Software Engineering

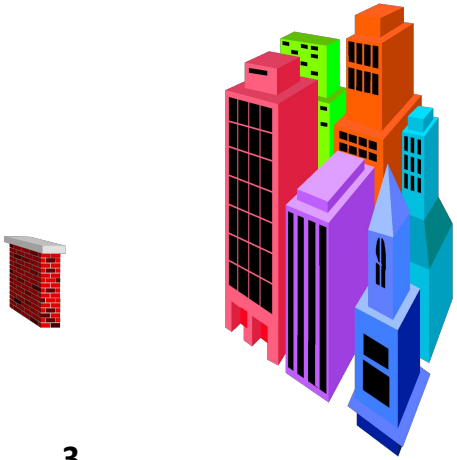
Lec 2

Contents

- Software Engineering

What is Software Engineering?

- Engineering approach to develop software.
  - Building Construction Analogy.
- Systematic collection of past experience:
  - Techniques,
  - Methodologies,
  - Guidelines.



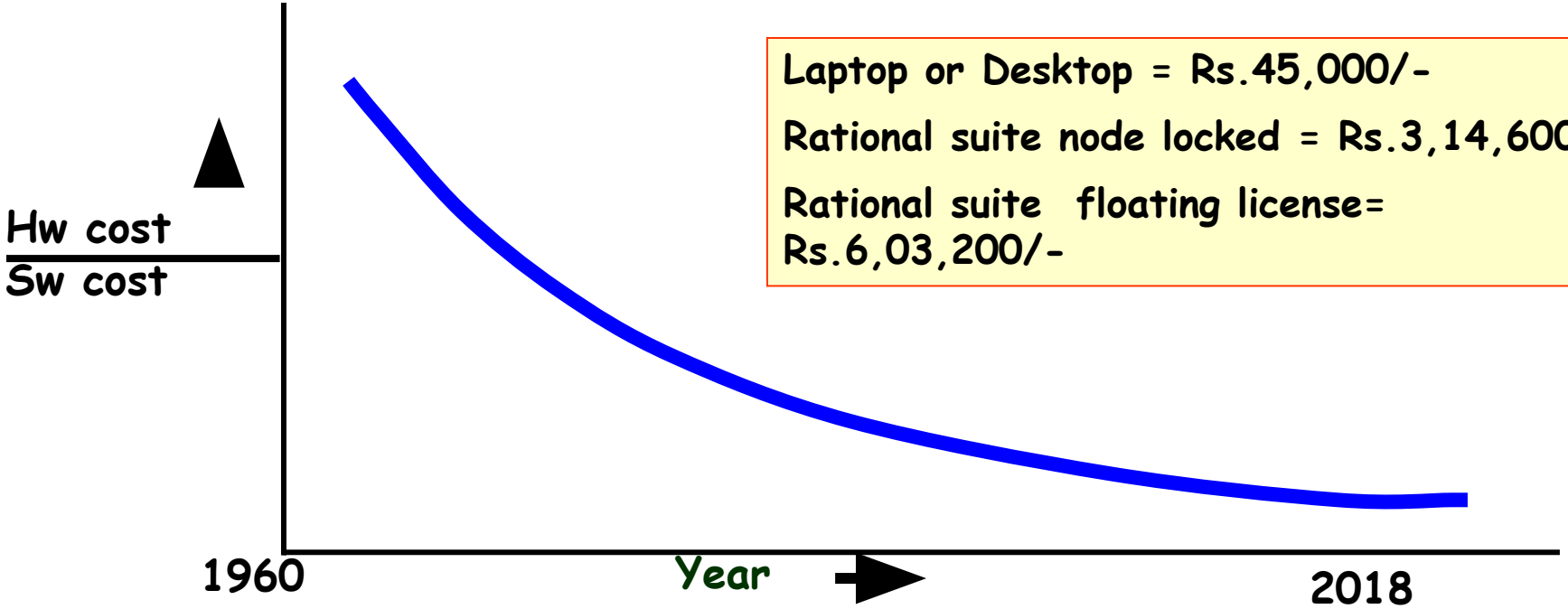
IEEE Definition

- “Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.”

Software Crisis

- It is often the case that software products:
  - Fail to meet user requirements.
  - Expensive.
  - Difficult to alter, debug, and enhance.
  - Often delivered late.
  - Use resources non-optimally.

Software Crisis (cont.)



