AI-Assisted Programming

AI 109

Richard Kelley

Last Time

- Nuts and bolts of computer programs.
- Simple examples of code.
- This level of detail is getting (relatively) less important.

- “Coding in the ***small***”

Today

- How to convert ideas into code: ***design*** and ***architecture.***
- An overview of the general process of creating software

- “Coding in the ***large***”

Next time

- Project 1 goes out
  - Create a GitHub account.
  - Create a public website.
  - Start creating programs for your site (a portfolio).
- Small quiz at the end of class.
  - The steps of the software development life cycle (covered today).

Abstraction

- “Computer Science is a **science of abstraction** - creating the right model for a problem and devising the appropriate mechanizable techniques to solve it.”
  - Alfred Aho
- “People who discover the power and beauty of high-level, abstract ideas often make the mistake of believing that concrete ideas at lower levels are worthless and might as well be forgotten. On the contrary, the best computer scientists are thoroughly grounded in basic concepts of how computers actually work. **The essence of computer science is an ability to understand many levels of abstraction simultaneously.”**
  - Donald Knuth

Making software is hard

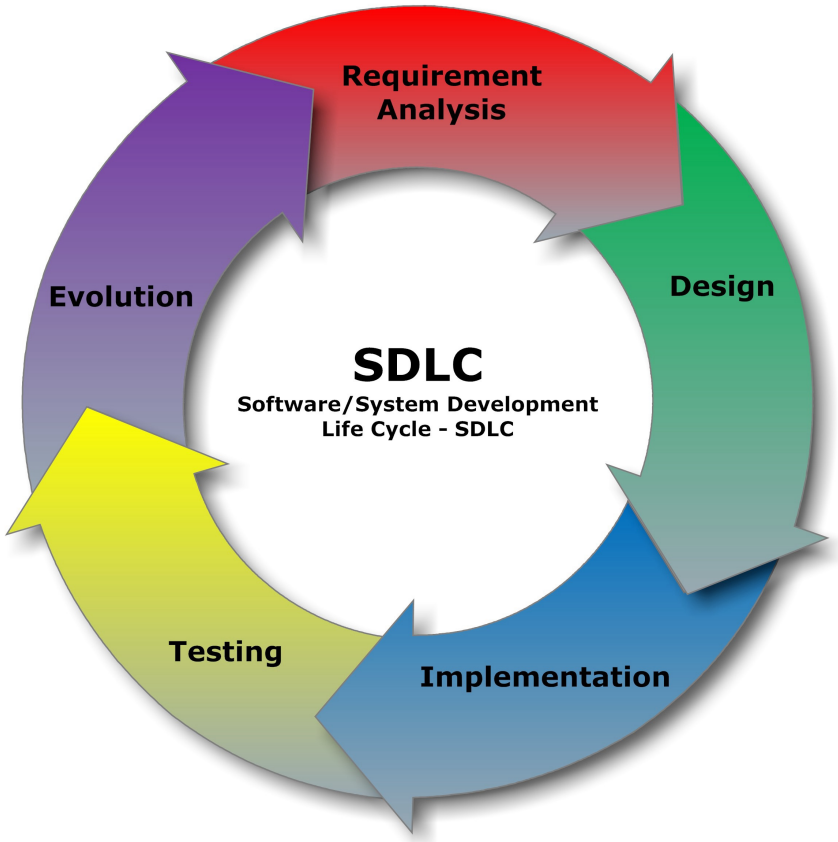
- We saw last time programming languages have lots of parts.
- Writing a program requires knowing those parts and how they fit together.
  - Just like expressing yourself in Latin requires that you know Latin grammar and vocabulary.
- If you make any mistakes, your program breaks.
  - Computers are unforgiving.
- How do we make software when it seems so hard?

Process

The Software Development Lifecycle

The Software Development Lifecycle

- **Requirements & Problem Definition**
- **Design & Planning**
- **Implementation (Coding)**
- **Testing & Verification**
- **Deployment & Release**
- **Maintenance & Evolution**



https://commons.wikimedia.org/wiki/File:SDLC_-_Software_Development_Life_Cycle.jpg

Requirements & Problem Definition

- Identify
  - The problem to be solved,
  - Stakeholders,
  - Constraints, and
  - Success criteria
- Clarify
  - *What* the software should do and
  - *Why*.

Design & Planning

- Translate requirements into
  - Architecture,
  - Data models,
  - Interfaces, and
  - A development plan.
- Decide *how* the system will be built.

Implementation (Coding)

- Write the source code
  - according to the design,
  - using appropriate languages, frameworks, and tools.

