COMP 201:
PRINCIPLES OF PROGRAMMING

Learning Outcomes

- To understand what computing entails and what the different branches of computing are.
- To understand the basic design of a computer and how it represents information.
- To understand the different levels of programming languages.
- To know about the department's program structure and the various courses involved.

WHAT IS COMPUTING?

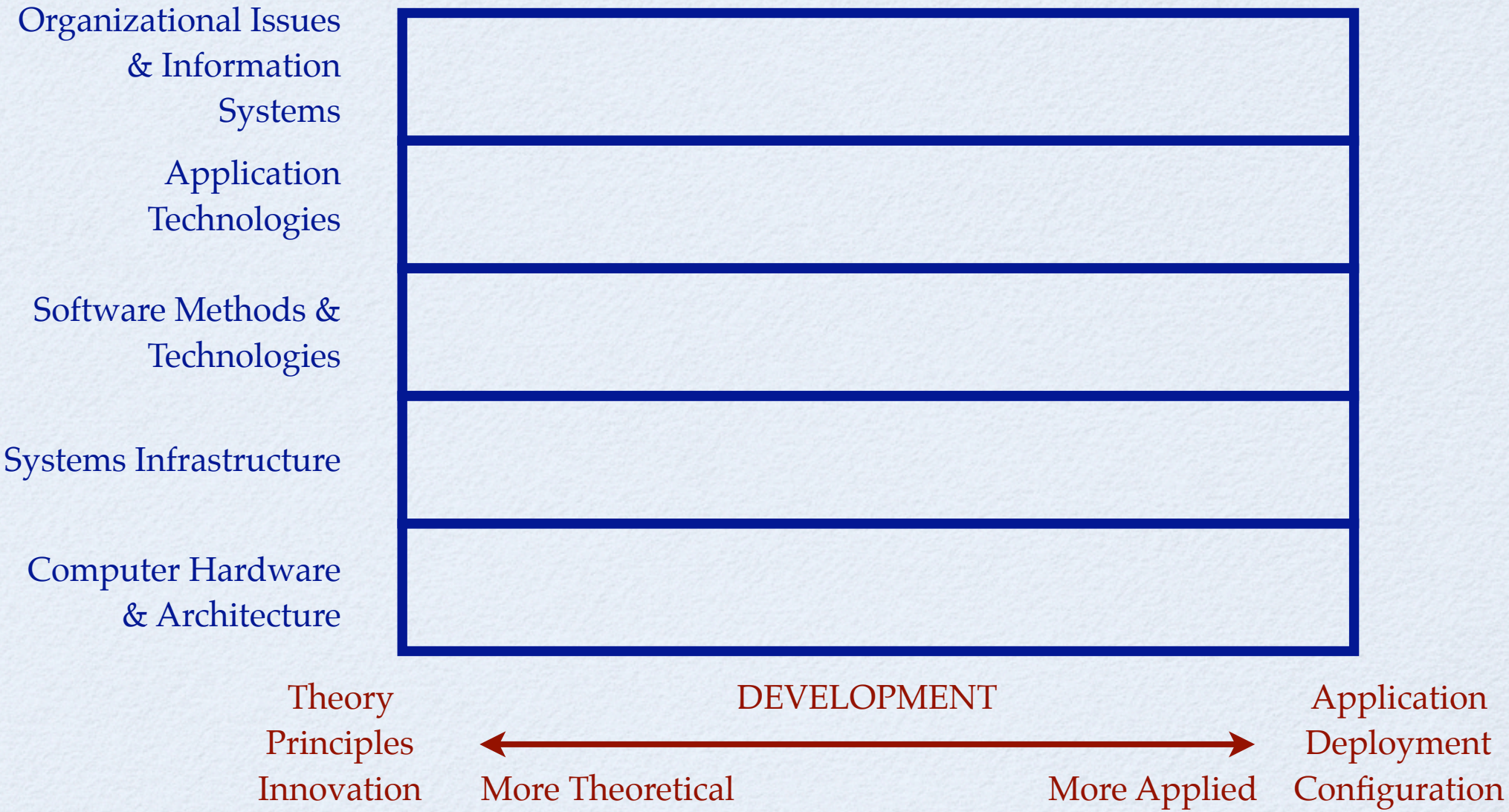


Figure from *Computing Curricula 2005, The Overview Report*

SOFTWARE ENGINEERING (COMPUTING)