Laptop or Desktop = Rs.45,000/-

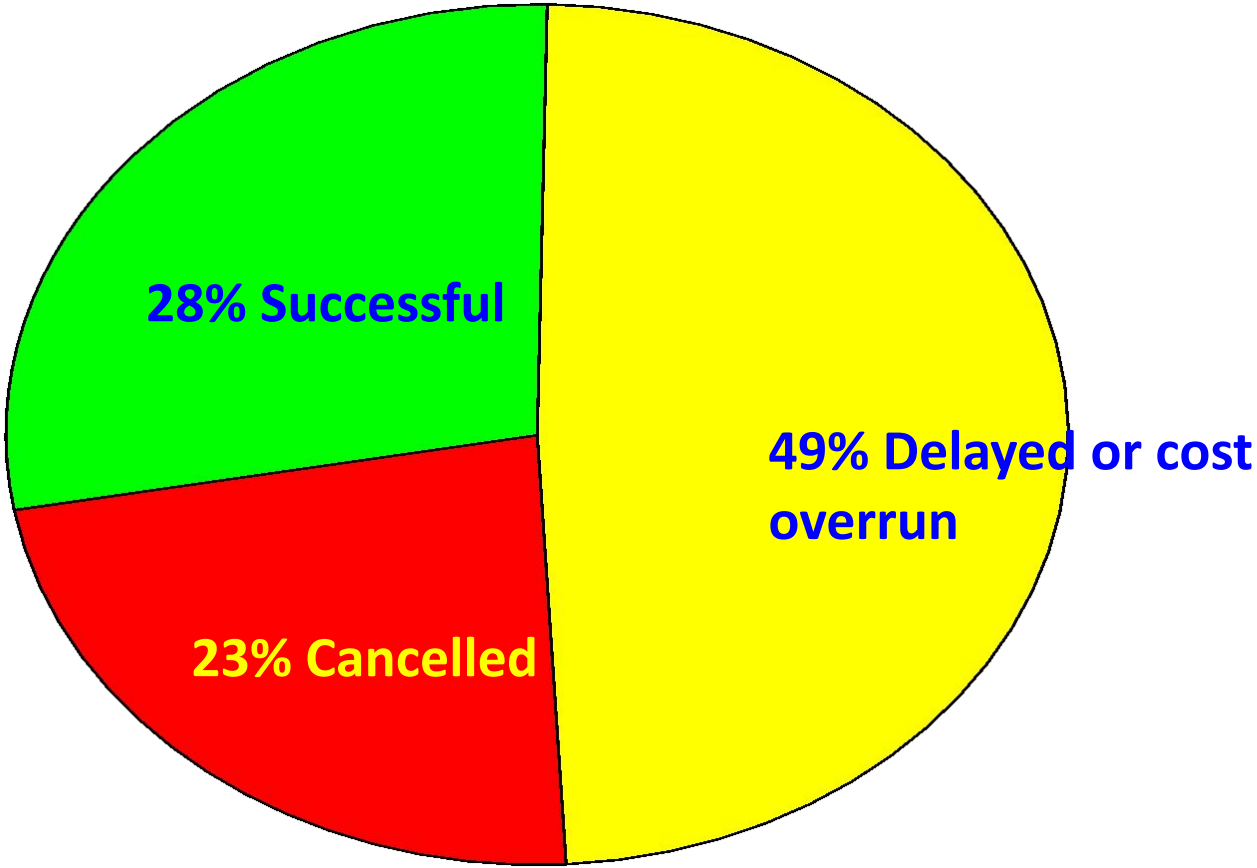
Rational suite node locked = Rs.3,14,600/-

Rational suite floating license= Rs.6,03,200/-

Relative Cost of Hardware and Software

Then why not have entirely hardware systems?...

- A virtue of software:
  - Relatively easy and faster to develop and to change…
  - Consumes no space, weight, or power…
  - Otherwise all might as well be hardware.
- The more is the complexity of software, the harder it is to change--why?
  - Further, the more the changes made to a program, the greater becomes its complexity.

