Trình bày lại được tầm quan trọng của kỹ thuật phần mềm, tổng quan về 5 mô hình quy trình phần mềm.

Cuộc khủng hoảng phần mềm :
- cuối những năm 1960 khi nhiều dự án phần mềm thất bại.
-Nhiều phần mềm đã trở nên vượt quá ngân sách.
-Đầu ra là một phần mềm không đáng tin cậy, tốn kém để duy trì.
- Phần mềm lớn hơn là khó khăn và khá tốn kém để duy trì.
- nhiều phần mềm không thể đáp ứng các yêu cầu ngày càng tăng của khách hàng.
- Sự phức tạp của các dự án phần mềm tăng lên bất cứ khi nào khả năng phần cứng của nó tăng lên.
- Nhu cầu về phần mềm mới tăng nhanh hơn so với khả năng tạo ra phần mềm mới.

- KTPM là ngành chuyên nghiên cứu và phát triển các quy trình, cách hoạt động của những ứng dụng công nghệ, được dùng để lập trình phần mềm hoặc ứng dụng
- là việc thực hiện các thao tác thiết kế, xây dựng, kiểm tra và bảo vệ các phần mềm
- Tầm quan trọng: tìm ra các biện pháp giúp ngành công nghiệp phần mềm tránh được nguy cơ khủng hoảng
- Giúp hoàn thành dự án 1 cách chất lượng đúng hạn trong chi phí cho phép
- giúp tăng hiệu suất làm việc đáp ứng được nhu cầu ngày càng tăng của khách hàng
- Duy trì phần mềm với chi phí thấp
**** Các mô hình cho việc xdung qui trình***
- **Mô hình thác nước**; Tiến triển theo chuỗi thứ tự các bước từ kniem khởi đầu đến gdoan cuối của phần mềm (Áp dụng qui mô vừa và nhỏ, dự án có yêu cầu rõ ràng ít thay đổi)
Ưu điểm: Tìm ra lỗi sớm ngay trong các gdoan đầu giúp giảm thiểu chi phí
nhược điểm: ko linh hoạt, khó xdinh đầy đủ yêu cầu khi bắt đầu dự án

Knao dùng mô hình này: Khi xác định sản phẩm ổn định và n vấn đề kĩ thuật đã biết rõ như tạo phiên bản mới hoặc xdung spham tương tự như sp đã tồn tại
- Mô hình Xoắn ốc ;Chú trọng vào phân tích rủi ro dự án. Mỗi giai đoạn trong mô hình được bắt đầu với yêu cầu/mục tiêu thiết kế và kết thúc với việc khách hàng kiểm tra tiến độ của từng giai đoạn
Các gdoan:Lập kế hoạch , Phân tích rủi ro ,Thực thi kỹ thuật ,Đánh giá
Ưu điểm: Phân tích rủi ro sâu rộng làm giảm rủi ro,
+ nhận dc feedback sớm tư khách hàng
+Phần mềm dc đưa ra sớm trong vòng đời
nhược điểm: + cần có chuyên môn cao khi ptich rủi ro
+ phức tạp hơn mô hình khác, tốn kém nếu thực hiện
+Thành công tổng thể of qtrinh phụ thuộc vào ptich rủi ro
- **Mô hình Agile(** qtrinh phát triển phần mềm linh hoạt) là nhóm các phương pháp phát triển phần mềm dựa trên sự phát triển gia tăng và lặp lại cao
- Là lập trình cấp độ cao hay gọi là XP, Scrum
+ Có thể dùng ở bất kì dự án nào n cần sự tham gia và tương tác của khách hàng
Ưu điểm:
Tăng cường tình thần làm việc nhóm và trao đổi công việc hiệu quả.
Các chức năng được xây dựng nhanh chóng và rõ ràng, dế quản lý.
Dễ dàng bổ sung, thay đổi yêu cầu.

Quy tắc tối thiểu, tài liệu dễ hiểu, dễ sử dụng.
Nhược điểm:
Không thích hợp để xử lý các phụ thuộc phức tạp.
Có nhiều rủi ro về tính bền vững, khả năng bảo trì và khả năng mở rộng.
Cần một team có kinh nghiệm.
- Mô hình tiến hóa bản mẫu
+ Khi dùng bản mẫu tiến hóa, hệ thống liên tục được cải tiến và xây dựng lại
-phát triển các phần of kthuat thay vì phát triển toàn bộ hthong
Ưu điểm: + phản hồi ngay lập tức, đưa bản mẫu cho khách và dc feedback. có thể giảm tối thiểu rủi ro of việc thực hiện sai hthong
nhược điểm: - khó lập kế hoạch trc cho khoảng tgian mà việc ptrien sẽ chiếm
Qui trình phần mềm thống nhất RUP
Làm việc theo cách lặp lại thể hiện các lần lặp khác nhau.

