INF 502 – SOFTWARE DEVELOPMENT METHODOLOGIES

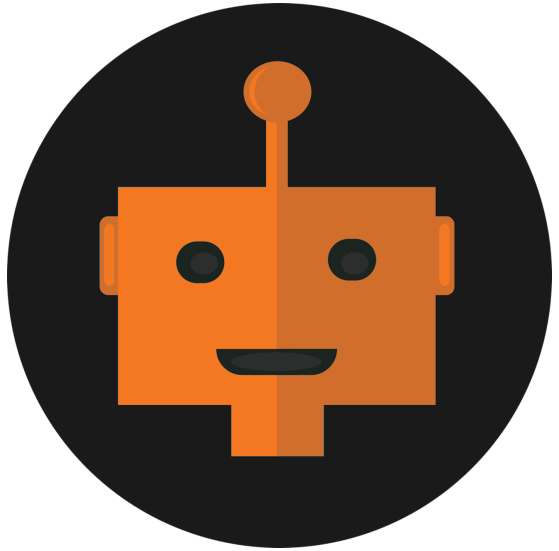
Week 1

NAU NORTHERN ARIZONA UNIVERSITY

Course instructor

- Dr. Ana Paula Chaves
  - Ph.D. in Informatics and Computing
  - Ms. in Computer Science
- Assistant Teaching Professor, NAU
- Contact:
  - Ana.Chaves@nau.edu
  - MS Teams: link on BBLearn
- Office hours:
  - Available on GitHub

