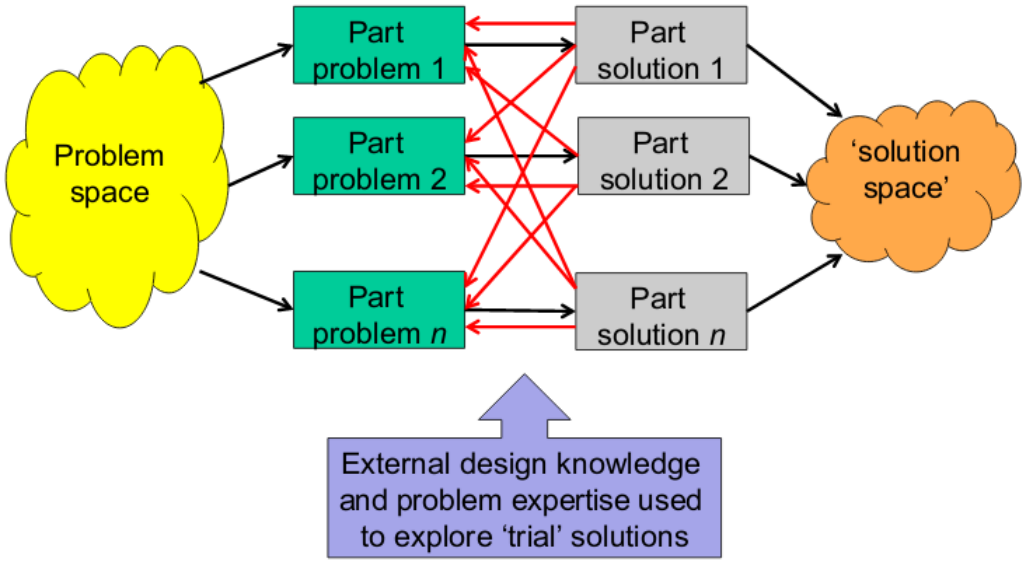
What is Software Engineering?

1 ISP - Ill-structured problems

- Many possible solutions, good/bad rather than right/wrong
- No definitive test of any outcome
- No 'stopping rule' that can be used to determine that a solution (or an optimum one) has been reached
- No definitive formulation. Our understanding of the nature of the problem is entwined with our ideas about solving it

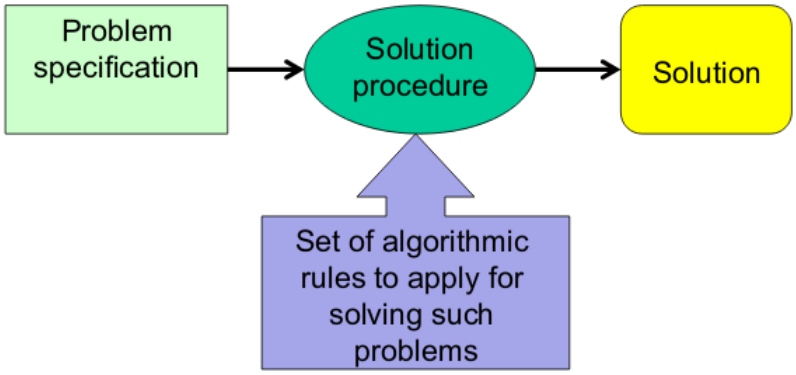


1.1 Addressing an ISP

Learning how to address ISPs requires you to learn effective ways to explore ideas (rather than learning the 'right' way to do something)

2 WSP - Well-structured problems

- The existence of a 'right' solution/answer
- Having a specific criterion for testing any proposed solution
- Having at least one problem space in which the initial problem state, goal state, and all other states that may be reached, can be represented



3 Software Development

- Computer programming is an example of an ISP. Even on a small scale, different people produce quite individual programs to meet a specific need
- So, more or less anything related to 'programming in the large' (software development) is going to be concerned with ISPs. Which means that SE will be largely concerned with 'solving' ISPs

