Principles of Software Construction: Objects, Design, and Concurrency

Course Introduction

toad

Spring 2014

Christian Kästner Charlie Garrod

School of Computer Science

isr institute for SOFTWARE RESEARCH

Construction of

Software Systems

at Scale

Libraries
Reuse
Design
I/O, GUI
Analysis
Concurrency

primes

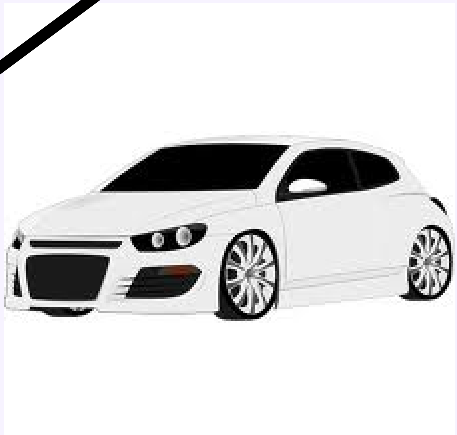
graph search

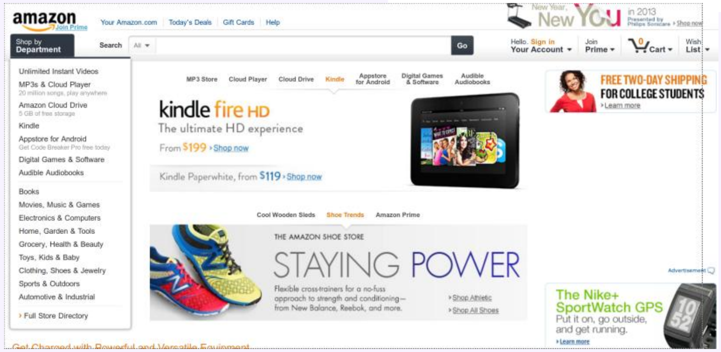
binary tree

GCD

sorting

BDDs




amazon Join Prime | Your Amazon.com | Today's Deals | Gift Cards | Help

New Year, New You in 2013

Shop by Department | Search All | Go | Hello. Sign in Your Account | Join Prime | Cart | Wish List

Unlimited Instant Videos
MP3s & Cloud Player
Amazon Cloud Drive
Kindle
Appstore for Android
Digital Games & Software
Audible Audiobooks
Books
Movies, Music & Games
Electronics & Computers
Home, Garden & Tools
Grocery, Health & Beauty
Toys, Kids & Baby
Clothing, Shoes & Jewelry
Sports & Outdoors
Automotive & Industrial
› Full Store Directory

MP3 Store | Cloud Player | Cloud Drive | Kindle | Appstore for Android | Digital Games & Software | Audible Audiobooks

kindle fire HD
The ultimate HD experience
From $199 › Shop now
Kindle Paperwhite, from $119 › Shop now

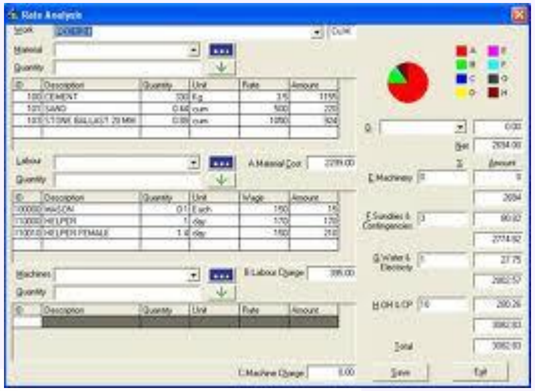
FREE TWO-DAY SHIPPING FOR COLLEGE STUDENTS › Learn more

Cool Wooden Sleds | Shoe Trends | Amazon Prime

THE AMAZON SHOE STORE
STAYING POWER
Flexible cross-trainers for a no-fuss approach to strength and conditioning—from New Balance, Reebok, and more.
› Shop Athletic
› Shop All Shoes