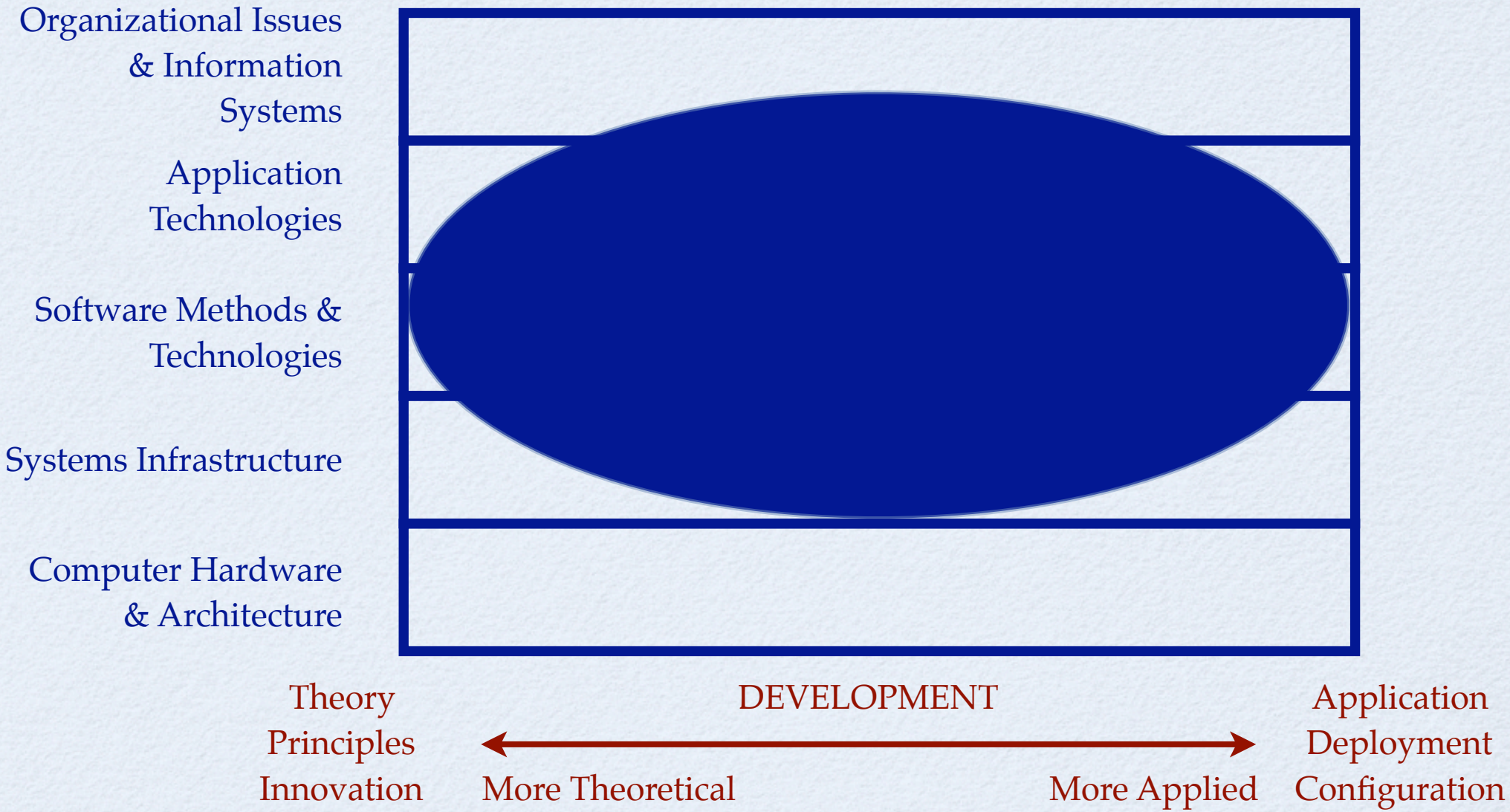


Figure from *Computing Curricula 2005, The Overview Report*

COMPUTER ENGINEERING (IT)

