SOFTENG 306 SOFTWARE ENGINEERING DESIGN 2 LECTURE

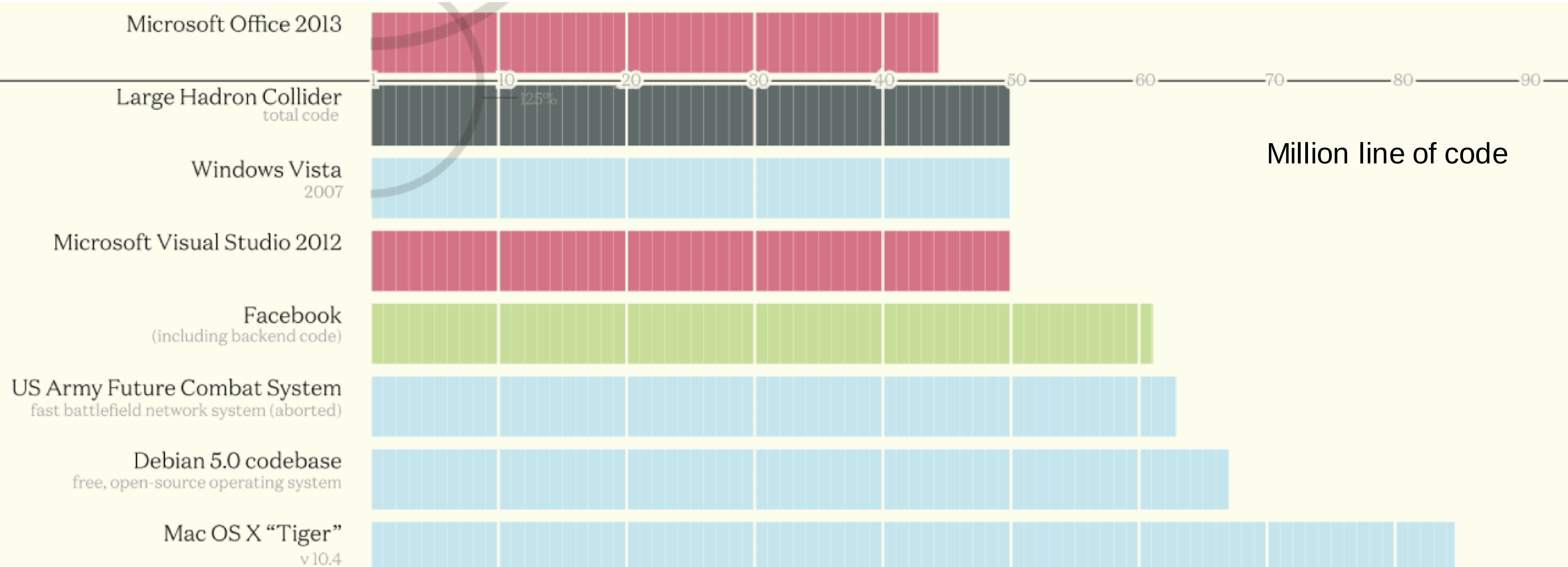
01 – SOFTWARE DESIGN - BACKGROUND

Dr. Seyed Reza Shahamiri [More Info]

How many of these questions can you answer?

1. What is your favourite SOLID principle, and why?
2. What are the pros and cons of test-first vs test-second development?
3. What do you hate to see when you're reviewing code?
4. In your opinion what is ‘good code’?
5. What do you look for in your ideal team in terms of practices, process, and technology?
6. Do you have any public contributions or work? (e.g. projects, open source, blog articles, or social media).
7. Describe your experience writing code in a team setting.