The Nike+ SportWatch GPS
Put it on, go outside, and get running.
› Learn more

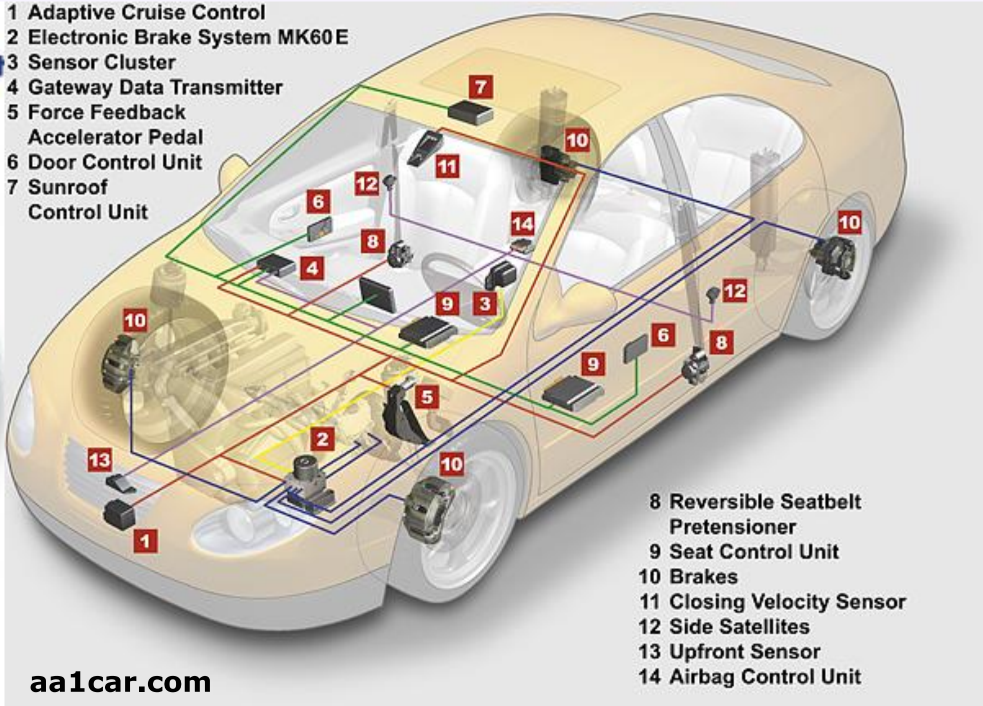
Get Charged with Powerful and Versatile Equipment