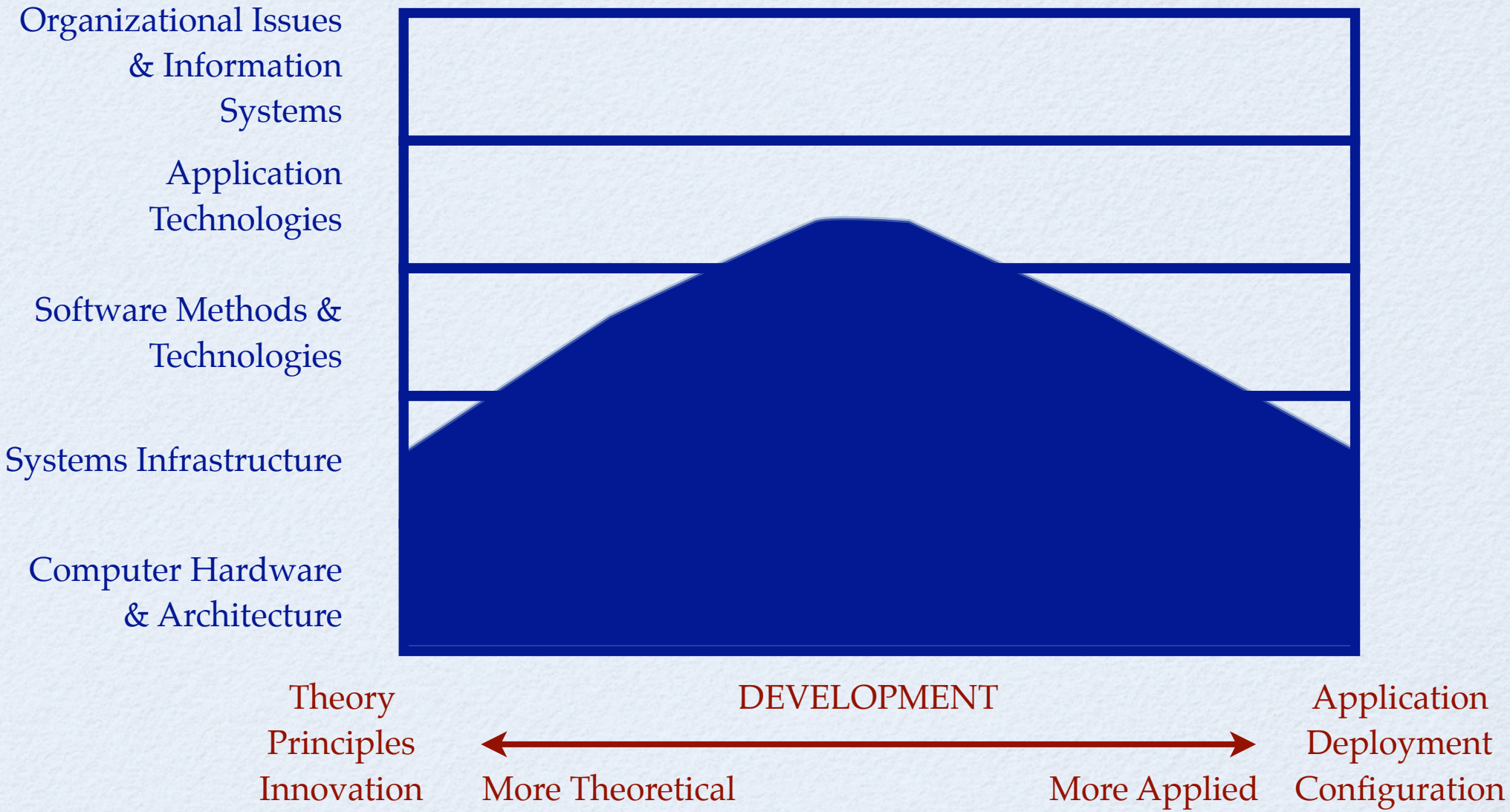


Figure from *Computing Curricula 2005, The Overview Report*

(ENTERPRISE) INFORMATION SYSTEMS

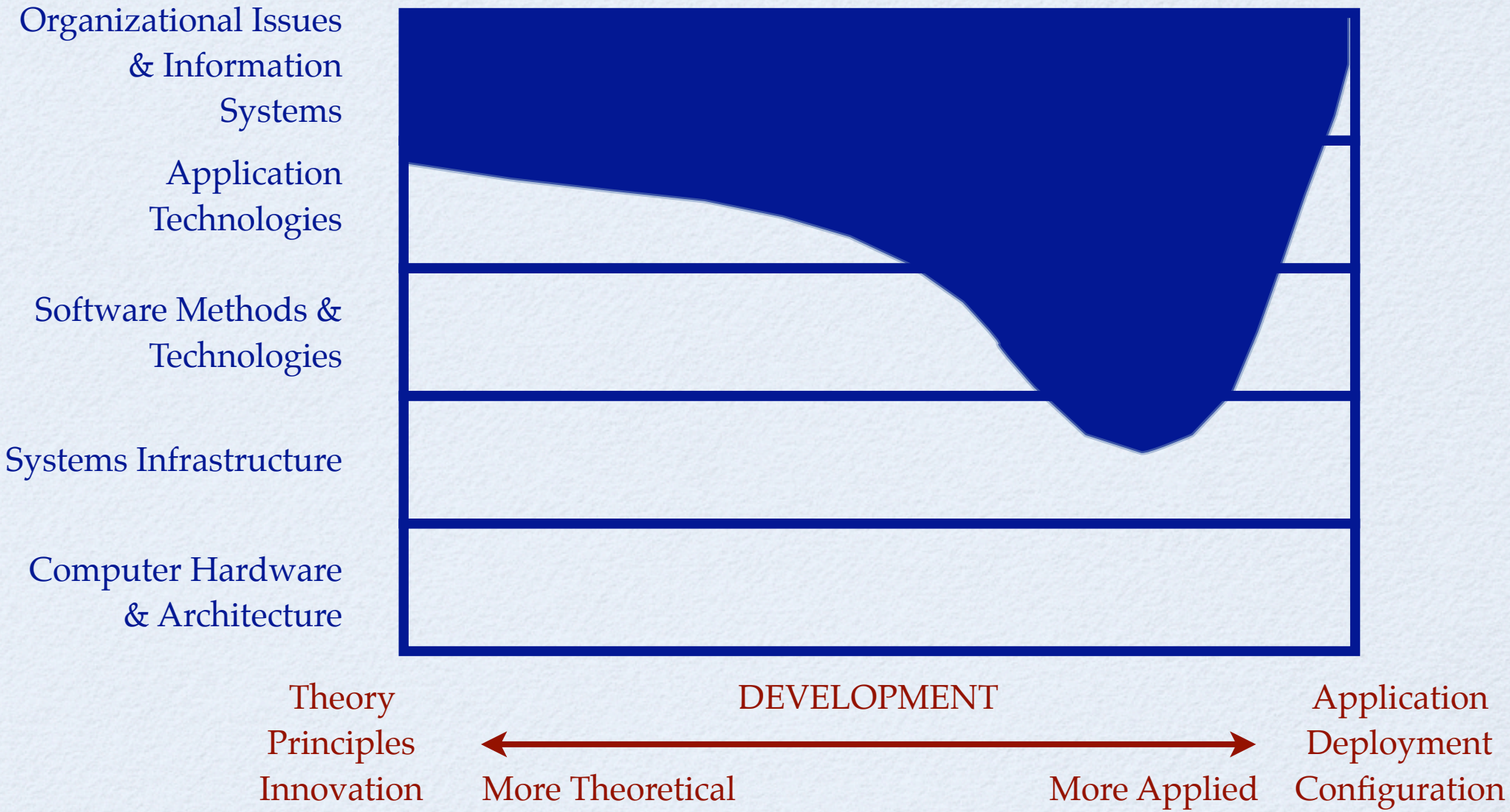