Testing & Verification

- Check that the software
  - Behaves correctly and
  - Meets requirements
- Using
  - Unit tests,
  - Integration tests, and
  - System testing.

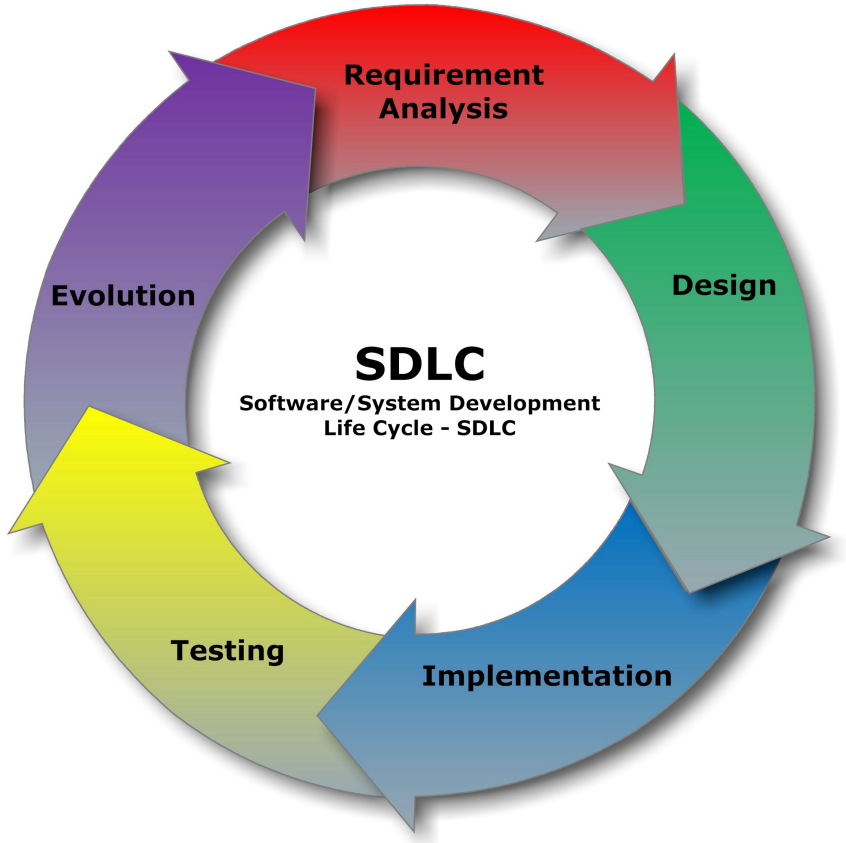
Deployment & Release

- Package and deliver the software to users or production environments.
- configure infrastructure and dependencies.

Maintenance & Evolution

- Fix bugs,
- Improve performance,
- Add features, and
- Adapt the software as requirements or environments change.

Software Development *Lifecycle**



*About half the written resources say “lifecycle” and half say “life cycle”. It doesn’t matter which you use, just be consistent.

https://commons.wikimedia.org/wiki/File:SDLC_-_Software_Development_Life_Cycle.jpg

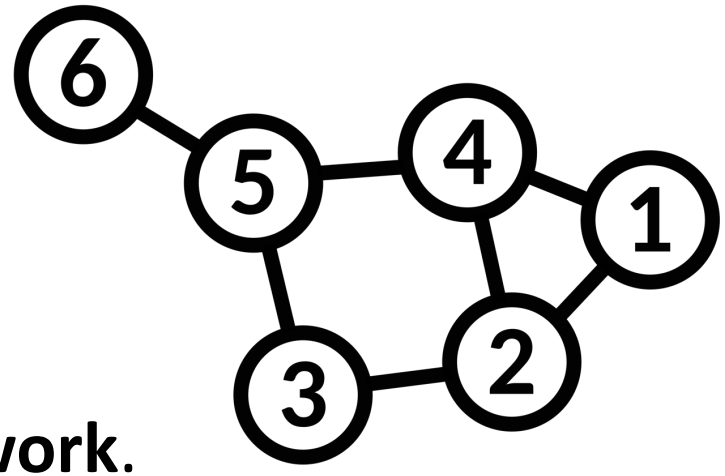
Demonstration

A Common Model: Graphs

- A ***graph*** is a fundamental concept in programming.
- Graphs are *everywhere*.
- BUT
  - Not talking about “graphs” like you may have seen in algebra.
  - Context always makes this clear.
- If you can express a problem in terms of graphs, you can (probably) solve it.