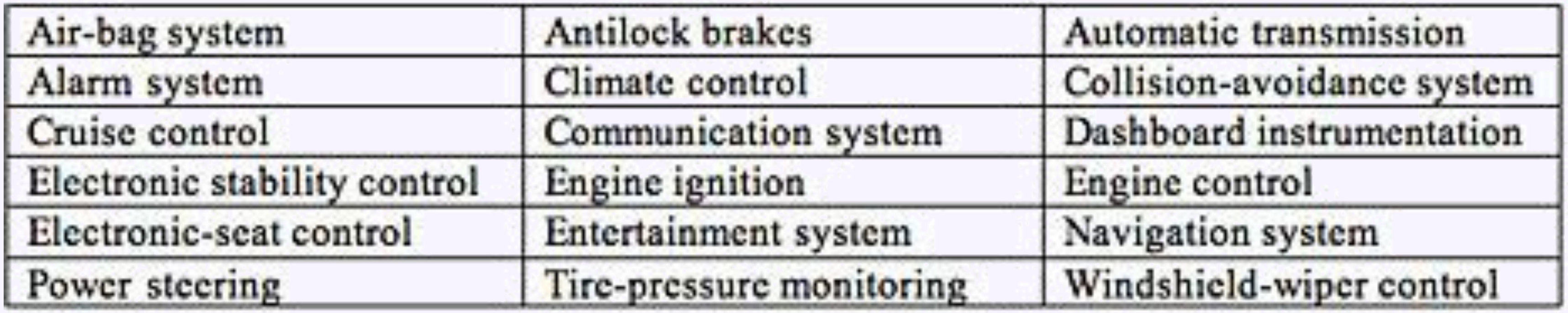
Software and automobiles

Chassis & Safety
Infotainment
Instrument Cluster
Gateways
Powertrain

automotive-eetimes.com

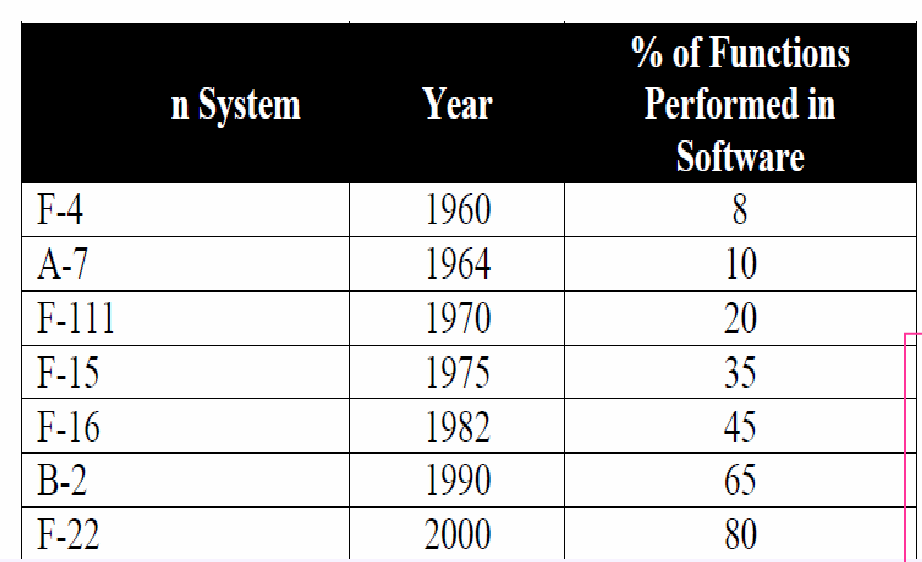

1 Adaptive Cruise Control
2 Electronic Brake System MK60E
3 Sensor Cluster
4 Gateway Data Transmitter
5 Force Feedback Accelerator Pedal
6 Door Control Unit
7 Sunroof Control Unit
8 Reversible Seatbelt Pretensioner
9 Seat Control Unit
10 Brakes
11 Closing Velocity Sensor
12 Side Satellites
13 Upfront Sensor
14 Airbag Control Unit