Figure from *Computing Curricula 2005, The Overview Report*

COMPUTER SCIENCE

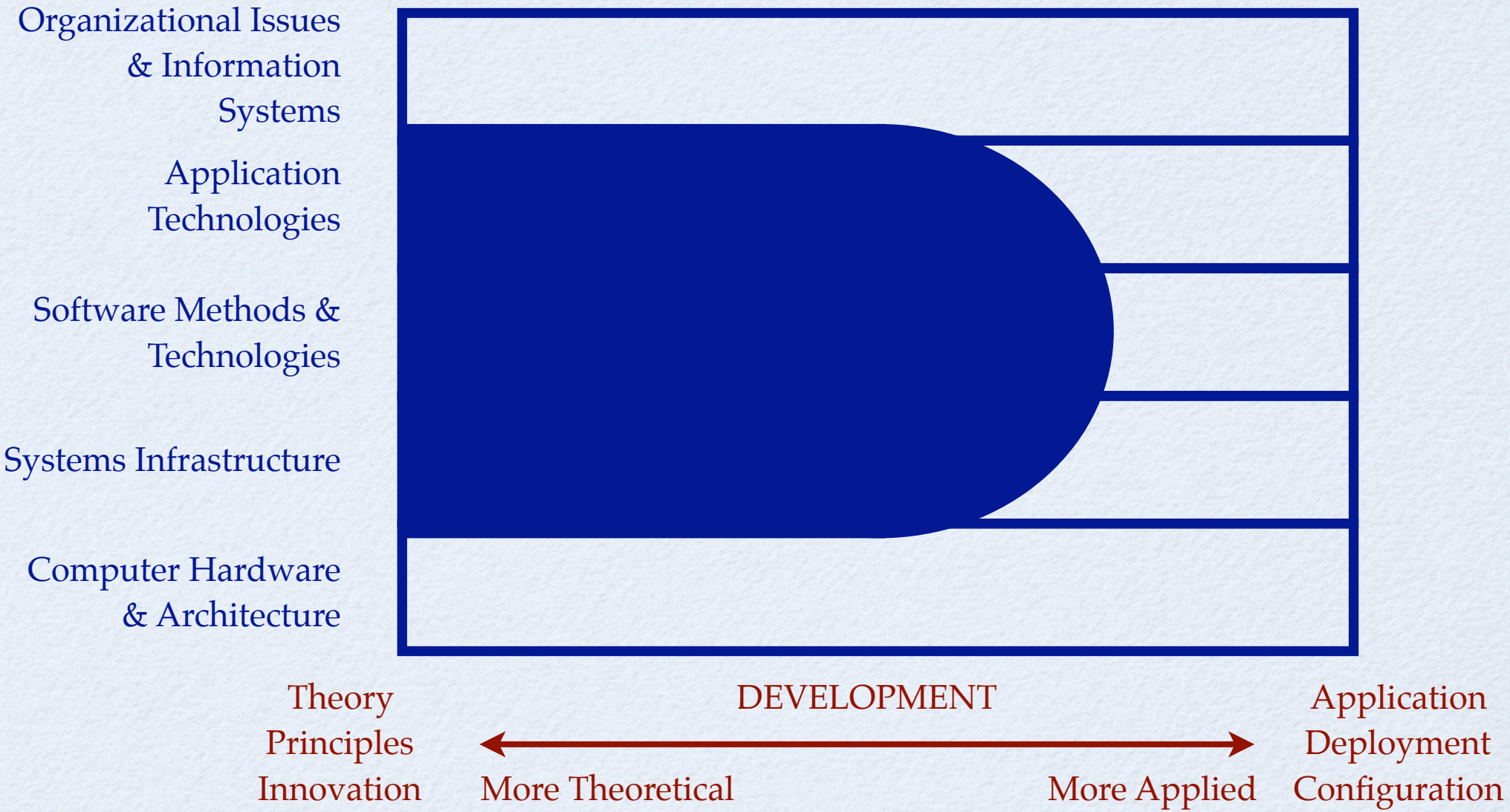


Figure from *Computing Curricula 2005, The Overview Report*

How about the computer?

