Software Engineering

(Engineering of Software Subsystems)

Spring 2022 - Course Overview

Instructor: Y. Raghu Reddy

raghu.reddy@iiit.ac.in

Courses/Experiences so far…

- Undergrads at IIIT: You have learned technologies and low-level OO principles in workshop courses (Intro to Software Systems) and team-based software development in Design and Analysis of Software Systems.

- Grads at IIIT: Problem Solving course and Software Systems Development

- All others (includes PGSSP): Some **software development experience** or a course at the undergrad level in Software Engineering/Systems Engineering or any other variant of it.

Underlying Assumptions (Pre-requisites)

- SE principles: Abstraction, Modularization/Decomposition, Coupling, Cohesion, etc.
- Some Technologies (for example, python, JavaScript, web2py, IDE' s etc.) : at least 1 OOP language, & 1 RDBMS.
- Basic OO principles and implementations
  - Find the nouns → objects/state
  - Find the verbs → behaviors; methods/functions
  - Encapsulation, Inheritance, Polymorphism, etc.
- Introduction to static and dynamic modeling
- SDLC (Iterative Incremental process knowledge)
- Minimal SE practices (Version control, bug tracking, task management, etc.) and any associated tools.

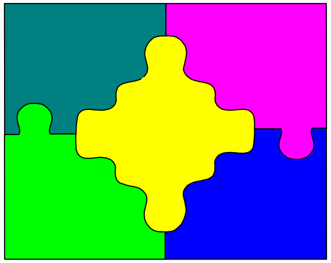
Bottom Line: You should be able to comprehend/Enhance CODE !!!

Why Study Software Engineering? (1)

- To acquire skills to develop large programs.
  - Exponential growth in complexity and difficulty level with size.
  - The ad hoc approach breaks down when size of software increases.

Why Study Software Engineering? (2)

- Ability to solve complex programming problems:
  - How to break large projects into smaller and manageable parts?
  - How to use abstraction?



- Also learn techniques of:
  - Specification, design, user interface development, testing, project management, etc.

Why Study Software Engineering? (3)

- To develop large, high quality software systems:
  - Large systems cannot be understood by one person
  - Requires team work
  - Achieve sufficient quality (e.g. Maintainability, Usability, etc)

Software Engineering – (major) Principles

Abstraction:

- Simplify a problem by omitting unnecessary details.
- Focus attention on only one aspect of the problem and ignore irrelevant details.

Decomposition:

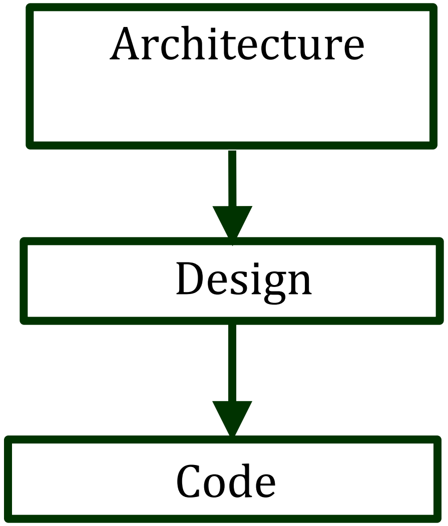
- Decompose a problem into many small independent parts.
  - The small parts are then taken up one by one and solved separately.
  - The idea is that each small part would be easy to grasp and can be easily solved.
  - The full problem is solved when all the parts are solved.

This Course is About…

Software Design

(moodle.iiit.ac.in)

