IT Systems Engineering in the Age of AI

Jürgen Döllner

Hasso-Plattner-Institute

September 24, 2025

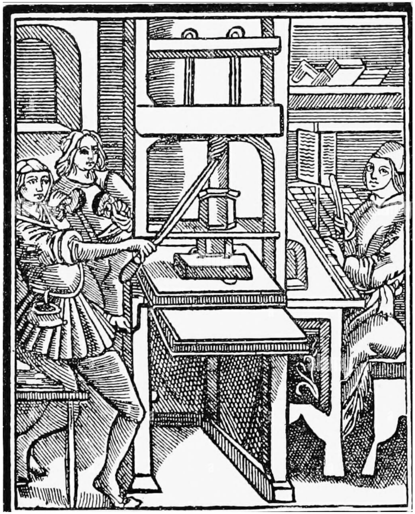
Introduction

- "Programming will be obsolete. I believe the conventional idea of 'writing a program' is headed for extinction, and indeed, for all but very specialized applications, most software, as we know it, will be replaced by AI systems that are *trained* rather than *programmed*.

Disruptive Innovations

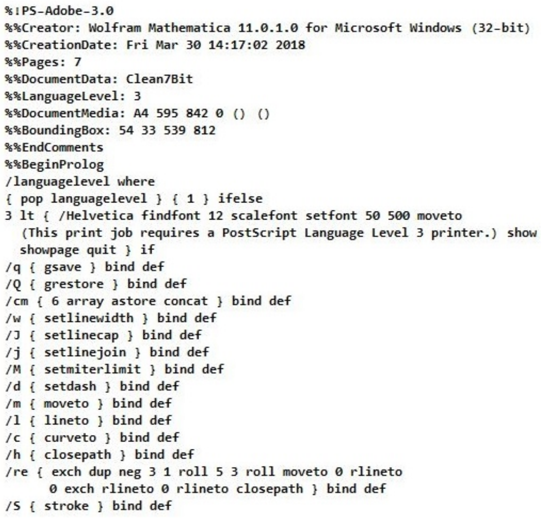
Scriptorium Era: The illustration shows work in a scriptorium using the example of the ‘Miracles de Notre Dame’ (1462) for manual book production.

Disruptive Innovation



Gutenberg Bookpress: The illustration shows Gutenberg's first book press, which made mass production of printed information possible for the first time. This triggered an unimaginable boom in business and science.

Disruptive Innovations



```
%!PS-Adobe-3.0
%%Creator: Wolfram Mathematica 11.0.1.0 for Microsoft Windows (32-bit)
%%CreationDate: Fri Mar 30 14:17:02 2018
%%Pages: 7
%%DocumentData: Clean7Bit
%%LanguageLevel: 3
%%DocumentMedia: A4 595 842 0 () ()
%%BoundingBox: 54 33 539 812
%%EndComments
%%BeginProlog
/languagelevel where
{ pop languagelevel } { 1 } ifelse
3 lt { /Helvetica findfont 12 scalefont setfont 50 500 moveto
  (This print job requires a PostScript Language Level 3 printer.) show
  showpage quit } if
/q { gsave } bind def
/Q { grestore } bind def
/cm { 6 array astore concat } bind def
/w { setlinewidth } bind def
/J { setlinecap } bind def
/j { setlinejoin } bind def
/M { setmiterlimit } bind def
/d { setdash } bind def
/m { moveto } bind def
/l { lineto } bind def
/c { curveto } bind def
/h { closepath } bind def
/re { exch dup neg 3 1 roll 5 3 roll moveto 0 rlineto
      0 exch rlineto 0 rlineto closepath } bind def
/S { stroke } bind def
```

Postscript and Digital Printing: PostScript as a universal description language for page-structured 2D information (John Warnock, Adobe, 1982) – silent omnipresence. PostScript and PDF democratized access to information, removed technical barriers, and ensured that documents could be created, shared, and printed anywhere in the world with absolute fidelity.

Disruptive Innovations

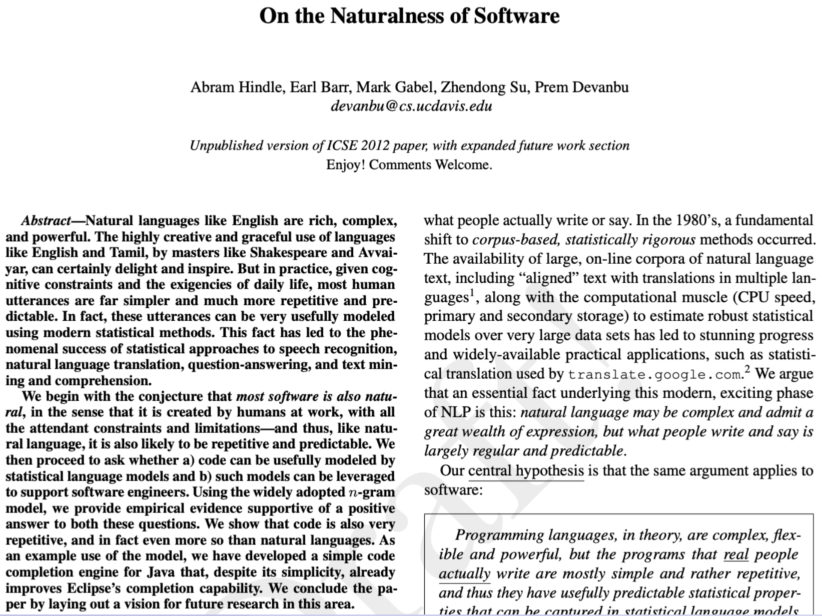
- Initially limited
- Foreseeably cheaper or simpler
- Mass adoption in the long run