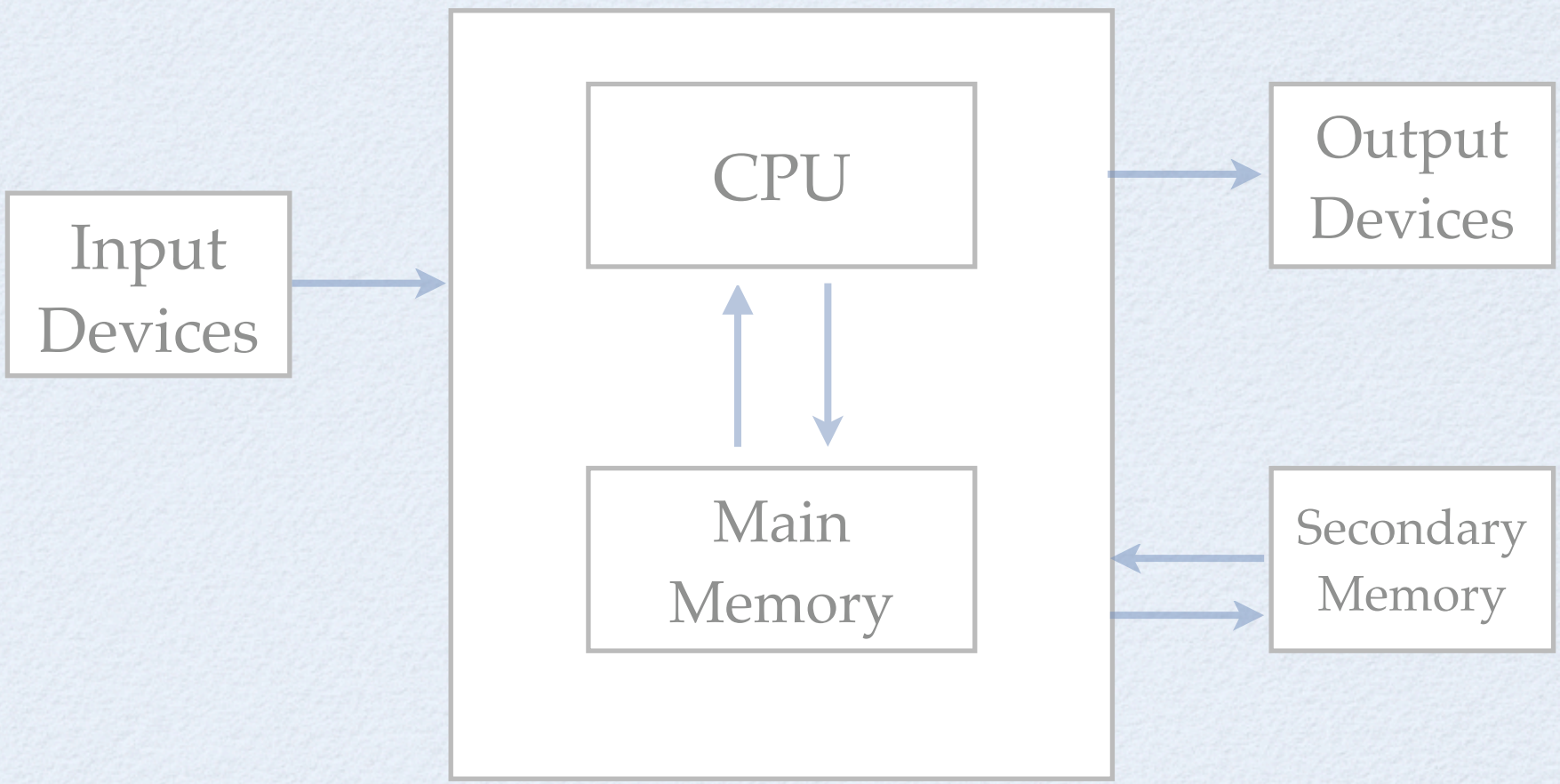
- "Computer science is no more about computers than astronomy is about telescopes."

  Edsger Dijkstra (1930-2002)

THE COMPUTER

- "Universal Information-Processing Machine"
- Understands Language.
- Can do anything that we can describe in enough detail.

FUNCTIONAL VIEW OF COMPUTERS



COMPUTER PROGRAM

- A complete, detailed set of steps for the computer to follow.
  - Algorithm: Step by step process to achieve the desired result.
- Example Task: Find how long it would take to get to Shek O from PolyU.
  - What would the algorithm be?

Natural Languages

- Computer Programs need to be written in a language for the computer to "read".
- Natural Languages: Chinese, English, French, Japanese, etc
- (Relatively) unstructured syntax
- Ambiguous semantics
  - "I sold my dog biscuits."
  - "I saw the man in the park with the telescope."

Programming Languages

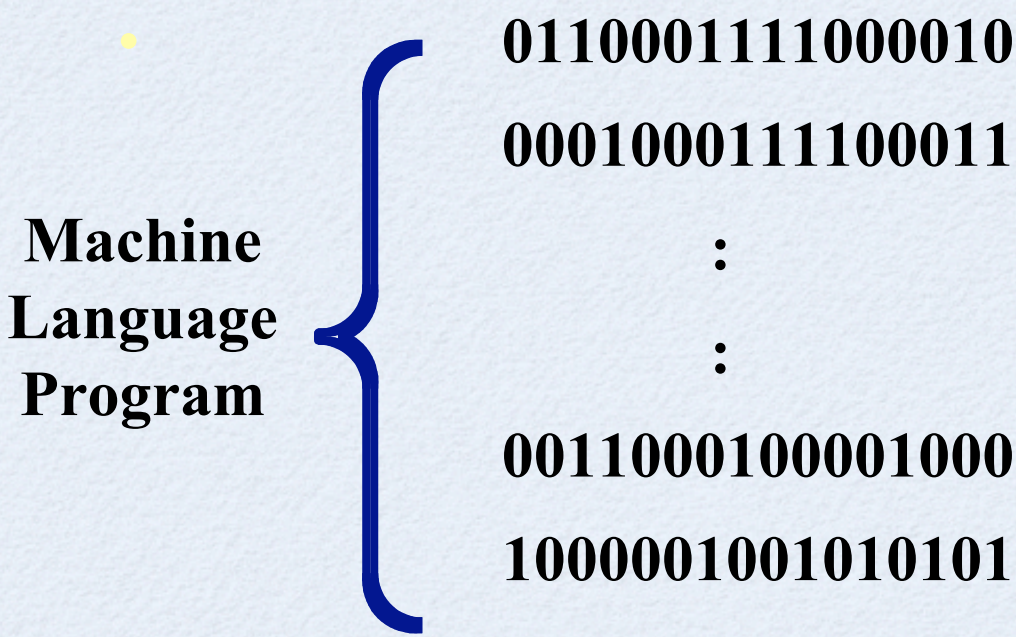
- Used for communicating with the computer.
- Many different kinds, and they all look different, but:
  - Precise Syntax
  - Precise Semantics
- Adding together two numbers:
  - Load the number from memory location 2001 into the CPU
  - Load the number from memory location 2002 into the CPU
  - Add the two numbers in the CPU
  - Store the result into memory location 2003