Complexity of Modern Software



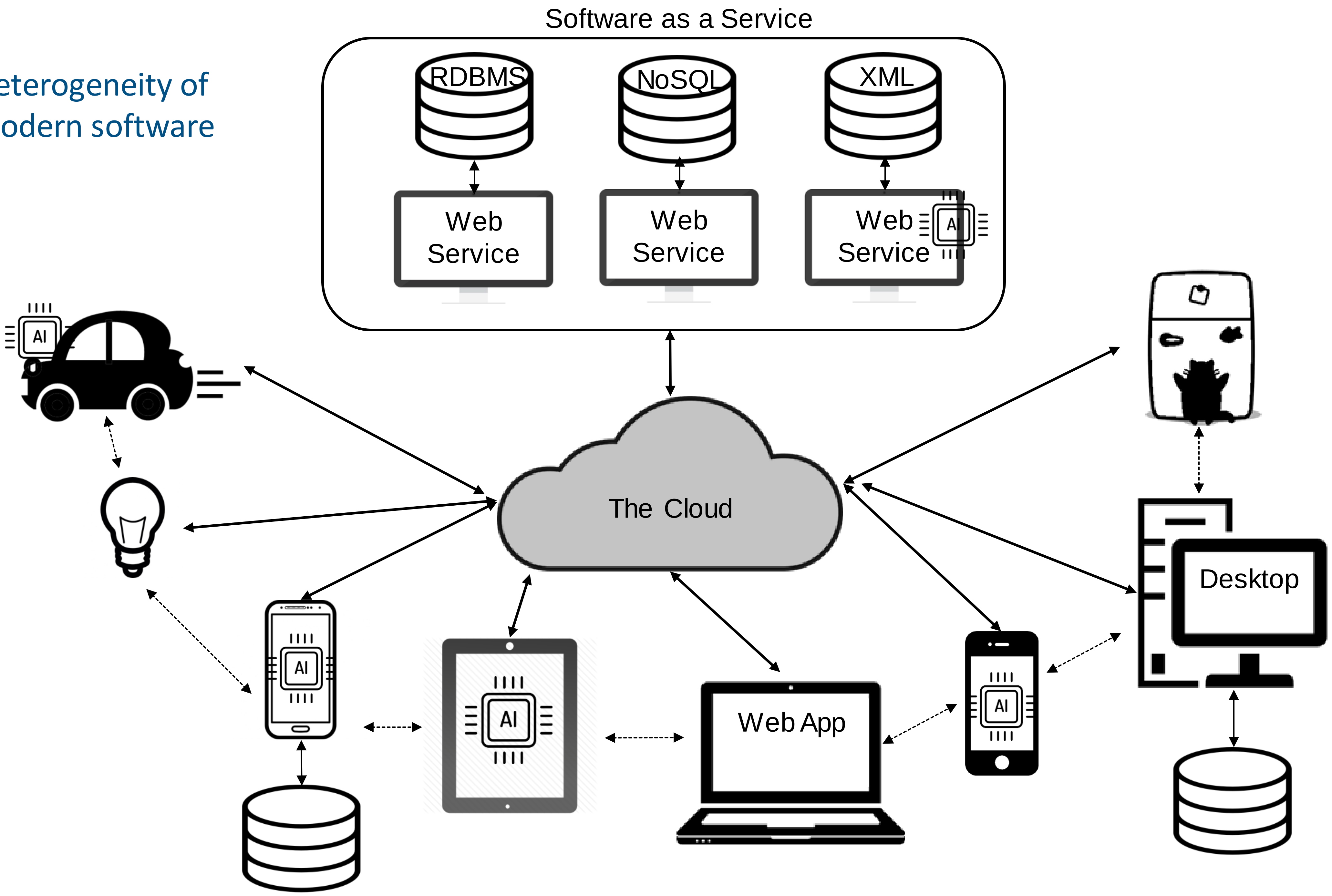
Million line of code

General issues in all types of software

1. Heterogeneity:

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices probably programmed in different languages and/or technologies, or to be integrated to older legacy systems.
- The challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity.

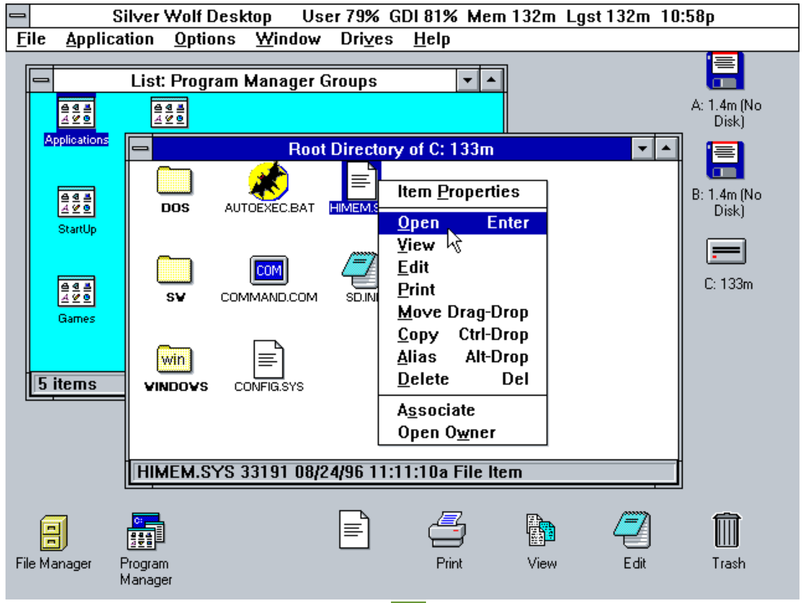
Heterogeneity of modern software



General Issues in all types of software

2. Business and social change

- Businesses and societies are changing incredibly quickly as emerging economies develop and new technologies become available.
- They need to be able to change their existing software and to rapidly develop new software.
- Tradition software development techniques need to evolve so that the time required for software to deliver values to its customers is reduced.