aa1car.com

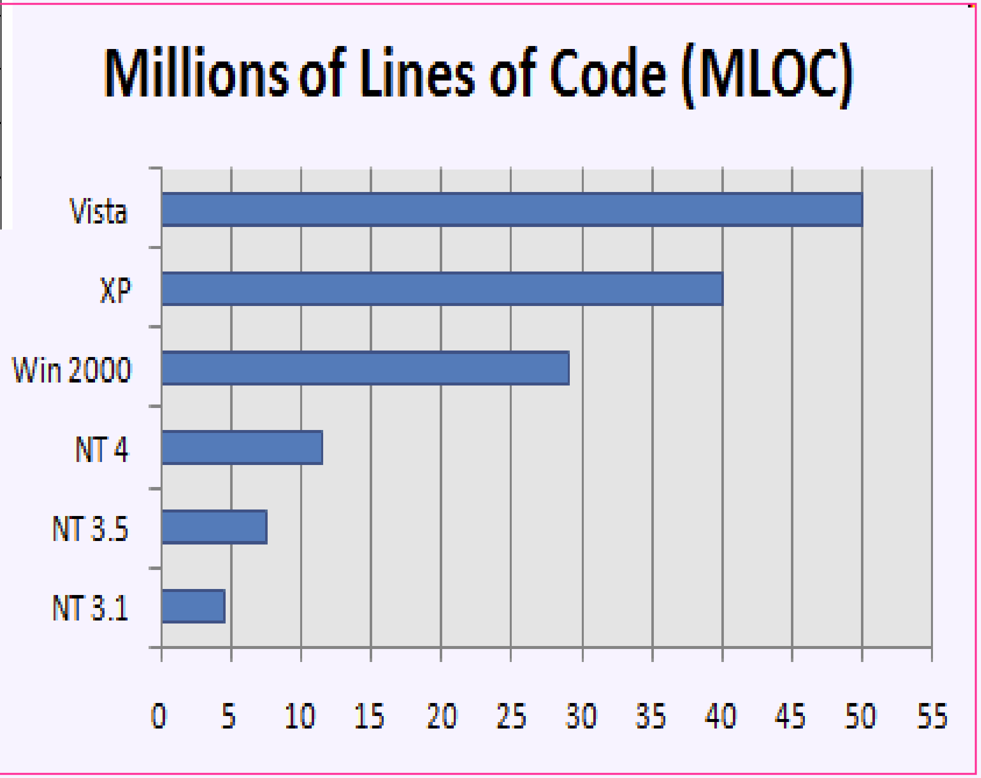


Air-bag system	Antilock brakes	Automatic transmission
Alarm system	Climate control	Collision-avoidance system
Cruise control	Communication system	Dashboard instrumentation
Electronic stability control	Engine ignition	Engine control
Electronic-seat control	Entertainment system	Navigation system
Power steering	Tire-pressure monitoring	Windshield-wiper control

How much software?