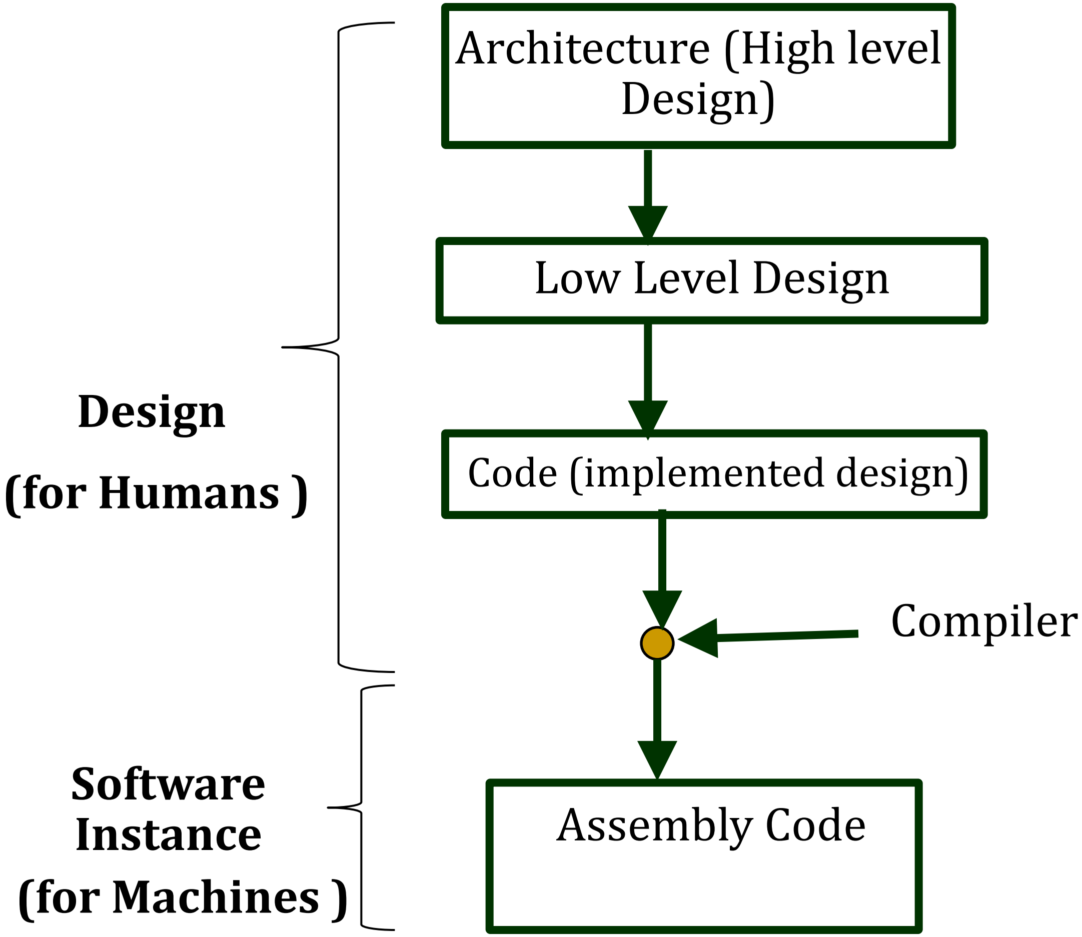
Why is design important?



Architecture

Design

Code



Design

(for Humans)

Software Instance

(for Machines)

Architecture (High level Design)