Computer Communication

- At the lowest level, computers are made of wires that conduct electricity.
  - The wire is either connected, or not connected.
  - Only two possible values — on/off, or high/low, or 1/0.
  - This is the **binary number system**.

Machine Language

- At the lowest level, computers "speak" in binary — 1 or 0.
- This is called **machine language**.



Assembly Language

- Created as a mnemonic to help humans to write programs.
- Maps machine instructions to "human-readable form"
- Assembler software translates assembly program to binary code.

LD R0, #05	Load the value 5 into the CPU
ADD R0, $1234	Add the contents of memory location 1234 to the value in the CPU
SUB R0, #22	Subtract the value 22 from the value in the CPU
LD $1234, R0	Store the value in the CPU into memory location 1234.

High-Level Programming Languages

- Designed to be used (and understood) by humans.
- Needs to be translated into machine language for computers to understand.

```
score = homework + test
if (score < 60):
   print “You Failed!”
else
   print “You Passed!”
```

HIERARCHY OF LANGUAGES

High-Level Languages	`score = homework + test`
	`if (score < 60):`
	`  print "You Failed!"`
	`else`
	`  print "You Passed!"`

Assembly Languages	`LD R0, #05`
	`ADD R0, $1234`
	`SUB R0, #22`
	`LD $1234, R0`

Machine Languages	`0110001111000010`
	`0001000111100011`
	`0011000100001000`
	`1000001001010101`

Hardware Level

High-level languages

- There are a number of high-level languages such as:
- Java, Perl, C, C++, C#, Ada, Fortran, BASIC, Python, LISP, Scheme, Ruby, etc.
- And they often look quite different!

High-Level Programming Languages

```
score = homework + test
if (score < 60):
   print “You Failed!”
else
   print “You Passed!”
```

```
(setq score (+ homework test))
(if (< score 60)
    (print “You Failed!”)
    (print “You Passed!”)
)
```

```
score = homework + test;
if (score < 60)
   System.out.println(“You Failed!”);
else
   System.out.println(“You Passed!”);
```