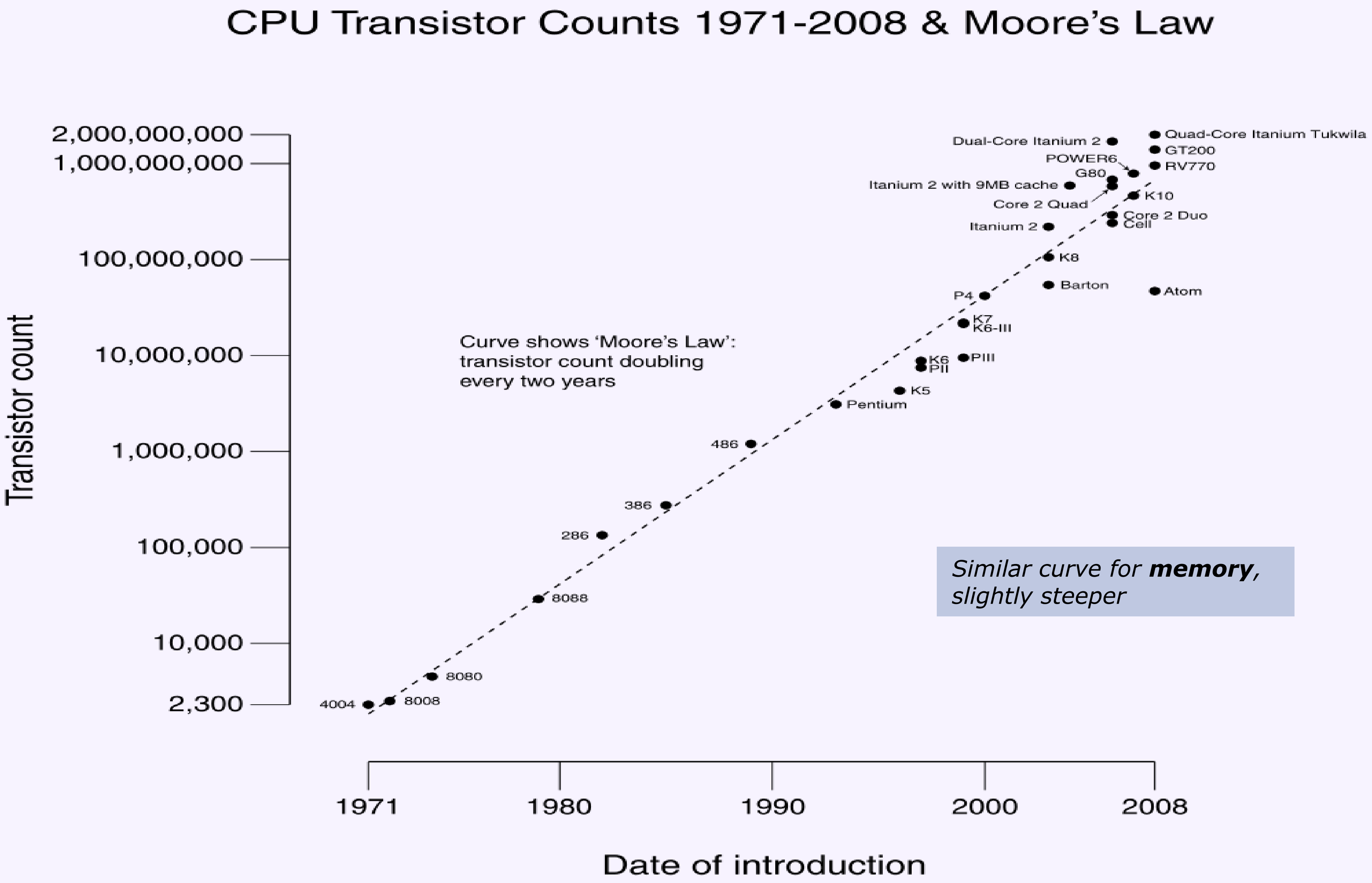


n System	Year	% of Functions Performed in Software
F-4	1960	8
A-7	1964	10