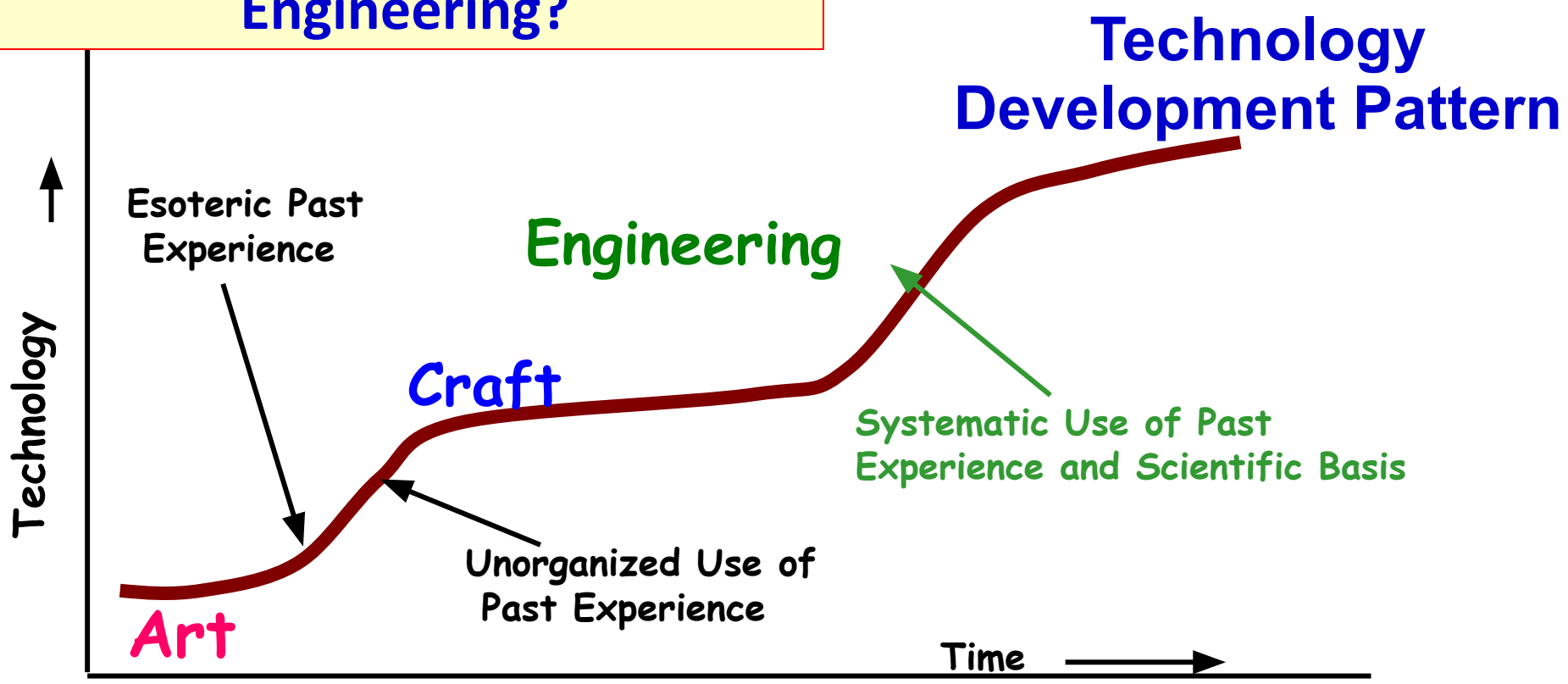


Standish Group Report

Which Factors are Contributing to the Software Crisis?

- Larger problems,
- Poor project management
- **Lack of adequate training in software engineering,**
- Increasing skill shortage,
- Low productivity improvements.

Programming: an Art or Engineering?