Graphs

- A Graph is a pair of Two things
  - **Nodes** – think of this as points.
    - Nodes may have ***labels***.
  - **Edges** – these are lines that connect nodes.
    - Edges may have **labels**.
    - Edges may have **weights** (numbers) attached to them.

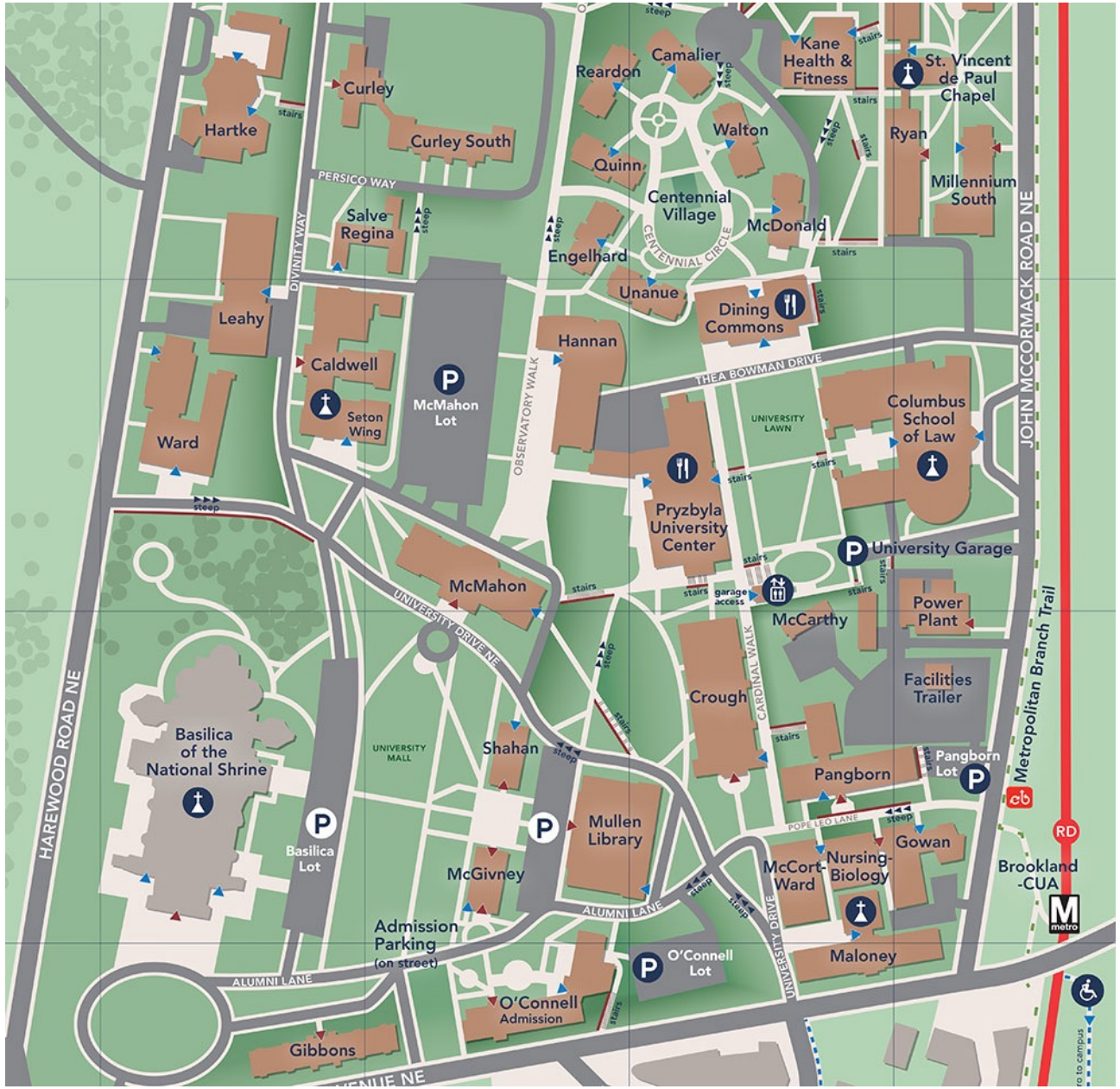


- Another (informal) name for a graph is a **network**.
- Graphs are useful any time you want to talk about relationships between things.
- Many AI algorithms are described in terms of graphs.
- The theoretical study of graphs is called **graph theory**.

