by category of college degree from ages 23 to 50 in 2016 dollars.

- Computer science and engineering grads start off with sizable advantage vs any other major
- However, this premium *falls* over time as the earnings of CS and engineering graduates plateau over time while the earnings of their peers grow *faster* for *longer*
- In fact, **life and physical science graduates' earnings surpass their computer and engineering classmates by the age of 40**:

Excluding business majors, the earnings premium of software engineering declines over time in both percentage *and* absolute dollar terms, to the point where engineers barely out-earn social science majors:

But the focus on college major is somewhat misleading. This phenomenon has less to do with one's field of study and more to do with *choice of occupation*.

To show this, researchers plotted the earnings premium of various categories workers relative to those with a non-Engineering/Computer Science major working in a non-Engineering/Computer Science job.

- Workers who major in Engineering or Computer Science but work in unrelated fields actually see their earnings advantage *compound* over time, rather than decline
- On the other hand, regardless of major, individuals who work in Engineering or Computer Science jobs see their earnings advantage erode over the years:

> ***Declining relative returns is a feature of STEM jobs, not majors.*** *The earnings premium for non-STEM majors in STEM occupations starts off near 40%, but declines to 20% within a decade. In contrast, the relative earnings advantage grows over time for computer science and engineering majors working in non-STEM occupations.*

The **profession** of software development drives the declining earnings premium, **not the college major**.

In fact, computer science majors who work in non-CS fields experience the *opposite* dynamic of their non-developer peers — their relative earnings premium rises as they advance. A CS major who eschews the profession doesn't earn much more than otherwise similar non-CS majors early on, but eventually out-earns their peers by nearly 20%.

OK, that's enough about *what* is happening. Now let's see *why* it's happening.

Human capital depreciates too

Imagine a simple model where workers choose their profession in order to maximize income, which is a derivative of their own skill or human capital. Over time, workers gain new skills, while the value of their existing skills depreciates somewhat due to changing times.

Some workers, endowed with superior ability, learn faster than others, picking up skills at a quicker pace. Those workers will tend to sort into high-skilled, fast-changing professions initially, maximizing their early career earnings. Less impressive workers will sort into low-skilled, slower-changing professions.

In a world where human capital never depreciated, we could imagine that high-skilled individuals like software developers would maintain a relative human capital (and earnings) advantage over other professionals, leading to consistently increasing pay and a stable relative premium:

But, if human capital depreciates over time and that rate of depreciation is higher in rapidly-changing fields like software development, then developers' initial advantage would erode over time, narrowing the gap vs. non-developers:

This simple model helps explain what we see in the data — the software engineering earnings advantage disappears as the *effective* human capital gap narrows.

> *Applied majors such as computer science, engineering, and business teach vintage-specific skills that become less valuable as new skills are introduced to the workplace over time.*

Specific skills in software development quickly become dated. Programming languages and development frameworks go out of style. Hadoop is hot one year, and it's old news the next. Like a fast, expensive car that quickly loses value as it's driven around town, the skills and human capital of software engineers fall apart without constant, expensive maintenance:

> *Intuitively, careers with high rates of obsolescence require workers to learn many new tasks each year, which **diminishes learning gains and lowers the returns to experience**.*

Quick learners are fast dropouts

The hits don't stop there. Ironically, **the quickest learners are also the quickest dropouts**.

To understand why, think back to the model we just outlined. Quick learners accumulate human capital faster than their slower peers, which means they have the most to lose when certain skills or abilities fall out of favor. In fact, **the return to being a fast learner is *higher* in jobs with *low* rates of skill change** because the learnings can compound over time instead of dwindle in relevance.

> *High-ability workers are faster learners, in all jobs. However, **the relative return to ability is higher in careers that change less, because learning gains accumulate.***

Said another way, **the opportunity cost of working in a rapidly-changing field is highest for the best learners**. This creates immense pressure to drop out of software engineering and other fast-changing careers into more stable roles and industries.

The researchers show this by regressing STEM job status on a number of other variables, including an interaction between age and score on the Armed Forces Qualifying Test (AFQT), a common measure of cognitive ability. The coefficient comes out negative and statistically significant, implying that the relative probability of working in STEM at any given age *declines* with cognitive ability (column 1 and 2 below):

> *The results imply that a worker with cognitive ability one standard deviation above average is 4.9 percentage points more likely to work in STEM at age 23, but **only 1.6 percentage points more likely to be working in a STEM job by age 34**.*

In fact, by age 40, the regression predicts **workers with higher cognitive ability are *less* likely to work in STEM than those with lower cognitive ability**.

Simply put, **professionals with higher cognitive ability drop out of STEM careers earlier and faster**:

> *... STEM majors with higher scores on the Armed Forces Qualifying Test (AFQT)—a widely used proxy for academic aptitude—**leave STEM careers more often and at younger ages**.*

Implication: Growing the software engineering talent pool is harder than you think

Two points I want to drive home:

First — software development has a *turnover* problem.

Growing the supply of software developers is not trivial because the field already sees high levels of developer dropout and turnover, and this would only increase if the field were to grow larger. A larger software development labor pool would presumably drive down wages, encouraging even more developers to shift out of the profession, especially those past their career midpoints. On that flat part of the earnings curve, the incentive to remain a developer is weak at best.

Second — software development also has a *selection* problem.

The highest ability, fastest learners disproportionately leave the field over time. They have a multitude of other ways to profitably leverage their intellect and skills. **Software development carries serious opportunity cost.** Again, this is ironic because one would normally expect the best to stay and worst to leave, but that's not what we see in the data. The software development talent pool mix shifts toward *lower cognitive ability* as any given cohort matures.

Combined, these points suggest software development may be destined for perennial labor shortages unless the pace of change slows sufficiently.

> *... **rapid technological change can lead to a short shelf life for technical skills**. This tradeoff between technology-specific and general skills is an important consideration for policymakers and colleges seeking to educate the workers of today, while also building the skills of the next generation.*

To conclude, I emphasize: **highly skilled people prefer highly stable careers in the long-run**. This lets their relative ability and human capital advantage compound over time. Rapid deterioration of skills continuously levels the playing field, preventing the best from separating themselves from the pack. In such a situation, **it makes more sense to quit the race early than get caught in the pile up**.

Thanks to David Deming, Kadeem Noray, the authors of the study from which much of this essay is derived.