About me…



Communication

MS Teams channels

Quick questions

Discussions

MS Teams private message

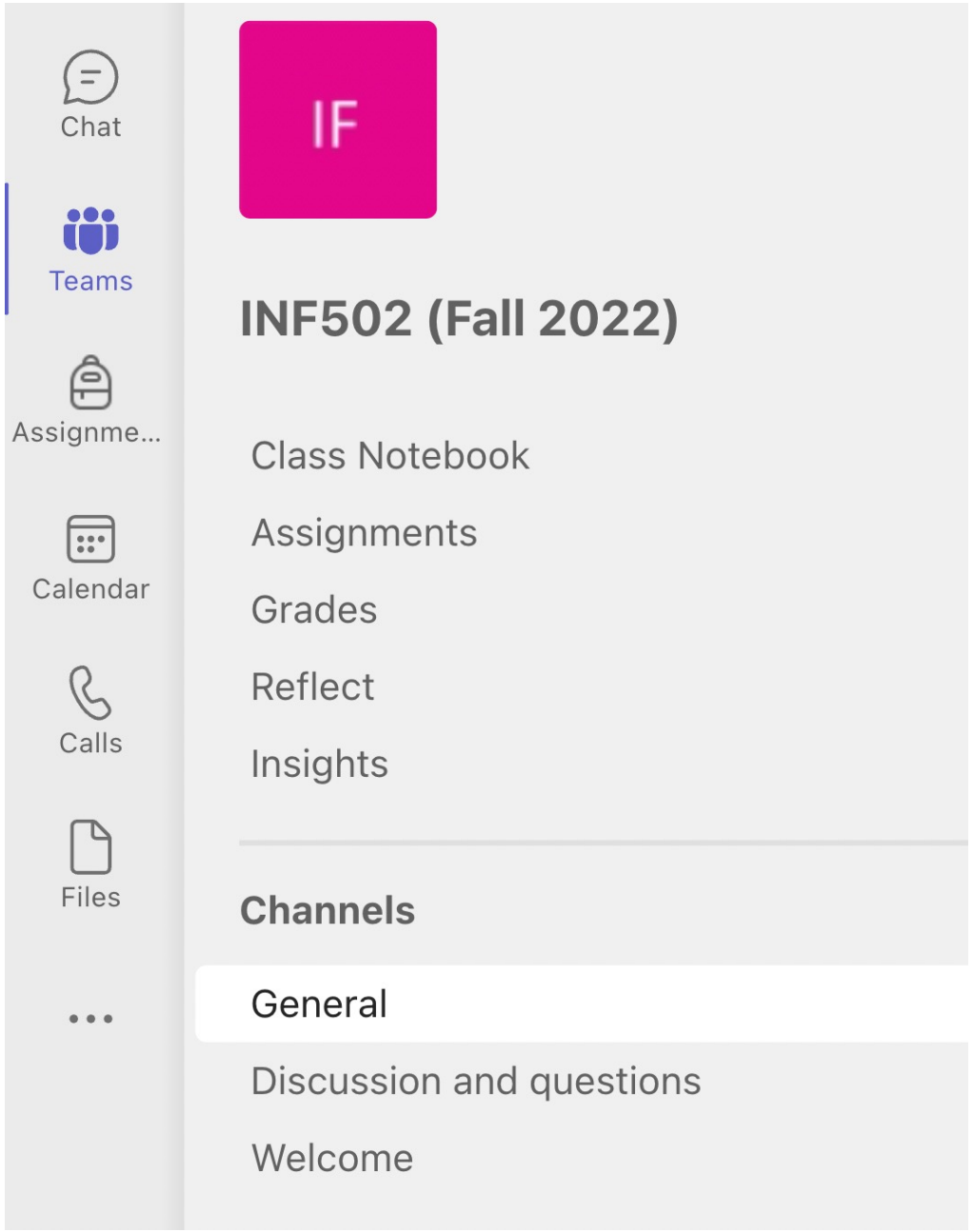
Individual interests

Email

Ana.Chaves@nau.edu

Office hours

In-person, SICCS building, rm 216



The course...

Course page

- https://github.com/chavesana/INF502-Fall22

What?

- Git/GitHub
- Python
  - With some extras
- Software engineering (Agile)

About you…

What is your background (BS, MS, etc.)

Knowledge in Programming (if any)

- where did you learn and how much do you know

What is your research topic (which program)

Your expectations about this course

Syllabus Time

NAU NORTHERN ARIZONA UNIVERSITY

INF 502 – SOFTWARE DEVELOPMENT METHODOLOGIES

Introduction to Programming Languages

NAU NORTHERN ARIZONA UNIVERSITY

Programming languages

Enable constructing representations of a computational process; well-defined algorithms processing information