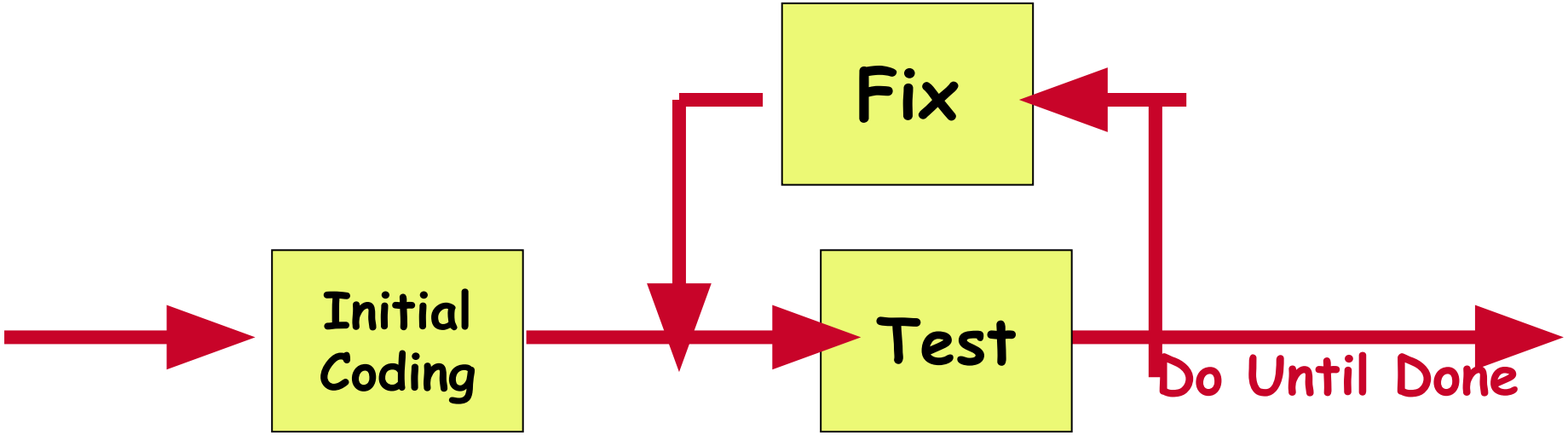
- Heavy use of past experience:
  - Past experience is systematically arranged.
- Theoretical basis and quantitative techniques provided.
- Many are just thumb rules.
- Tradeoff between alternatives.
- Pragmatic approach to cost-effectiveness.

Programming an Art or Engineering?

What is Exploratory Software Development?

- Early programmers used **exploratory** (also called build and fix) style.
  - A `dirty' program is quickly developed.
  - The bugs are fixed as and when they are noticed.
  - Similar to how a junior student develops programs…

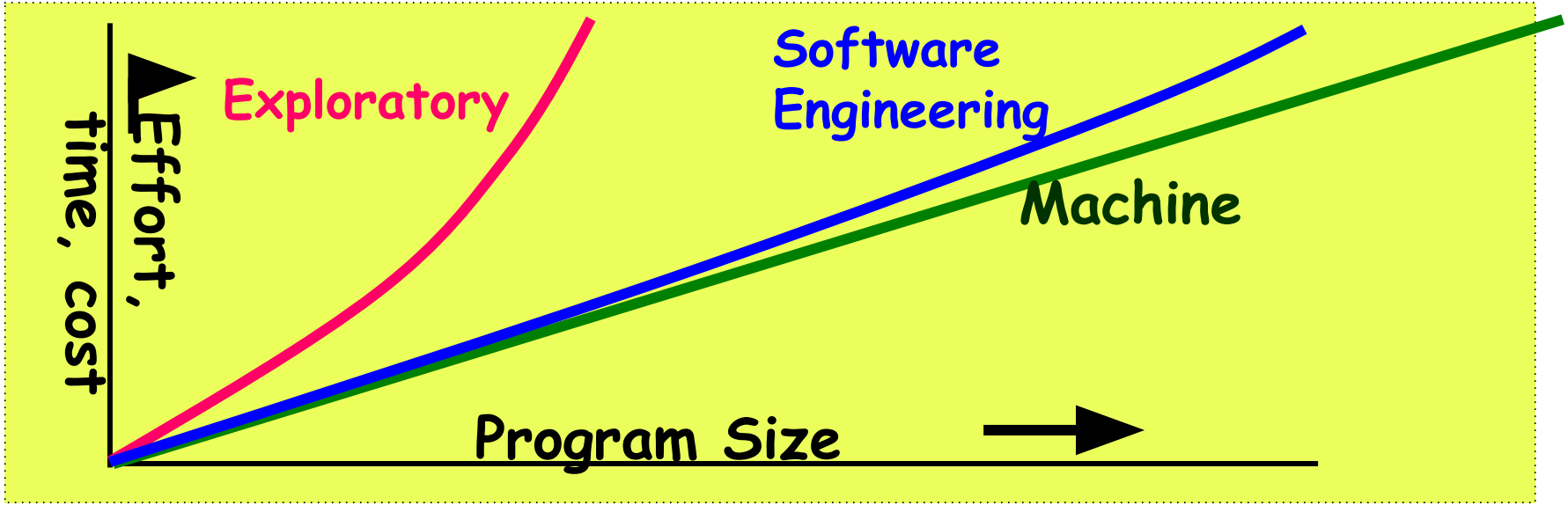
Exploratory Style



Does not work for nontrivial projects… Why?…

What is Wrong with the Exploratory Style?