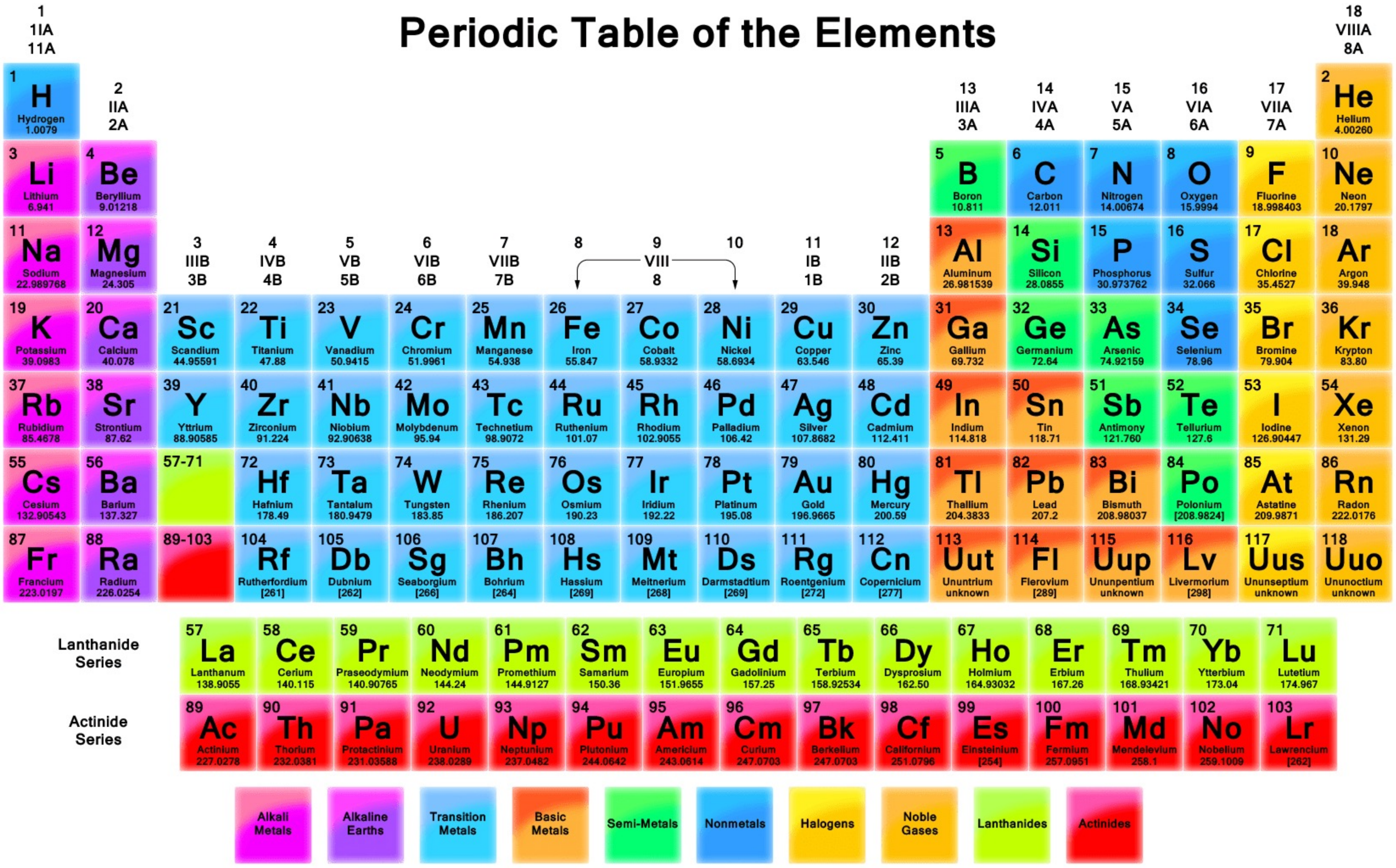
Low Level Design

Code (implemented design)

Compiler

Assembly Code

Periodic table – Why leave gaps?



Periodic Table of the Elements

1 1IA 11A	2 IIA 2A	3 IIIB 3B	4 IVB 4B	5 VB 5B	6 VIB 6B	7 VIIB 7B	8	9 VIII 8	10	11 IB 1B	12 IIB 2B	13 IIIA 3A	14 IVA 4A	15 VA 5A	16 VIA 6A	17 VIIA 7A	18 VIIIA 8A
1 H Hydrogen 1.0079																	2 He Helium 4.00260
3 Li Lithium 6.941	4 Be Beryllium 9.01218											5 B Boron 10.811	6 C Carbon 12.011	7 N Nitrogen 14.00674	8 O Oxygen 15.9994	9 F Fluorine 18.998403	10 Ne Neon 20.1797
11 Na Sodium 22.989768	12 Mg Magnesium 24.305											13 Al Aluminum 26.981539	14 Si Silicon 28.0855	15 P Phosphorus 30.973762	16 S Sulfur 32.066	17 Cl Chlorine 35.4527	18 Ar Argon 39.948
19 K Potassium 39.0983	20 Ca Calcium 40.078	21 Sc Scandium 44.95591	22 Ti Titanium 47.88	23 V Vanadium 50.9415	24 Cr Chromium 51.9961	25 Mn Manganese 54.938	26 Fe Iron 55.847	27 Co Cobalt 58.9332	28 Ni Nickel 58.6934	29 Cu Copper 63.546	30 Zn Zinc 65.39	31 Ga Gallium 69.732	32 Ge Germanium 72.64	33 As Arsenic 74.92159	34 Se Selenium 78.96	35 Br Bromine 79.904	36 Kr Krypton 83.80
37 Rb Rubidium 85.4678	38 Sr Strontium 87.62	39 Y Yttrium 88.90585	40 Zr Zirconium 91.224	41 Nb Niobium 92.90638	42 Mo Molybdenum 95.94	43 Tc Technetium 98.9072	44 Ru Ruthenium 101.07	45 Rh Rhodium 102.9055	46 Pd Palladium 106.42	47 Ag Silver 107.8682	48 Cd Cadmium 112.411	49 In Indium 114.818	50 Sn Tin 118.71	51 Sb Antimony 121.760	52 Te Tellurium 127.6	53 I Iodine 126.90447	54 Xe Xenon 131.29
55 Cs Cesium 132.90543	56 Ba Barium 137.327	57-71	72 Hf Hafnium 178.49	73 Ta Tantalum 180.9479	74 W Tungsten 183.85	75 Re Rhenium 186.207	76 Os Osmium 190.23	77 Ir Iridium 192.22	78 Pt Platinum 195.08	79 Au Gold 196.9665	80 Hg Mercury 200.59	81 Tl Thallium 204.3833	82 Pb Lead 207.2	83 Bi Bismuth 208.98037	84 Po Polonium [208.9824]	85 At Astatine 209.9871	86 Rn Radon 222.0176
87 Fr Francium 223.0197	88 Ra Radium 226.0254	89-103	104 Rf Rutherfordium [261]	105 Db Dubnium [262]	106 Sg Seaborgium [266]	107 Bh Bohrium [264]	108 Hs Hassium [269]	109 Mt Meitnerium [268]	110 Ds Darmstadtium [269]	111 Rg Roentgenium [272]	112 Cn Copernicium [277]	113 Uut Ununtrium unknown	114 Fl Flerovium [289]	115 Uup Ununpentium unknown	116 Lv Livermorium [298]	117 Uus Ununseptium unknown	118 Uuo Ununoctium unknown