Mapping to machine instructions

Syntax and associated semantics

Fundamentally just like human languages and form of expression

Non-functional properties become critical

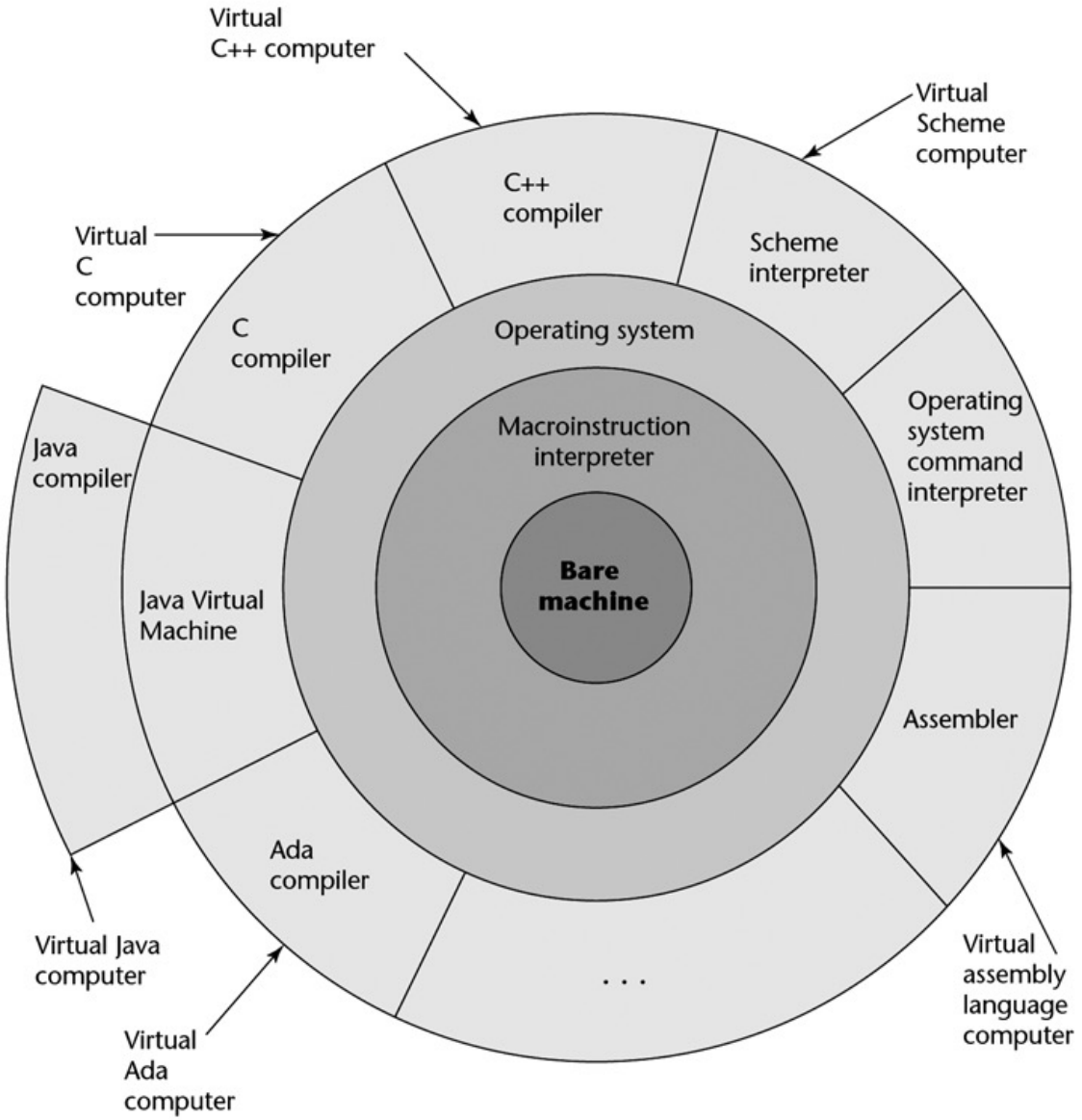
Language Implementations

Layered architectures

- Mappings from high-level to low-level instructions

Figure 1.2

Layered interface of virtual computers, provided by a typical computer system



Compiler-based implementations

Mapping

- High-level syntax to machine code
  - Plus linking of external resources

(Some) Advantages:

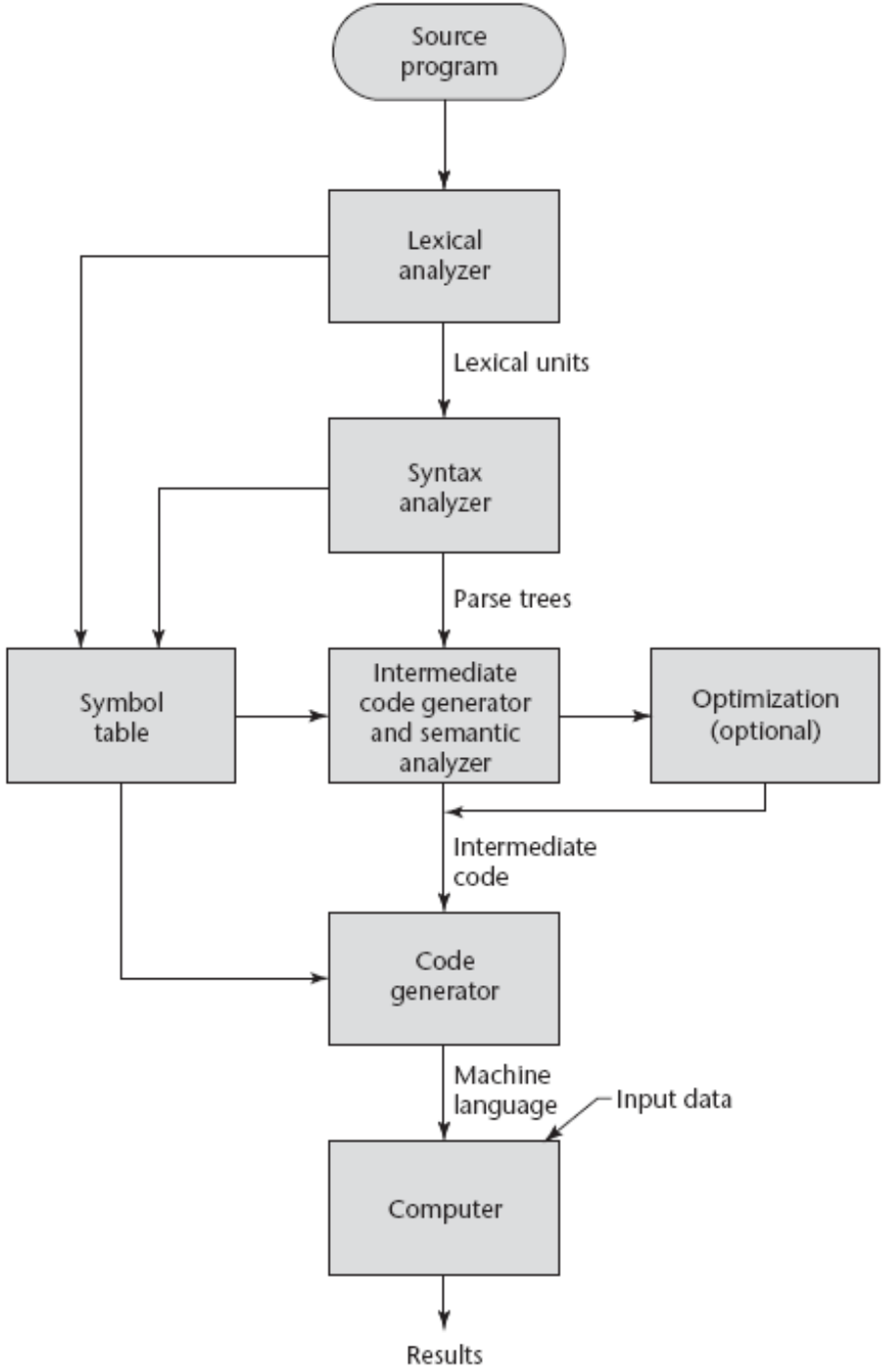
- (Usually) faster execution due to optimizations
  - Both algorithmic and machine-specific