- Can successfully be used for developing only very small (toy) programs.



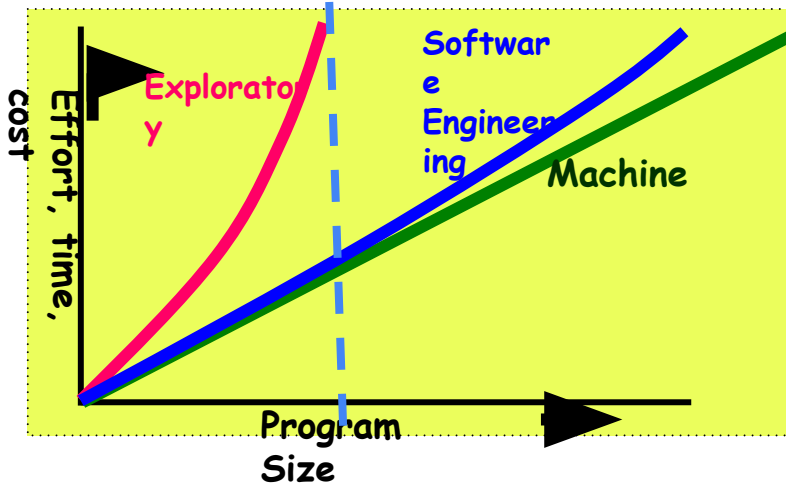
What is Wrong with the Exploratory Style? Cont…

- Besides the exponential growth of effort, cost, and time with problem size:
  - Exploratory style usually results in unmaintainable code.
  - **It becomes very difficult to use the exploratory style in team development environments…**

What is Wrong with the Exploratory Style? Cont…

- **Why does the effort required to develop a software grow exponentially with size?**

- Why does the approach completely breaks down when the size of software becomes large?



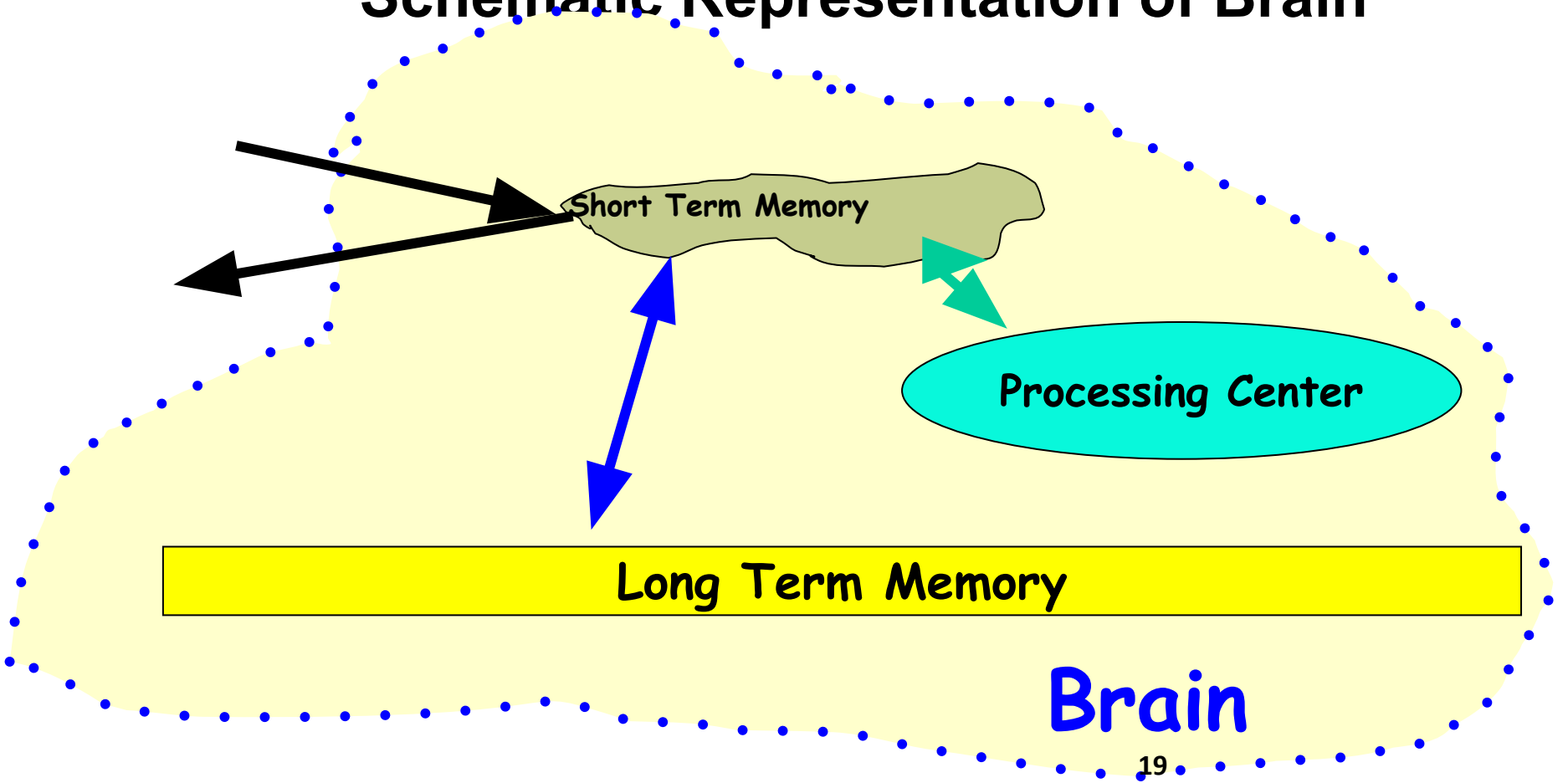
An Interpretation Based on Human Cognition Mechanism