Silver Wolf Desktop User 79% GDI 81% Mem 132m Lgst 132m 10:58p
File Application Options Window Drives Help
List: Program Manager Groups
Root Directory of C: 133m
Item Properties
Open Enter
View
Edit
Print
Move Drag-Drop
Copy Ctrl-Drop
Alias Alt-Drop
Delete Del
Associate
Open Owner
HIMEM.SYS 33191 08/24/96 11:11:10a File Item
File Manager
Program Manager
Print
View
Edit
Trash

Windows 3.1 to Windows 10

General Issues in all types of software

3. Security and trust

- As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
- This is especially true for remote software systems accessed through a web page or web service interface.
- We have to make sure that malicious users cannot attack our software and that information security is maintained.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

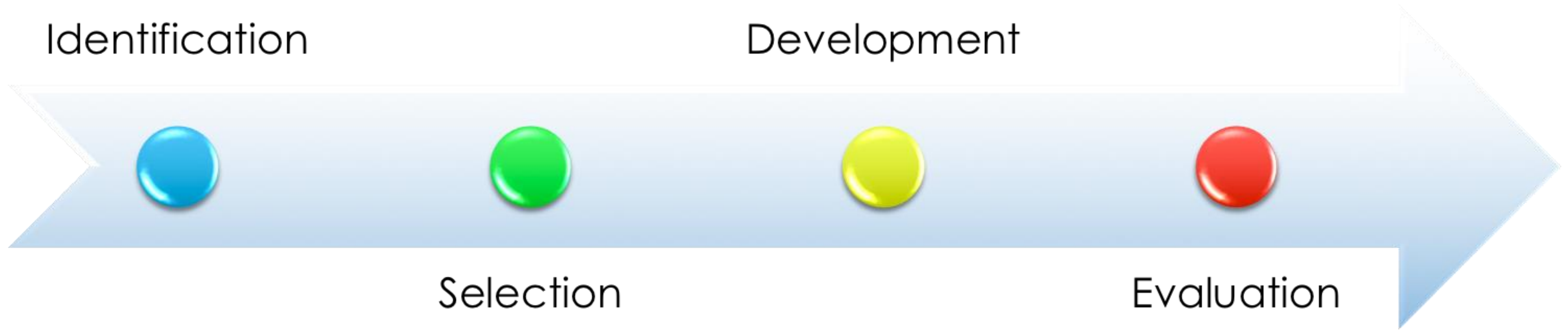
Engineering Design Process

- The typical assumption of software design is that the process can proceed in the general manner of architectural design or engineering design, which can be summarized as:
  1. Feasibility stage: identifying a set of feasible concepts for the design as a whole. In particular, a set of alternative arrangements for the design as a whole is quickly derived.
  2. Preliminary design stage: selection and development of the best concept.
     - In particular, one of the alternatives identified by stage 1 is selected for further development.

Engineering Design Process

3. Detailed design stage: development of engineering descriptions of the concept.
4. Planning stage: evaluating and altering the concept to suit the requirements of production, distribution, consumption and product retirement.