(Some) Disadvantages:

- Compiled code coupled to specific hardware architecture
- Long iterative cycle
- Requires complete program

Figure 1.3

The compilation process

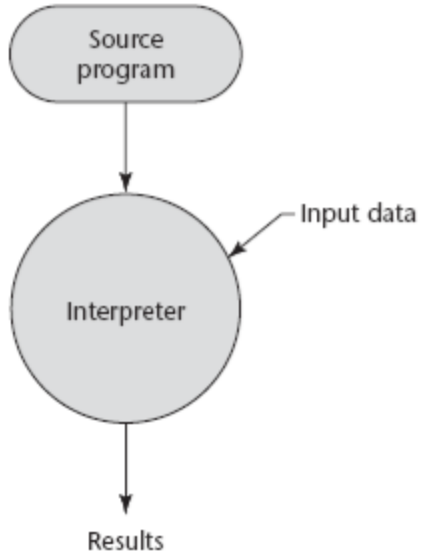


Interpreter-based implementations

- Mapping
  - High-level syntax executed by interpreter
  - Interpreter “wraps” around machine and maps to machine code
- (Some) Advantages:
  - Higher accessibility
    - Ease of experimentation
  - Portable from machine to machine
    - As long as an interpreter exists for each
  - Dynamic code generation
- (Some) Disadvantages:
  - (Usually) slower due to interpreter layer

Figure 1.4

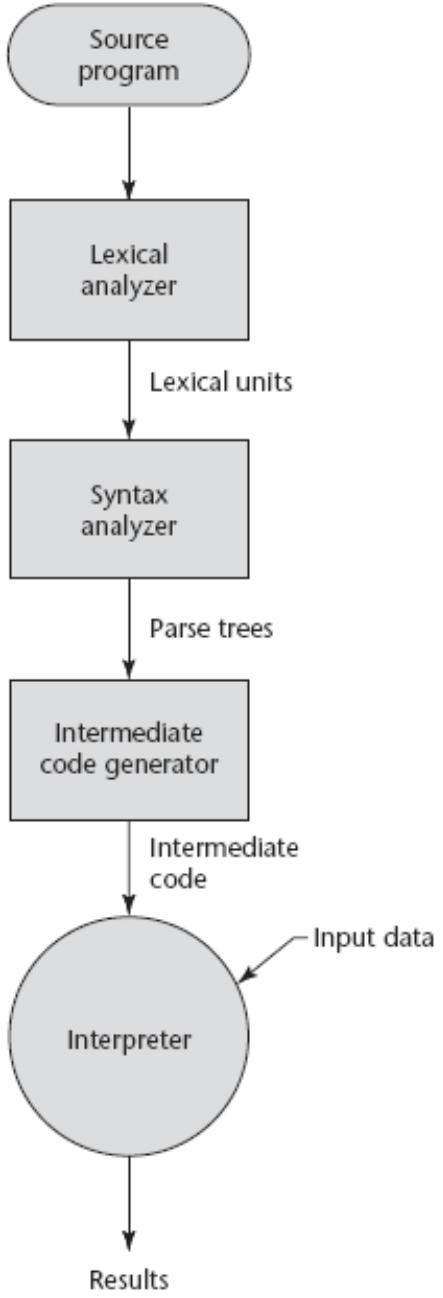
Pure Interpretation



Hybrid implementations

- Mapping
  - High-level syntax to interpreter instructions (intermediate representation)
    - Or purely interpreted
  - Interpreter still “wraps” around machine and maps intermediate representation to machine code
- (Some) Advantages:
  - Improved performance (over fully interpreted options)
    - Enabling compiler-type optimizations
  - Higher accessibility
    - Intermediate representation portable from machine to machine
- (Some) Disadvantages:
  - Longer iterative cycle than fully interpreted options
  - (Usually) still slower due to interpreter layer

Figure 1.5
Hybrid implementation system



Other Characteristics of Prog. Lang.

Binding

- Association between entity and attribute
  - Type bindings: static/dynamic, implicit/explicit
  - Storage bindings: static, stack, heap

Abstraction

- Support for defining new composite elements

Be ready for what's next…

Create a GitHub account: www.github.com

NAU NORTHERN ARIZONA UNIVERSITY