- Radical change to existing processes
- New target groups and markets

LLMs

- Language creates and shapes thinking
- LLMs become operating systems for thinking
- Wittgenstein: the meaning of a word is its use in the language ("language games")

Naturalness of Software



On the Naturalness of Software

Abram Hindle, Earl Barr, Mark Gabel, Zhendong Su, Prem Devanbu
devanbu@cs.ucdavis.edu

Unpublished version of ICSE 2012 paper, with expanded future work section
Enjoy! Comments Welcome.

***Abstract*—Natural languages like English are rich, complex, and powerful. The highly creative and graceful use of languages like English and Tamil, by masters like Shakespeare and Avvaiyar, can certainly delight and inspire. But in practice, given cognitive constraints and the exigencies of daily life, most human utterances are far simpler and much more repetitive and predictable. In fact, these utterances can be very usefully modeled using modern statistical methods. This fact has led to the phenomenal success of statistical approaches to speech recognition, natural language translation, question-answering, and text mining and comprehension.**

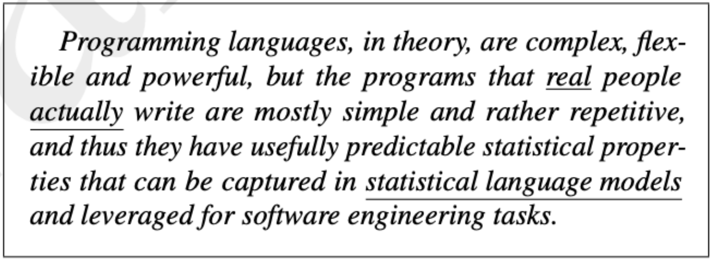
We begin with the conjecture that *most software is also natural*, in the sense that it is created by humans at work, with all the attendant constraints and limitations—and thus, like natural language, it is also likely to be repetitive and predictable. We then proceed to ask whether a) code can be usefully modeled by statistical language models and b) such models can be leveraged to support software engineers. Using the widely adopted n-gram model, we provide empirical evidence supportive of a positive answer to both these questions. We show that code is also very repetitive, and in fact even more so than natural languages. As an example use of the model, we have developed a simple code completion engine for Java that, despite its simplicity, already improves Eclipse's completion capability. We conclude the paper by laying out a vision for future research in this area.

what people actually write or say. In the 1980's, a fundamental shift to *corpus-based, statistically rigorous* methods occurred. The availability of large, on-line corpora of natural language text, including "aligned" text with translations in multiple languages[1], along with the computational muscle (CPU speed, primary and secondary storage) to estimate robust statistical models over very large data sets has led to stunning progress and widely-available practical applications, such as statistical translation used by `translate.google.com`.[2] We argue that an essential fact underlying this modern, exciting phase of NLP is this: *natural language may be complex and admit a great wealth of expression, but what people write and say is largely regular and predictable.*

Our central hypothesis is that the same argument applies to software:

> *Programming languages, in theory, are complex, flexible and powerful, but the programs that real people actually write are mostly simple and rather repetitive, and thus they have usefully predictable statistical properties that can be captured in statistical language models*

Naturalness of Software



Programming languages, in theory, are complex, flexible and powerful, but the programs that real people actually write are mostly simple and rather repetitive, and thus they have usefully predictable statistical properties that can be captured in statistical language models and leveraged for software engineering tasks.

Human-Centered IT Systems Engineering

- Human problem understanding and problem solving
- Human collaboration
- Human problem solving strategies

- Limited scalability
- Proliferation of artifacts
- Increasing complexity of legacy code
- Non-deterministic development processes
- Ongoing so-called "Software Crisis"

Software Crisis

It’s 1968 and there are 10,000 computers in Europe. This number is set to double every year. A group of 40 academics gather for the Garmisch conference. NATO sponsors it so they send a few people along to listen and learn.

Photograph from the conference.

Some of the names we may recognise today are Edsger Dijkstra, CAR Hoare, Alan Perlis, Peter Naur, and Niklaus Wirth. The attendees are largely from a scientific background, and rotate between industry jobs and academia. They typically work on systems such as operating systems, compilers, and programming languages.

1968: The NATO Software Engineering Conference coined the terms software *engineering* and *software crisis* recognizing the complexity and difficulty in software development (isthisit.nz).

Some (Ideas for) Predictions

- Programming languages for machines will differ from those invented by and for humans
- Programming is becoming a step that maps specifications to implementation
- AI agents will test and verify system variants
- Traditional IT systems engineering will continue to be used for niche tasks.
- **Almost everything that has been invented in software development to date is human-centered; almost everything will be redesigned and rethought by AI.**

Evolutionary Perspective