PROGRAMMING LANGUAGE HISTORY

Algol — Publication

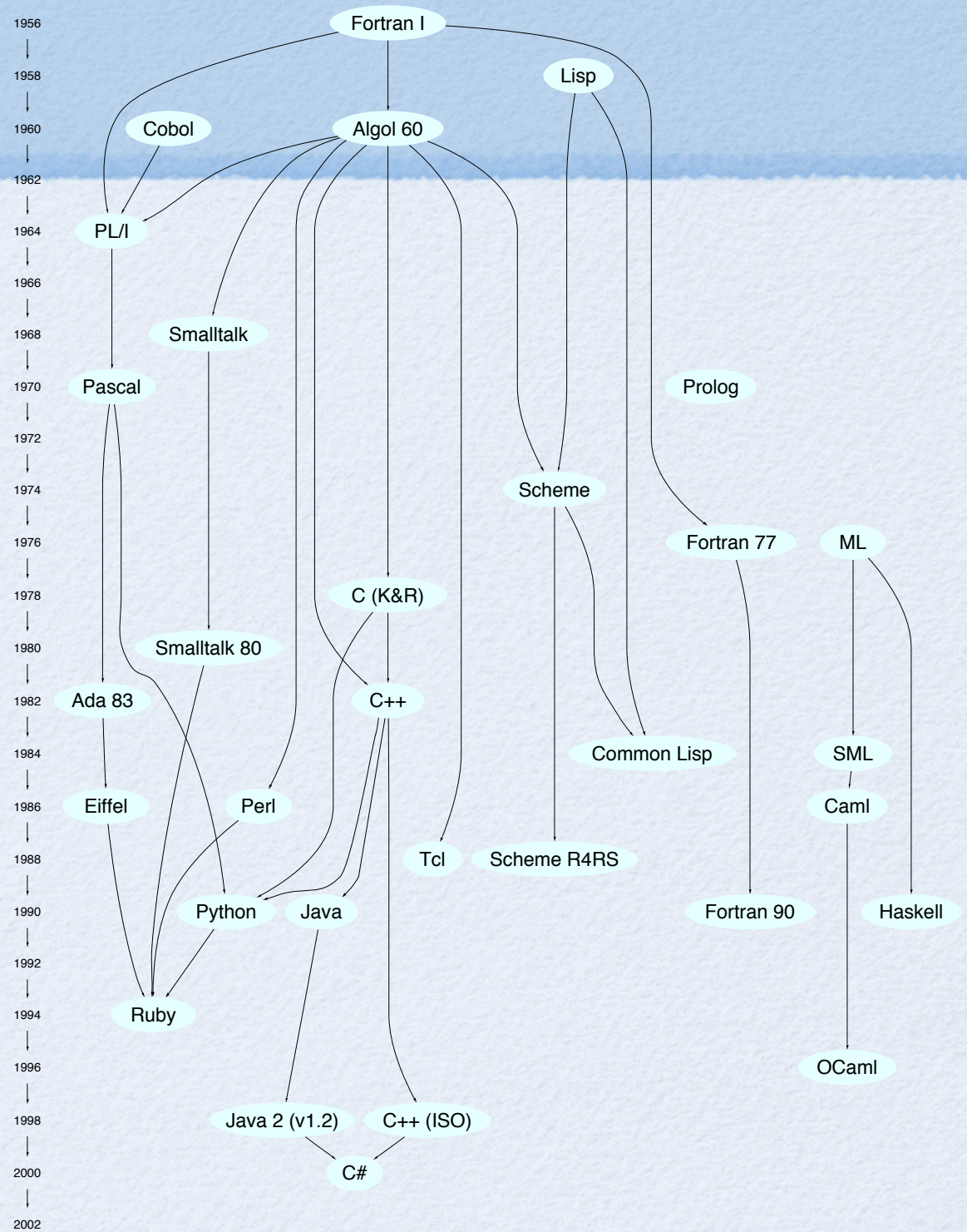
Fortran — Scientific

COBOL — Banking & Finance

Pascal — Teaching

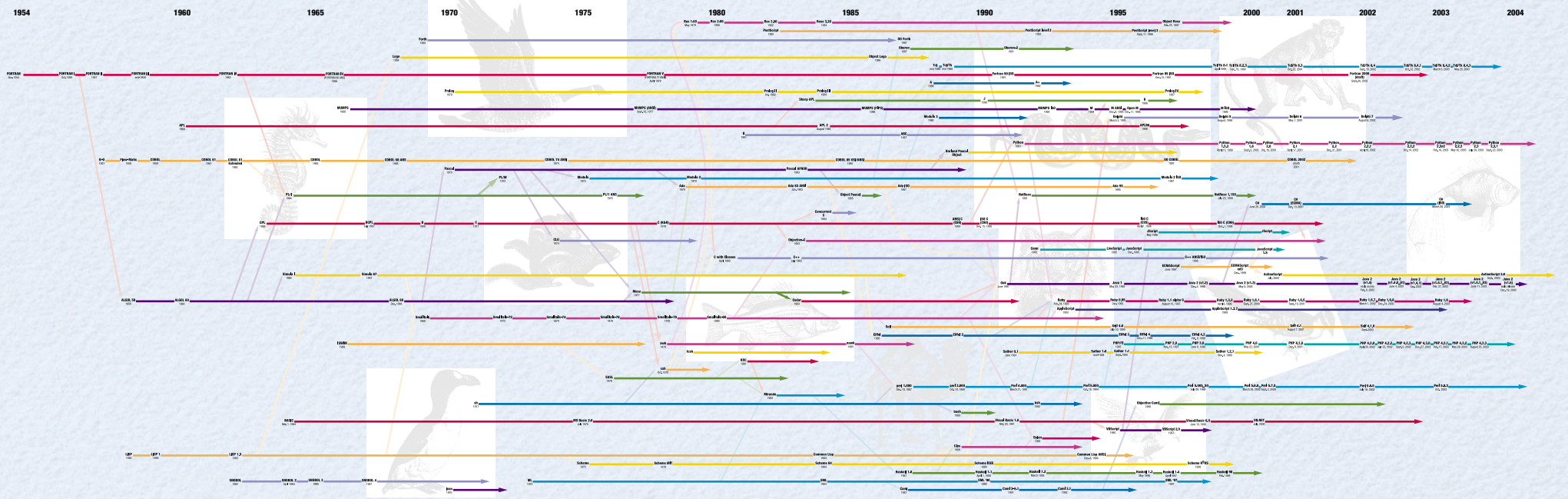
C/C++ — General Usage

Java?



History of Programming Languages

O'REILLY®



www.oreilly.com

For more than half of the fifty years computer programmers have been writing code, O'Reilly has provided developers with comprehensive, in-depth technical information. We've kept pace with rapidly changing technologies as new languages have emerged, developed, and matured. Whether you want to learn something new or need answers to tough technical questions, you'll find what you need in O'Reilly books and on the O'Reilly Network.

This timeline includes fifty of the more than 2500 documented programming languages. It is based on an original diagram created by Éric Lévénez (*www.levenez.com*), augmented with suggestions from O'Reilly authors, friends, and conference attendees.

For information and discussion on this poster, go to *www.oreilly.com/go/languageposter*.

Programming Language History

- The fact is, many of those languages (e.g. FORTRAN, BASIC, COBOL) were "all the rage" at some point in history.
- Java was hot a few years ago.
  - But debatable whether it's still as hot right now.
  - Seems like a lot of people want php, Ruby, etc.
- New languages have been invented, will be invented, and will eventually take over.
- Also, certain fields have their own specialized language(s).
  - Graphics: C; Banking, COBOL; etc