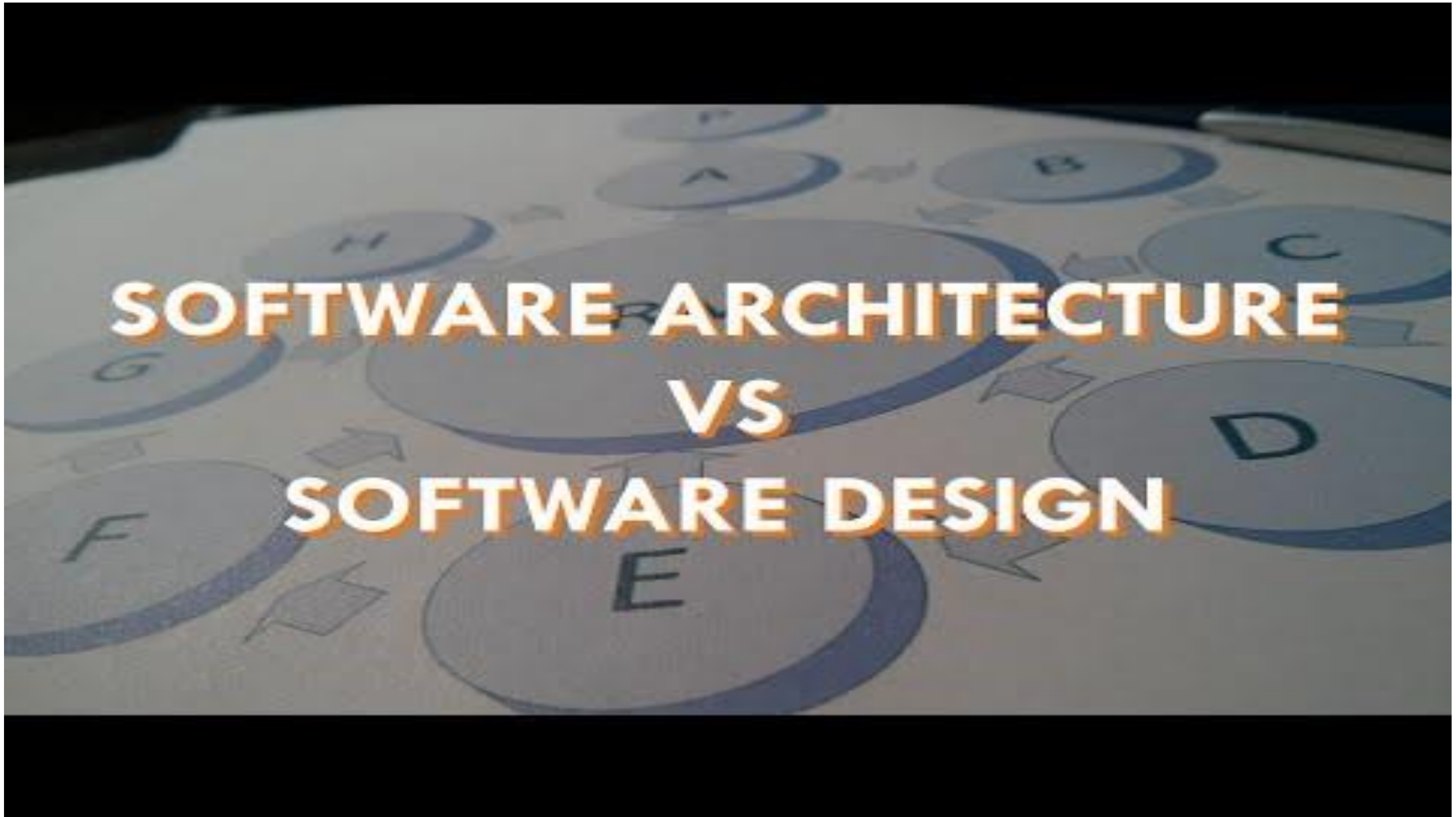
Engineering Design Process



Non-Functional Properties (NFP)

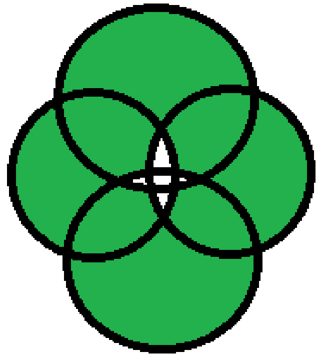
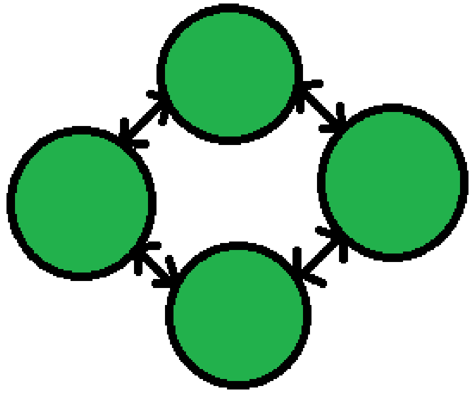
- NFPs are the result of architectural and design choices.
- NFP questions are raised as the result of architectural and design choices.
- Specification of NFP might require an architectural framework to even enable their statement.
- An architectural framework will be required for assessment of whether the properties are achievable.