Examples of Graphs

- Road/sidewalk networks
- The World Wide Web
- Social Networks
- Power grid
- Airline route map
- Concept similarity graph
- Document similarity network

Maps


Hartke
Curley
Curley South
PERSICO WAY
DIVINITY WAY
Salve Regina
Leahy
Caldwell
Seton Wing
Ward
McMahon Lot
OBSERVATORY WALK
Reardon
Camalier
Quinn
Walton
Centennial Village
CENTENNIAL CIRCLE
Engelhard
McDonald
Unanue
Dining Commons
Hannan
Kane Health & Fitness
St. Vincent de Paul Chapel
Ryan
Millennium South
JOHN MCCORMACK ROAD NE
THEA BOWMAN DRIVE
UNIVERSITY LAWN
Columbus School of Law
Pryzbyla University Center
University Garage
McMahon
UNIVERSITY DRIVE NE
garage access
McCarthy
Power Plant
Facilities Trailer
CARDINAL WALK
Crough
HAREWOOD ROAD NE
Basilica of the National Shrine
Basilica Lot
UNIVERSITY MALL
Shahan
Mullen Library
McGivney
Admission Parking (on street)
ALUMNI LANE
O'Connell Admission
O'Connell Lot
Gibbons
Pangborn
Pangborn Lot
POPE LEO LANE
Gowan
McCort-Ward
Nursing-Biology
Maloney
UNIVERSITY DRIVE
Metropolitan Branch Trail
Brookland-CUA
RD
metro