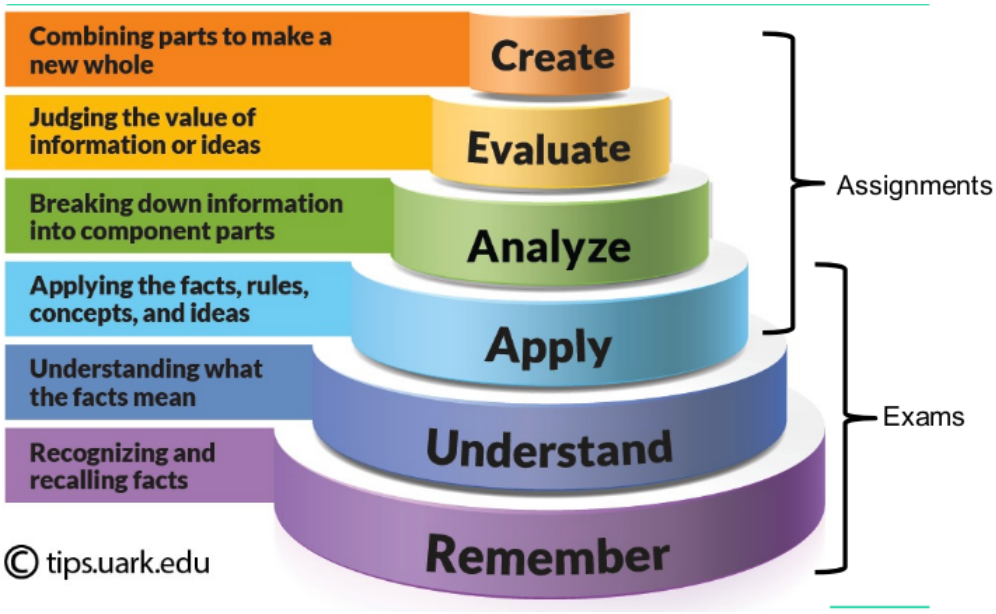
3.1 What is SE about?

Early thinking about how to do this sought to mix design concepts with formal and rigorous ideas drawn from science. However, as understanding has evolved it has become recognised that there is a strong element of social process involved too, and that human factors play a major role in the various activities.

4 Technical Debt

- Decisions in many software engineering activities will involve making trade-offs between different characteristics of a solution (design)
- A term often used in SE for a major form of trade-off is technical debt. This reflects the implied cost that additional later reworking of the system will impose if we choose an easy solution now, instead of going for a better solution that will take longer.
- This reflects a key difference of perspective related to software development: students rarely rework code; whereas in industry, it occurs extensively.

5 How ISP characteristics affect (exam) assessment



5.1 Exam words

Remembering - Distinguished by words such as 'describe'
Understanding - Needs to be demonstrated for 'explain'
Application - Involves some form of problem.

6 Design rather than scientific analysis

6.1 What do we mean by design

Product - Related to styling
Planning - Use familiar items, layouts
Engineering - Adapting past experience and domain knowledge to create a new way of doing something

6.2 Software Engineering

- A 'construction and use' perspective on computing
- Major knowledge elements include:
  - Systems/software concepts and strategies
  - Systems/software management
  - Computer concepts
- Knowledge is largely encapsulated as 'lessons' from experience (plan-driven methods) or in the form of models (such as diagrams)
- Research largely uses non-mathematical forms of analysis and 'concept implementation'

6.3 Example: Developing a Web Browser

- CS: Algorithms for finding pages, ways of formatting pages
- SE: Structuring of browser objects, development planning, coding, testing
- IS (Information Systems): What end users might use it for, how they organise the way they use it in order to meet their needs

Remember that design thinking applies to many of the activities that occur on software development, not just to 'system design'

7 Characteristics and Challenges

- Abstraction
- Static vs Dynamic properties
- Complexities
- Quantity measures
- (no) Manufacturing cycle
- Evolution
- People issues

8 Why Software Engineering Matters