Design vs Architecture



Coupling

- Coupling is the degree of interdependence between software modules:
  - A measure of how closely connected two routines or modules are
  - The strength of the relationships between modules

Tight coupling:
1. More Interdependency
2. More coordination
3. More information flow

Loose coupling:
1. Less Interdependency
2. Less coordination
3. Less information flow

- Adopted from https://en.wikipedia.org/wiki/Coupling_(computer_programming)
- Image from https://www.geeksforgeeks.org/coupling-in-java/

Coupling

- Tightly coupled systems tend to exhibit the following disadvantages:
  1. A change in one module usually forces a ripple effect of changes in other modules.
  2. Assembly of modules might require more effort and/or time due to the increased inter-module dependency.
  3. A particular module might be harder to reuse and/or test because dependent modules must be included.

Cohesion

- Cohesion refers to the degree to which the elements inside a module belong together.
  - For example, it is a measure of the strength of relationship between the methods and data of a class and some unifying purpose or concept served by that class.
  - In another sense, it is a measure of the strength of relationship between the class's methods and data themselves.
- High cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability.

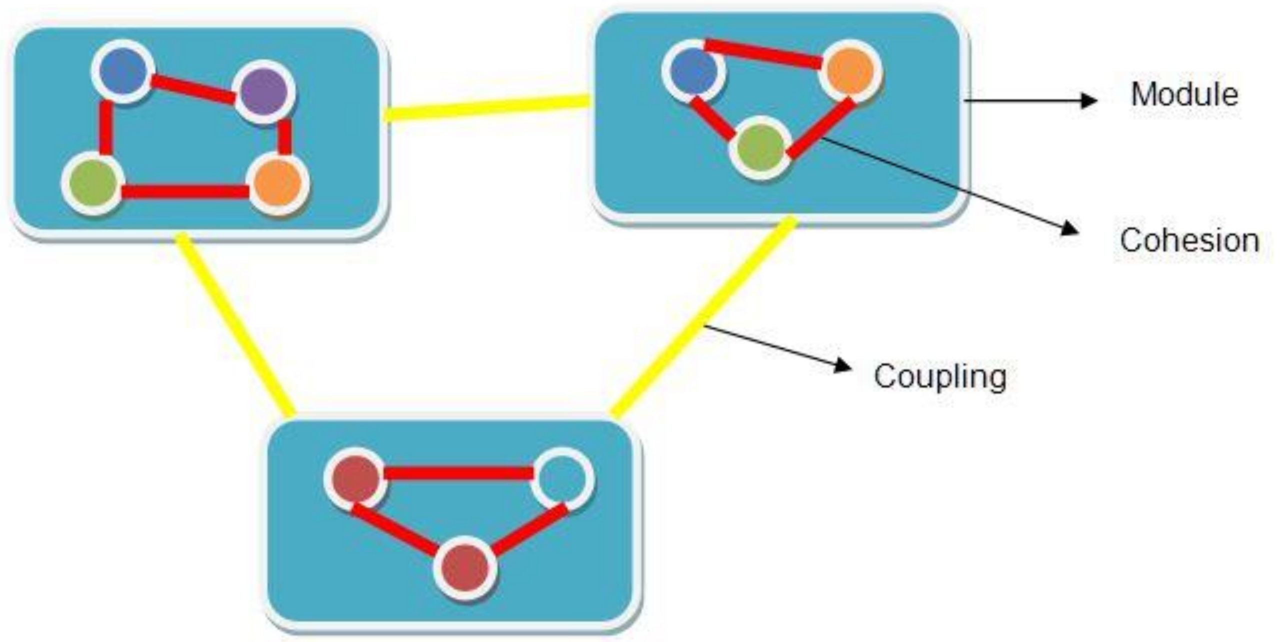
Coupling vs Cohesion
Separation of Concerns

- Separation of concerns is the subdivision of a problem into (hopefully!) independent parts.
- For example, the visual interface that a customer sees at an ATM is independent from the logic used to control operation on that customer’s accounts.
- The difficulties arise when the issues are either actually or apparently intertwined (i.e. closely related to each other) (highly coupled).

Coupling vs Cohesion

Separation of Concerns

- Less coupling is always better (less dependency).
- High coherency is always better (i.e. more consistency).



Coupling vs Cohesion

Separation of Concerns

- Separations of concerns frequently involves many tradeoffs.
  - The designer is left with the substantial obligation of assessing performance, cost, appearance, or functional trade-offs between competing conceptions.
- Total independence of concepts may not be possible.