F-111	1970	20
F-15	1975	35
F-16	1982	45
B-2	1990	65
F-22	2000	80



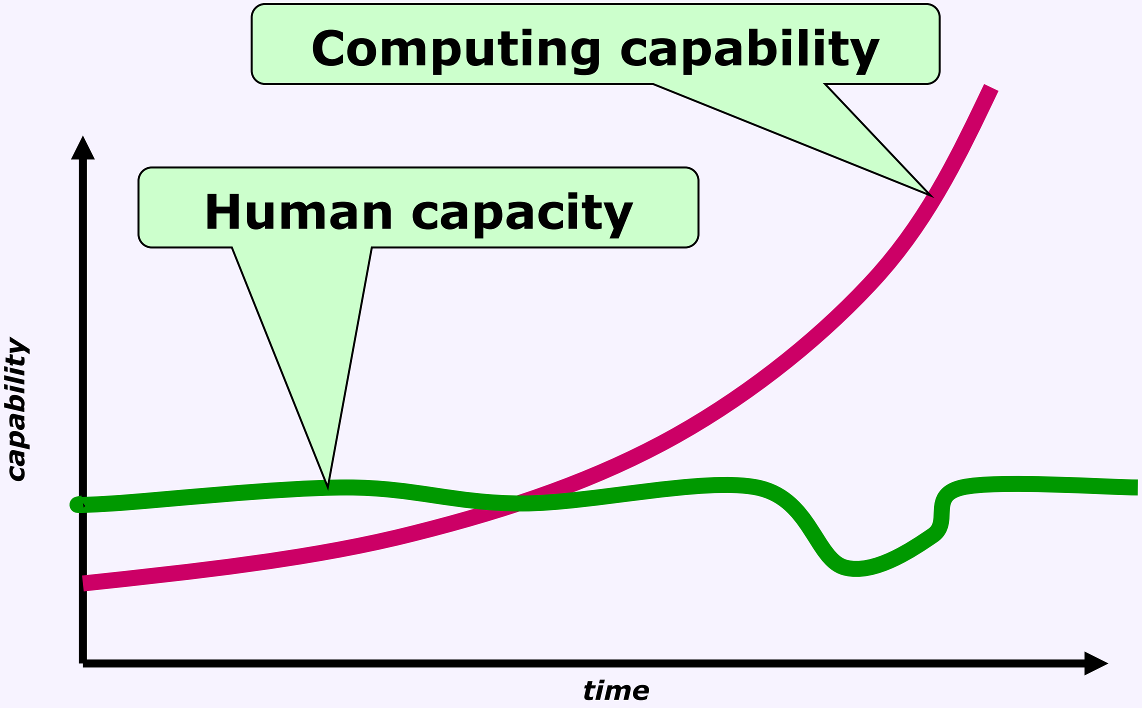
(informal reports)

