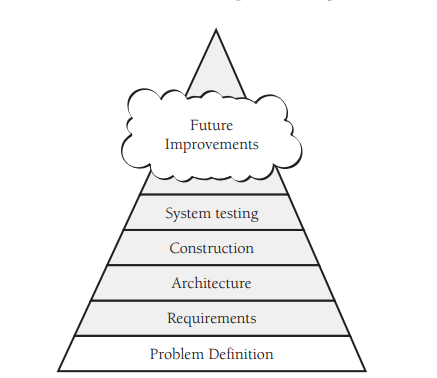
**PHƯƠNG PHÁP LẬP TRÌNH**

**Nguyến Phi Quốc Thanh**

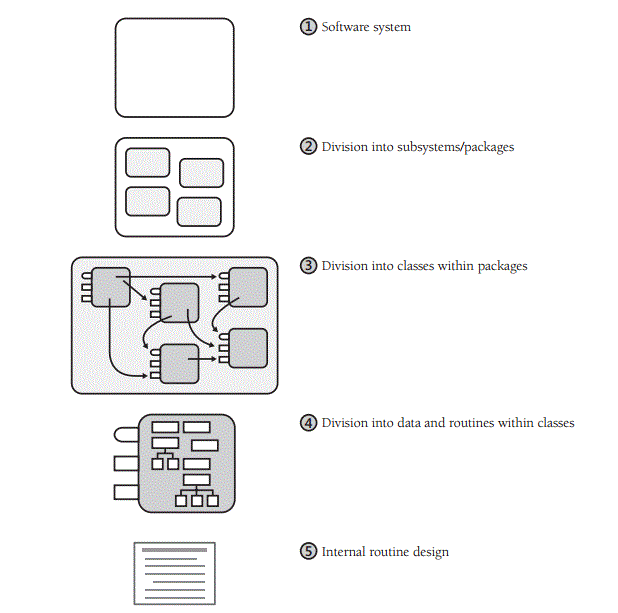
1. **Tổng quan về phát triển phần mềm**



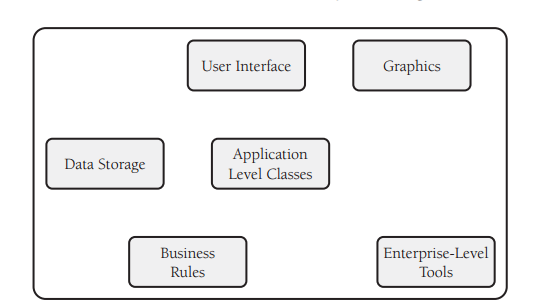
* Phát triển hệ thống phần mềm gồm các bước: Định nghĩa vấn đề, xây dựng các yêu cầu cho hệ thống, thiết kế kiến trúc, viết chương trình, kiểm tra
* Trước khi làm 1 hệ thống, cần biết hệ thống đó sẽ làm cái gì và sẽ làm như thế nào.
* Dự án sẽ tốt nếu những sự chuẩn bị thích hợp được thực hiện trước khi vào lập trình.
* Mục tiêu của sự chuẩn bị: giảm rủi ro
* Các bước cần làm trước khi lập trình:
  + **Định nghĩa vấn đề**: định nghĩa về hệ thống. Cần dễ hiểu, không đề cập giải pháp thiết kế, giải pháp lập trình trong phần này
  + **Xây dựng các yêu cầu của hệ thống:** Yêu cầu ổn định là điều quan trọng để dự án thành công. Khi yêu cầu rõ ràng sẽ tránh được sự tranh cãi trong quá trình làm.
  + **Thiết kế kiến trúc:** Kiến trúc phần mềm bao gồm 1 số thành phần như: cách tổ chức chương trình, các khối chính, cấu trúc dữ liệu, giao diện người dùng, cơ sở dữ liệu, quản lý tài nguyên, bảo mật, hiệu suất,…..