Lanthanide Series	57 La Lanthanum 138.9055	58 Ce Cerium 140.115	59 Pr Praseodymium 140.90765	60 Nd Neodymium 144.24	61 Pm Promethium 144.9127	62 Sm Samarium 150.36	63 Eu Europium 151.9655	64 Gd Gadolinium 157.25	65 Tb Terbium 158.92534	66 Dy Dysprosium 162.50	67 Ho Holmium 164.93032	68 Er Erbium 167.26	69 Tm Thulium 168.93421	70 Yb Ytterbium 173.04	71 Lu Lutetium 174.967
Actinide Series	89 Ac Actinium 227.0278	90 Th Thorium 232.0381	91 Pa Protactinium 231.03588	92 U Uranium 238.0289	93 Np Neptunium 237.0482	94 Pu Plutonium 244.0642	95 Am Americium 243.0614	96 Cm Curium 247.0703	97 Bk Berkelium 247.0703	98 Cf Californium 251.0796	99 Es Einsteinium [254]	100 Fm Fermium 257.0951	101 Md Mendelevium 258.1	102 No Nobelium 259.1009	103 Lr Lawrencium [262]

Alkali Metals	Alkaline Earths	Transition Metals	Basic Metals	Semi-Metals	Nonmetals	Halogens	Noble Gases	Lanthanides	Actinides

How do You Design?

- What do you think about?
- What considerations are important?
- When have you done enough?

What This Course is About

- Standard *patterns* of interactions between classes/sub-systems?
  - Design patterns & Architectural patterns
- How to apply them to your application
  - Deal with subsystems at the higher level of abstraction provided by the patterns
- What to do when it does not fit exactly
  - Evaluate options and analyze the trade-offs
- How do you document the design knowledge (very important, given the focus on AGILE delivery models)

Do You Always Reinvent the Wheel?

- Consider code level patterns

  How do you walk through an array in Java?

```
for (i = 0;i < array.length;i++) {
    // use the array element
}
```

Our Design Level

- Higher than what we've done before
  - Not specific data structures
  - Not algorithmic approaches
- Lower than complex system level architectures
  - Not financial systems
  - Not air-traffic control
- Interactions of 10-20 classes in solution domain.

  i.e., the small sized subsystem

Problem-based learning methodology

- Solving problems motivates your learning
- Lecturing is minimal
- This is better because
  - Learner actively engages the material
  - Deeper learning when learner motivates need for knowledge
  - More closely resembles true career situation
- Over the past few years students were very positive about this approach and seemed to learn a lot more