- Human memory can be thought to be made up of two distinct parts [Miller 56]:
  - **Short term memory and**
  - **Long term memory.**

Human Cognition Mechanism

- Suppose I ask: **“It is 10:10AM now, how many hours are remaining today?"**
  - 10AM would be stored in the short-term memory.
  - “A day is 24 hours long.” would be fetched from the long term memory into short term memory.
  - The mental manipulation unit would compute the difference (24-10).

Schematic Representation of Brain



- **An item is any set of related information.**
  - A character such as `a' or a digit such as `5'.
  - A word, a sentence, a story, or even a picture.
- Each item normally occupies one place in memory.

What is an Item?

- When you are able to relate several different items together (chunking):
  - The information that should normally occupy several places, takes only one place in memory.

Chunking

- If I ask you to remember the number **110010101001**
  - It may prove very hard for you to understand and remember.
  - But, the octal form of **6251** **(110)(010)(101)(001)** would be easier.
  - You have managed to create chunks of three items each.

Evidence of Short Term Memory

- In many of our day-to-day experiences:
  - **Short term memory is evident.**
- Suppose, you look up a number from the telephone directory and start dialling it.
  - If you find the number is busy, you can dial the number again after a few seconds without having to look up the number from directory.
- But, after several days:
  - You may not remember the number at all
  - Would need to consult the directory again.

- If a person deals with seven or less number of items:
  - These would be accommodated in the short term memory.
  - So, he can easily understand it.
- As the number of new information increases beyond seven:
  - It becomes exceedingly difficult to understand it.

The Magical Number 7

What is the Implication in Program Development?

- A small program having just a few variables:
  - Is within easy grasp of an individual.
- As the number of independent variables in the program increases:
  - **It quickly exceeds the grasping power of an individual…**
  - **Requires an unduly large effort to master the problem.**

Implication in Program Development

- Instead of a human, if a machine could be writing (generating) a program,
  - The slope of the curve would be linear.
- But, how does use of software engineering principles helps hold down the effort-size curve to be almost linear?
  - **Software engineering principles extensively use techniques specifically targeted to overcome the human cognitive limitations.**

Which Principles are Deployed by Software Engineering Techniques to Overcome Human Cognitive Limitations?

- Two important principles are profusely used:
  - **Abstraction**
  - **Decomposition**

Two Fundamental Techniques to Handle Complexity

What is Abstraction?

- Simplify a problem by omitting unnecessary details.
  - **Focus attention on only one aspect of the problem and ignore other aspects and irrelevant details.**
  - Also called model building.

Abstraction Example

- Suppose you are asked to develop an overall understanding of some country.
  - Would you:
    - Meet all the citizens of the country, visit every house, and examine every tree of the country?
  - You would possibly refer to various types of maps for that country only.