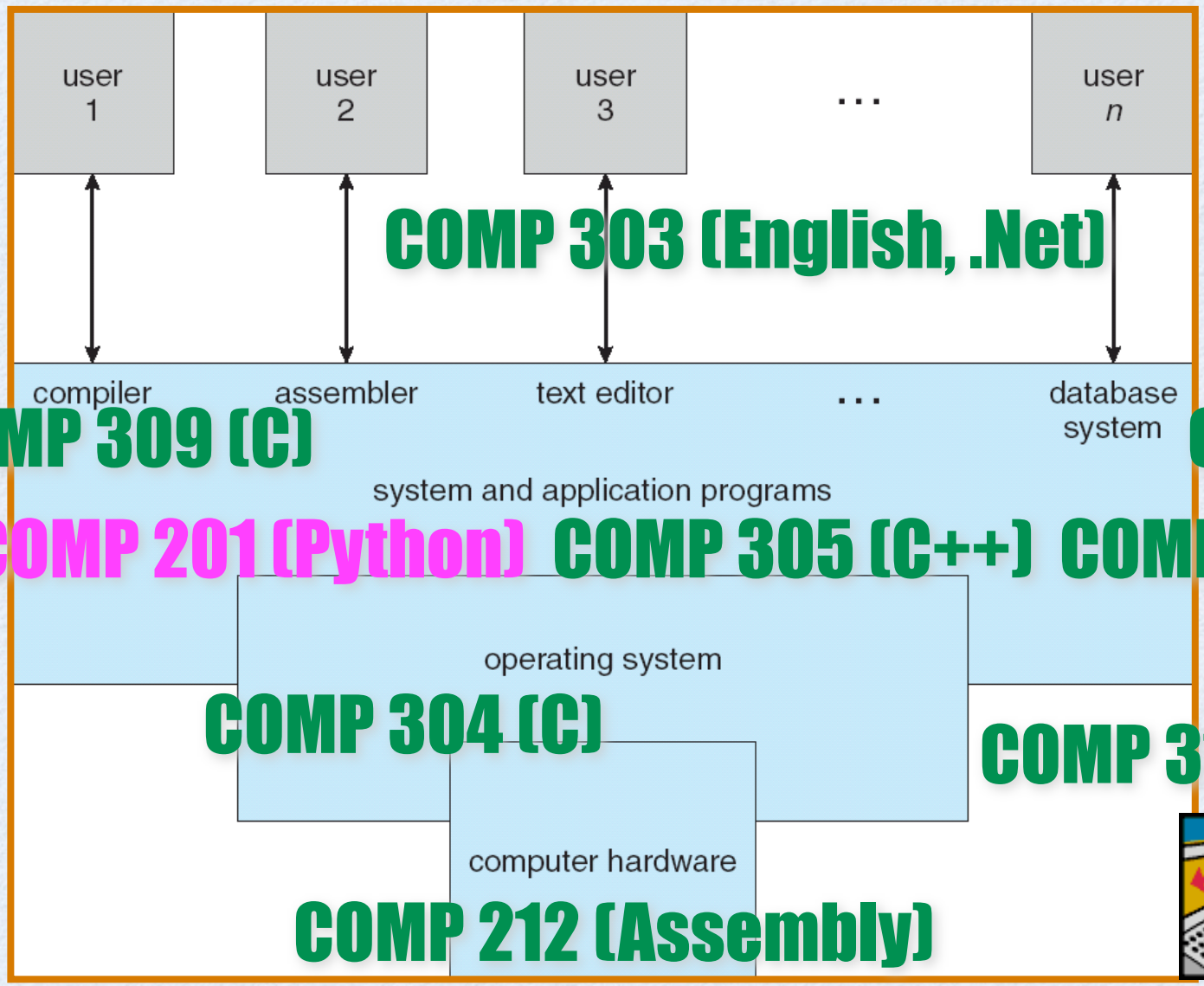
FOR COMP 201

- In COMP 201, we will be using Python, a high-level programming language.
- Why Python?
  - Small Language Set
  - Clean and minimal syntax
  - Hides a lot of the complexity of the computer
  - Can focus more on logic than on syntax issues
  - Gaining momentum in the commercial world

Learning to Program

- Obviously, it is not possible to formally learn (i.e. in a classroom) all possible languages.
  - COMP 201 is the *only* programming-specific course you'll encounter here at PolyU.
- You will have to teach yourself a new language at some point.
- Therefore, it is more important to master *programming*, rather than a particular language.
- The principles (logic, sequential thinking, problem solving) remain the same.

Programming languages you'll meet later



COMP 303 (English, .Net)

COMP 309 (C)

COMP 311 (SQL)

COMP 201 (Python)

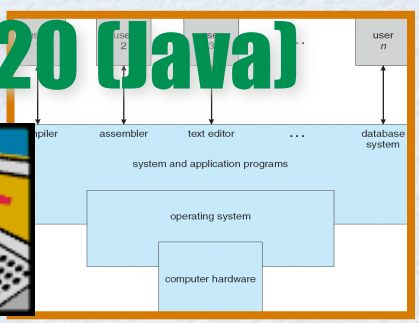
COMP 305 (C++)

COMP 302 (English)

COMP 304 (C)

COMP 312, 320 (Java)

COMP 212 (Assembly)



Mastering Programming

- How do I master programming?
- Practice, practice, practice.
  - If you don't understand something, try it!
  - Always have a little "Test program" sitting around, where you can test out little bits of code.
- Read, read, read
  - There are lots of great online (and free) references out there for programming.
  - If you aren't sure about something, google it!

What we've learned

- We know what computing is and what's the difference between Computing, IT and EIS.
- We have a high-level overview of the different parts of a computer and we know how a computer stores its information.
- We know why there are different levels of programming languages and what is different between them.
- We have an idea how the different courses in the department relate and complement each other.