Moore's Law: transistors per chip



Similar curve for ***memory****,*
slightly steeper

The limits of exponentials



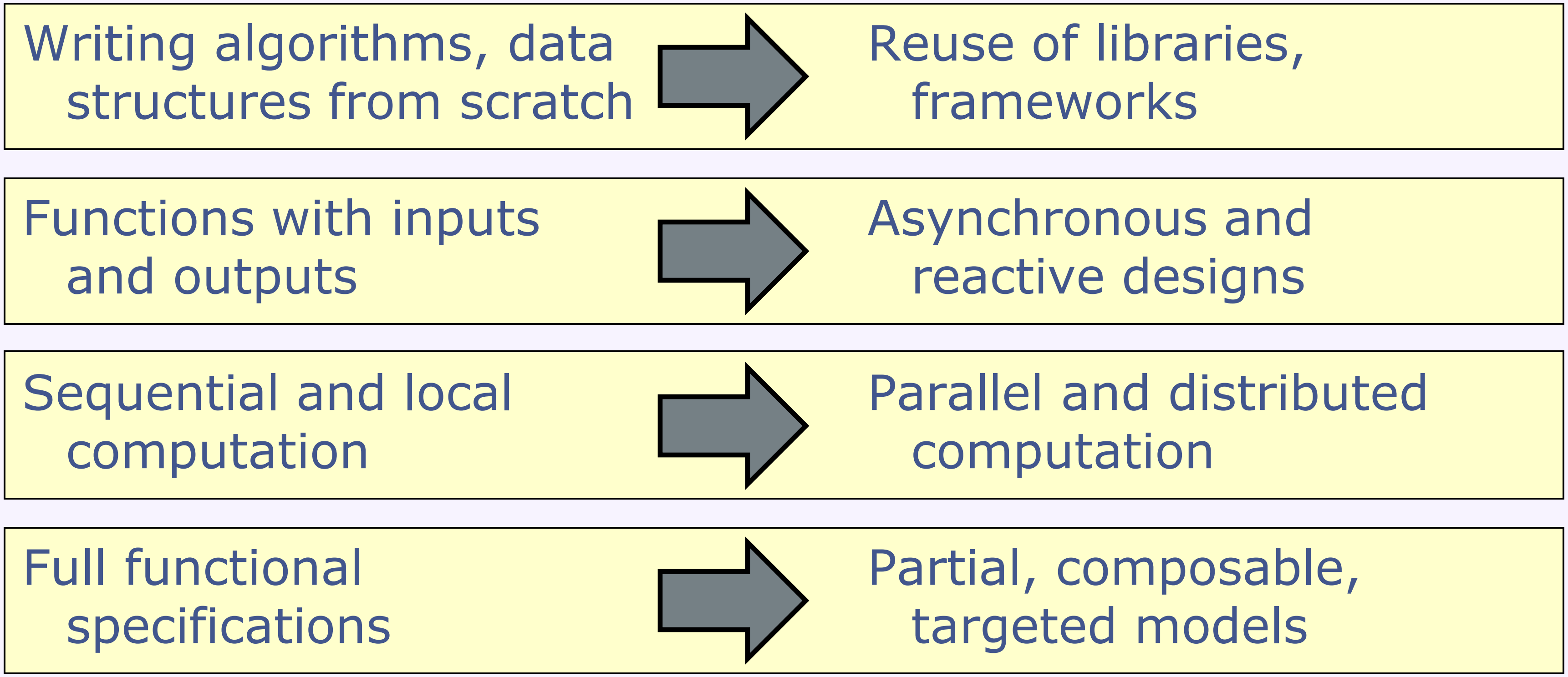
Scaling Up: From Programs to Systems

- You’ve written small- to medium-size programs in 15-122

- This course is about managing software complexity
  - **Scale** of code: KLOC -> MLOC
  - Worldly **environment**: external I/O, network, asynchrony
  - Software **infrastructure**: libraries, frameworks, components
  - Software **evolution**: change over time, design for change
  - Understanding: writing maintainable code
  - Correctness: testing, static analysis