You would study an Abstraction…

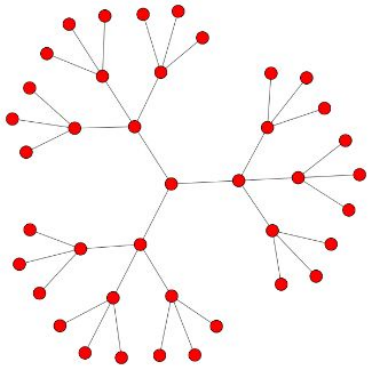
- A map is:
  - An abstract representation of a country.
  - Various types of maps (abstractions) possible.

Does every Problem have a single Abstraction?

- Several abstractions of the same problem can be created:
  - Focus on some specific aspect and ignore the rest.
  - Different types of models help understand different aspects of the problem.

Abstractions of Complex Problems

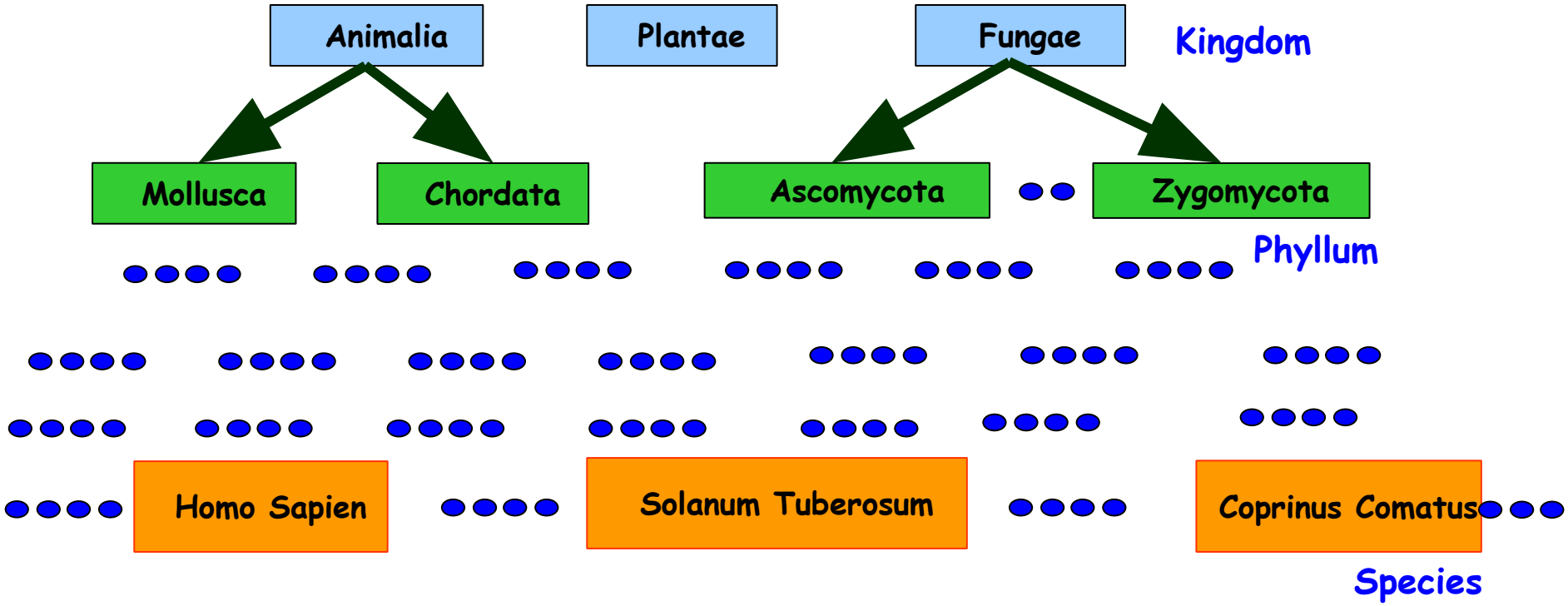
- For complex problems:
  - A single level of abstraction is inadequate.
  - A hierarchy of abstractions may have to be constructed.
- Hierarchy of models:
  - A model in one layer is an abstraction of the lower layer model.
  - An implementation of the model at the higher layer.



Abstraction of Complex Problems -- An Example

- Suppose you are asked to understand all life forms that inhabit the earth.
- Would you start examining each living organism?
  - You will almost never complete it.
  - Also, get thoroughly confused.
- **Solution: Try to build an abstraction hierarchy.**

Living Organisms



Quiz

- What is a model?
- Why develop a model? That is, how does constructing a model help?
- Give some examples of models.