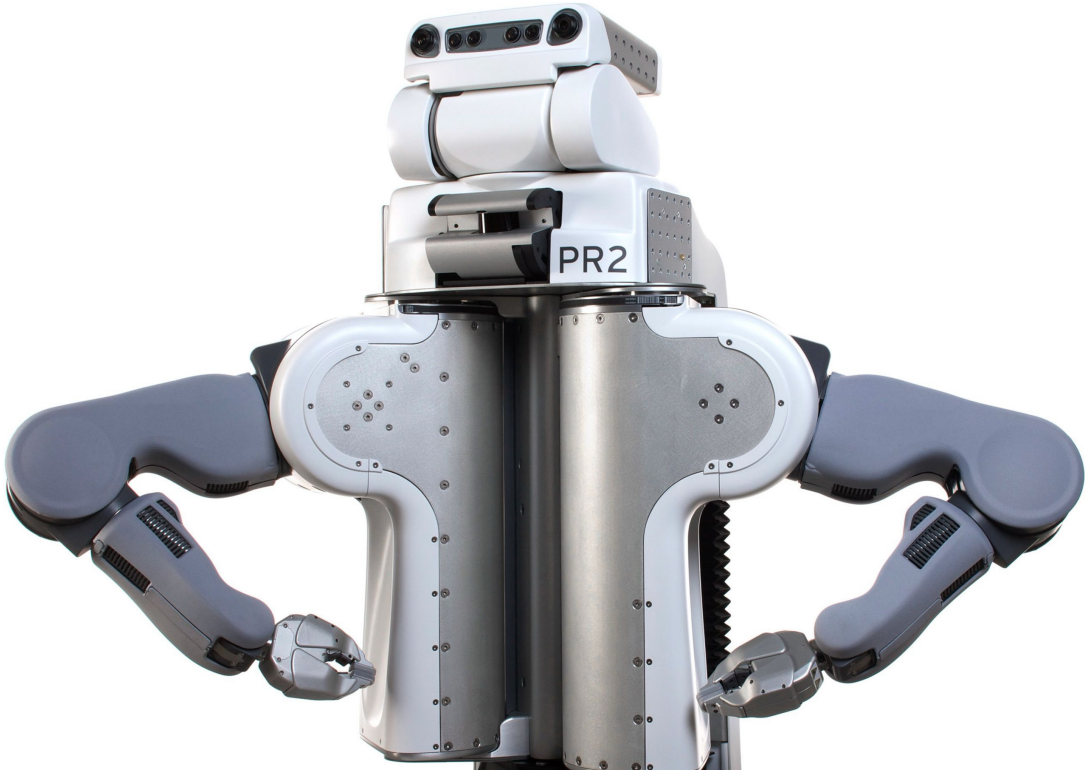
… are graphs

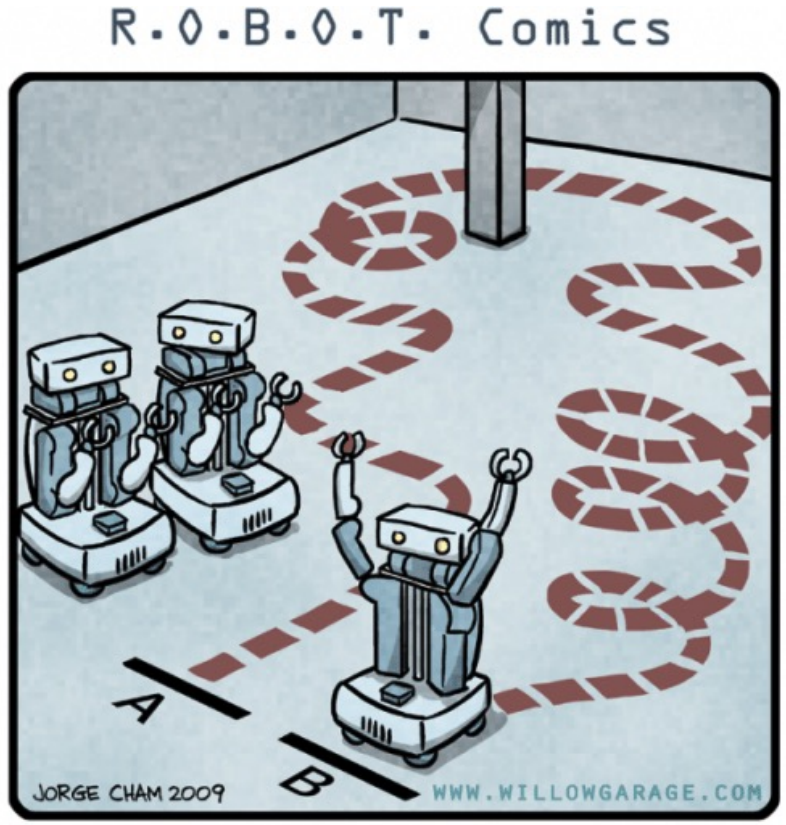

Hartke
Curley
Curley South
Reardon
Camalier
Kane Health & Fitness
St. Vincent de Paul Chapel
Walton
Ryan
Quinn
Millennium South
PERSICO WAY
Centennial Village
Salve Regina
McDonald
CENTENNIAL CIRCLE
Engelhard
Unanue
Dining Commons
Leahy
Hannan
Caldwell
McMahon Lot
Seton Wing
Columbus School of Law
UNIVERSITY LAWN
Ward
Pryzbyla University Center
University Garage
McMahon
McCarthy
Power Plant
Facilities Trailer
Crough
Basilica of the National Shrine
UNIVERSITY MALL
Shahan
Pangborn
Pangborn Lot
Mullen Library
Gowan
McGivney
McCort-Ward
Nursing-Biology
Brookland-CUA
Admission Parking
Maloney
O'Connell Lot
O'Connell Admission
Gibbons
HAREWOOD ROAD NE
JOHN MCCORMACK ROAD NE
Metropolitan Branch Trail
UNIVERSITY DRIVE NE

A Common Problem: Routing

- How do I get from Point A to Point B?
- Called “routing” or “path finding” or “path planning” depending on the context.
- Basic question: ***can*** I get from point A to point B?
- Variation: What’s the **best** way to get from point A to point B?

Routing


PR2


R.O.B.O.T. Comics
A
B
JORGE CHAM 2009
WWW.WILLOWGARAGE.COM
"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

The Problem

- I want the user to be able to take an image (a map) and
  - Create a graph on top of the image.
  - Save and load the graph (and the map).
  - ***Query*** the graph to find routes between points.

- But first we need a bit more clarity on: how do we work with an AI?

How do we communicate with an AI?

- On the web, we can just type questions and get answers.
- In an app (like Cursor), we can make the AI write its thoughts down in ***files***, so that it can refer to them later.
- We can give instructions to the AI to refer to files before it changes anything.
- The predominant format for these files is a popular format called ***markdown***.

Markdown

- Plain-text ***markup language***
- It lets authors add structure
  - headings
  - lists
  - links
  - code blocks and
  - emphasis
- Easy to write and edit without special tools
- Files typically end in “.md”

Markdown

Section headings

```
# Heading 1
## Heading 2
### Heading 3
```

Numbered lists

```
1. first
2. second
```

Bulleted lists

```
- item
- item
  - subitem
```

Emphasis (formatting)

```
*italic*      _italic_
**bold**      __bold__
~~strikethrough~~
```

Hyperlinks

```
[link text](https://example.com)
```

Markdown

Block quoted text

```
> quoted text
```

Inline code

```
`code`
```

Tables

```
| A | B |
|---|---|
| 1 | 2 |
```

Horizontal separator

```
---
```

Block code

````
```javascript
code block
```
````

The Software Development Lifecycle

- **Requirements & Problem Definition**
- **Design & Planning**
- **Implementation (Coding)**
- **Testing & Verification**
- **Deployment & Release**
- ~~Maintenance & Evolution~~