1. **Thiết kế chương trình**

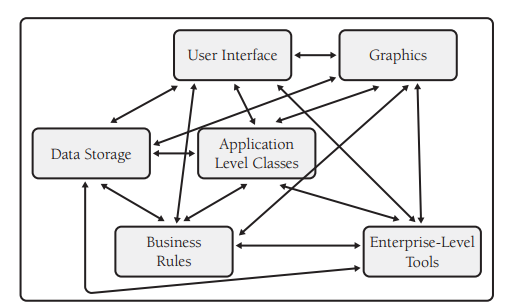
* Có rất nhiều cách để thiết kế 1 chương trình.
* **Các đặc tính mà 1 thiết kế cần đạt được:**
  + **Giảm sự phức tạp**: Nếu 1 dự án thất bại vì yếu tố kỹ thuật (ngoài yếu tố yêu cầu, kế hoạch, quản lý) thì đó là vì không kiểm soát được sự phức tạp.
  + Nên tổ chức chương trình mà ở 1 thời điểm, ta chỉ cần tập trung vào 1 phần và có thể bỏ qua những phần khác
  + Ở cấp kiến trúc, để giảm sự phức tạp, ta có thể chia nhỏ hệ thống thành nhiều hệ thống nhỏ.
  + Tránh tạo ra những thiết kế lạ, độc đáo, vì nó thường khó hiểu. Thay vào đó hãy tạo ra thiết kế đơn giản, dễ hiểu. Thiết kế phải cho phép ta có thể tập trung vào 1 phần mà ko cần quan tâm đến những phần #
  + **Dễ bảo trì**: Xây dựng chương trình mà khi 1 người khác đọc vào thấy dễ hiểu.
  + **Loose Coupling:** Giảm thiểu các sự kết nối giữa các phần trong hệ thống 1 cách tối đa
  + **Khả năng mở rộng:** mở rộng hệ thống mà ko va chạm, ảnh hưởng đến các kiến trúc nền tảng. Thay đổi 1 phần mà ko ảnh hưởng đến phần #.
  + **Khả năng sử dụng lại:** Sử dụng lại những phần khác
  + **Kĩ thuật thông dụng:** Sử dụng những kĩ thuật, giải thuật thông dụng, dễ hiểu để ko làm hệ thống khó hiểu
* **Các mức độ thiết kế**



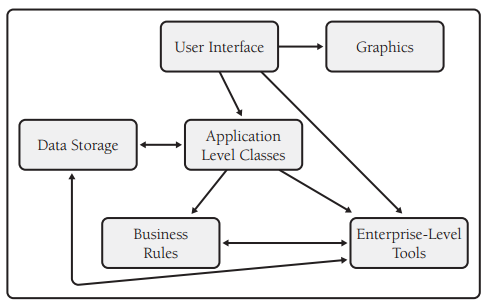
* + **Level 1: Software system**
  + **Level 2: Division into subsystems or packages**
  + Chia hệ thống thành những hệ thống phụ
  + Các hệ thống phụ có thể là DB, UI, log,system dependencies, …
  + Nếu tất cả hệ thống phụ có thể giao tiếp với tất cả hệ thống khác, thì ta mất đi lợi ích từ việc phân chia các hệ thống phụ, do đó cần hạn chế sự giao tiếp giữa các phần
  + Giả sử có 1 hệ thống



* + Nếu ta ko giới hạn sự giao tiếp giữa các khối, sự phức tạp sẽ tăng lên



* + Mỗi hệ thống phụ có thể giao tiếp trực tiếp với tất cả các hệ thống #, tạo ra câu hỏi : Khi thay đổi 1 thứ gì đó trong 1 khối, có bao nhiêu phần khác mà ta cần phải quan tâm.
  + 1 phần giao tiếp với phần khác chỉ khi nó thực sự cần biết về phần đó



* + **Level 3: Division into Classes**
  + Thiết kế ở mức độ này bao gồm xác định tất cả các class của hệ thống
  + **Level 4**
  + Thiết kế data, method cho các class
  + **Level 5**
  + Gồm viết code giả, tìm giải thuật, tổ chức code trong 1 hàm, và viết code