Decomposition

- Decompose a problem into many small independent parts.
  - The small parts are then taken up one by one and solved separately.
  - **The idea is that each small part would be easy to grasp and therefore can be easily solved.**
  - **The full problem is solved when all the parts a solved.**

Decomposition

- A popular example of decomposition principle:
  - Try to break a bunch of sticks tied together versus breaking them individually.

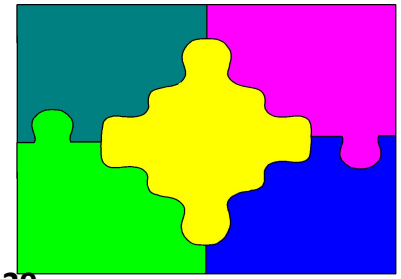
- Any arbitrary decomposition of a problem may not help.
  - The decomposed parts must be more or less independent of each other.

Decomposition: Another Example

- Example use of decomposition principle:
  - You understand a book better when the contents are organized into independent chapters.
  - Compared to when everything is mixed up.

Why Study Software Engineering? (1)

- To acquire skills to develop large programs.
  - **Handling exponential growth in complexity with size.**
  - Systematic techniques based on abstraction (modelling) and decomposition.



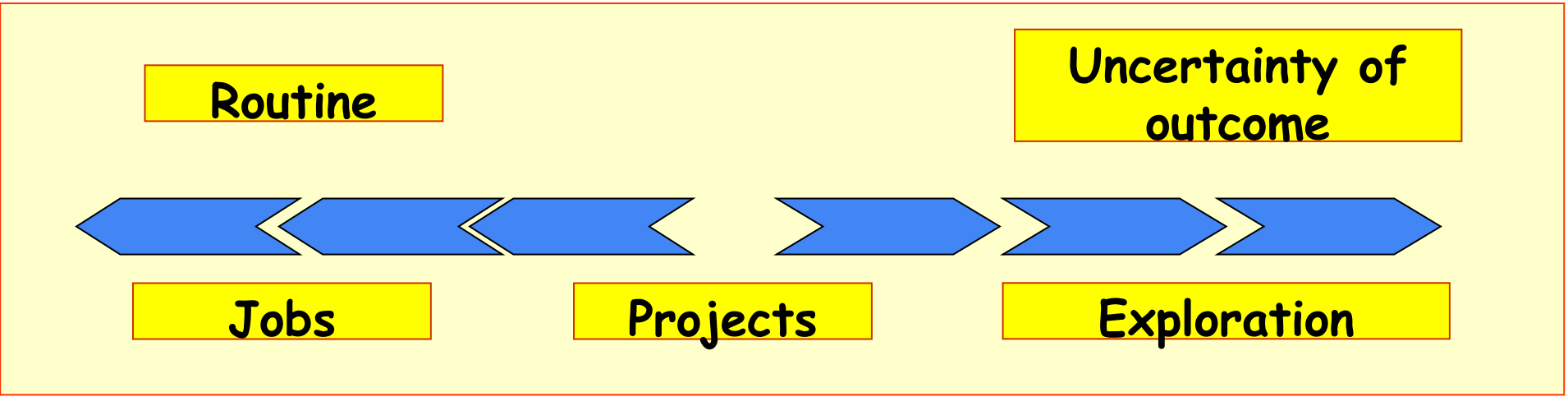
Why Study Software Engineering? (2)

- Learn systematic techniques of:
  - **Specification, design, user interface development, testing, project management, maintenance, etc.**
  - Appreciate issues that arise in team development.

Why Study Software Engineering? (3)

- To acquire skills to be a better programmer
  - Higher Productivity
  - Better Quality Programs

Jobs versus Projects



Jobs – repetition of very well-defined and well understood tasks with very little uncertainty

Exploration – The outcome is very uncertain, e.g. finding a cure for cancer.

Projects – in the middle! Has challenge as well as routine…

Types of Software Projects

- Two types of software projects:
  - **Products (Generic software)**
  - **Services (custom software)**
- Total business – Several Trillions of US $
  - Half in products and half services
  - **Services segment is growing fast!**

Types of Software

Packaged software —prewritten software available for purchase

- **Horizontal market software**—meets needs of many companies
- **Vertical market software**—designed for particular industry

Custom software — software developed at some user's requests-Usually developer tailors some generic solution

Thank You
Get Vaccinated and Wear Mask