  - In contrast: algorithmic complexity not an emphasis in this course

From Programs to Systems



Writing algorithms, data structures from scratch

Reuse of libraries, frameworks

Functions with inputs and outputs

Asynchronous and reactive designs

Sequential and local computation

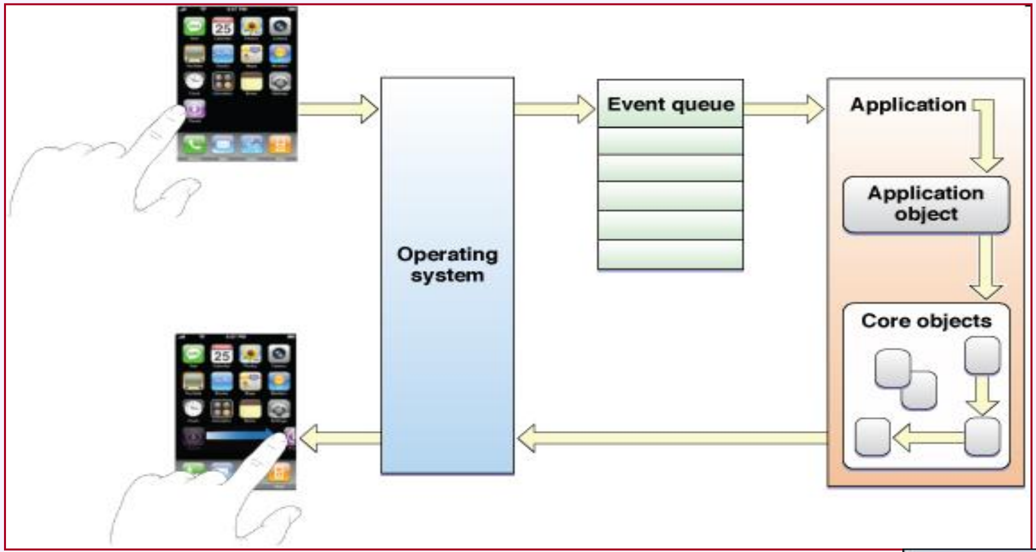
Parallel and distributed computation

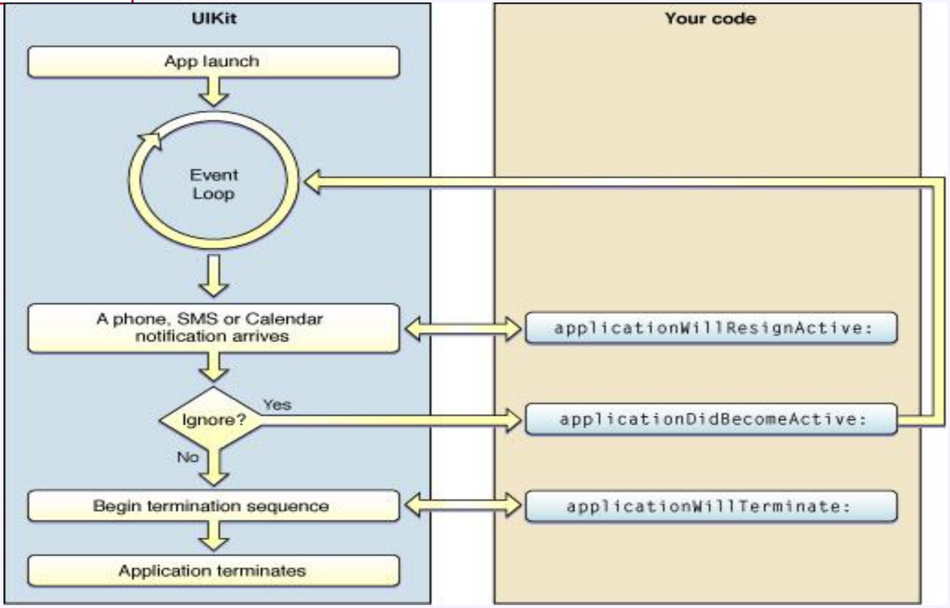
Full functional specifications

Partial, composable, targeted models

Our goal: understanding both the **building blocks** and also the **principles** for construction of software systems at scale

A framework for mobile app software (iOS)




UIKit
Your code
App launch
Event Loop
A phone, SMS or Calendar notification arrives
Ignore?
Yes
No
Begin termination sequence
Application terminates
applicationWillResignActive:
applicationDidBecomeActive:
applicationWillTerminate:

The four course themes

- **T**hreads and Concurrency
  - Concurrency is a crucial system abstraction
    - E.g., background computing while responding to users
  - Concurrency is necessary for performance
    - Multicore processors and distributed computing
  - *Our focus*: application-level concurrency
    - Cf. functional parallelism (150, 210) and systems concurrency (213)
- **O**bject-oriented programming
  - For flexible designs and reusable code
  - A primary paradigm in industry – basis for modern frameworks
  - Focus on Java – used in industry, some upper-division courses
- **A**nalysis and Modeling
  - *Practical* specification techniques and verification tools
  - Address challenges of threading, correct library usage, etc.
- **D**esign
  - Proposing and evaluating alternatives
  - Modularity, information hiding, and planning for change
  - Patterns: well-known solutions to design problems

Motivating example: Virtual Worlds

Discussion: Virtual Worlds

- How can the virtual world to scale to thousands of users?

- How can we organize the system to easily add new things?

- How can we support different kinds of things, while taking advantage of their similarities? (can you think of an example?)

Considering the examples

- **T**hreads and Concurrency
  - In the GUI-based app
  - On game clients
  - On the game servers
- **O**bject-oriented programming
  - Organizing by object types, then actions
- **A**nalysis and Modeling
  - How to gain confidence regarding *all* possible executions
- **D**esign
  - How to organize systems that grow and evolve
  - How to define the interfaces between infrastructure and our code

After 214?

- 214 consists primarily of code-level software engineering, including the design of systems and applications
  - TOAD
- 313: Foundations of software engineering
  - Human and business aspects
  - Plan the process for and manage a software project, manage risk, coordinate teams
  - Elicit, describe, and evaluate a system's requirements
  - Design a software system and evaluate a design with regard to various quality attributes (software architecture)
  - Develop and justify a quality-assurance strategy for a software project (static analysis, inspection, …)
  - Business models and open source
- 413: Software Engineering Practicum (a project course)
- Software Engineering Minor

Toad’s Take-Home Messages

- 214: managing complexity, from programs to systems
  - Threads and concurrency
  - Object-oriented programming
  - Analysis and modeling
  - Design
- Virtual worlds illustrate some challenges
- Object-oriented programming organizes code around **concepts**
  - Methods capture behavior, fields capture state
  - As we will see, this organization allows
    - Greater reuse of concepts
    - Better support for change when concepts vary