2. Phân biệt được các giai đoạn phát triển cơ bản trong một số quy trình phần mềm phổ biến.
*Giai đoạn phát triển cơ bản
- QUi trình phần mềm là cách thức chia nhỏ nhiệm vụ chưa thể qly thành các bước nhỏ hơn để giải quyết, phá vỡ sự phức tạp 1 cách có hệ thống
1. Giải pháp yêu cầu: Thực hiện khảo sát chi tiết yêu cầu của khách hàng để từ đó tổng hợp vào tài liệu giải pháp. Tài liệu này phải mô tả đầy đủ các yêu cầu về chức năng, phi chức năng và giao diện.
Kết quả: Đầu ra của giai đoạn này là Tài liệu đặc tả yêu cầu
- Đây là giai đooạn quan trọng, tổn thất chi phí nhỏ nhất. Càng đến các gdoan sau chi phí càng lớn dần vdu để sửa chữa gdoan kiểm thử ta phải hoàn tác các qui trình trc đã đưa ra để sửa lỗi
2. Thiết kế: Các yêu cầu được phân tích để đưa ra mô tả về cấu trúc và tchuc bên trong hệ thống
Có các gdoan: Thiết kế kiến trúc, đặc tính lý thuyết, thiết kế giao diện, thiết kế tphan, cấu trúc dlieu, thiết kế thuật toán
- Kết quả có dc: Tài liệu thiết kế tổng thể, thiết kế module, thiết kế CSDL
3. Lập trình: lập trình dựa trên tài liệu Giải pháp và Thiết kế đã được phê duyệt.
- Kết quả: mã nguồn (sourse code)
4. Test kiểm thử Tester tạo kịch bản kiểm thử (test case) theo tài liệu đặc tả yêu cầu, thực hiện kiểm thử và cập nhật kết quả vào kịch bản kiểm thử, log lỗi trên các tool quản lý lỗi. kiểm tra xem ht phần mềm có đáp ứng yêu cầu đặc điểm kthuat of nó hay ko
+ Kiểm thử từng đơn vị riêng lẻ
+ kiểm tra tương tác giữa các đơn vị khác nhau
+ Kiểm thử tổng thể hệ thống đảm bảo tất cả làm việc với nhau đúng cách
Kết quả: Test case , lỗi trên hệ thống quản lý lỗi.
5. Triển khai: Triển khai sản phẩm cho khách hàng.
- Đầu ra biên bản triển khai với khách

3. Mô tả được mối quan hệ giữa hệ thống quản lý phiên bản VCS và GIT, cũng như cách cài đặt và các luồng công việc của GIT
- hệ thống qly phiên bản là một hệ thống lưu trữ các thay đổi của một tập tin (file) hoặc tập

hợp các tập tin theo thời gian, do đó nó giúp bạn có thể quay lại một phiên bản xác định nào đó sau này.
* VCS tập trung: source code của dự án sẽ được lưu trữ trên một kho kho trung tâm trên một máy chủ. Muốn tạo thay đổi cho source code cần update source code trên máy tính với kho tập trung trước sau đó mới được thay đổi.

* VCS dạng phân tán thì mỗi lập trình viên sẽ có riêng một kho (repository) của mình và có thể tự do phát triển mà không cần phải đồng bộ với kho trên server.
+ ko cần update code trên mtinh với server thường xuyên giúp giảm thiểu xugn đột . Nếu có xung đột chỉ cần xử lý 1 lần
VCS và Git:
+ qly code lsu thay đổi
+ làm việc nhóm trao đổi code với nhau nếu ko sdung email, chat phức tạp mất time hơn
* khác nhau: VCS theo hướng tập trung (subversion...) >< git theo hướng phân tán
* Subversion coi thông tin được lưu trữ như là một tập hợp các tập tin
* Git coi dữ liệu của nó giống như một tập hợp các "ảnh" (snapshot) của một hệ thống tập tin nhỏ. Mỗi lần bạn "commit", hoặc lưu lại trạng thái hiện tại của dự án Git "chụp một bức ảnh" ghi lại nội dung của tất cả các tập tin tại thời điểm đó và tạo ra một tham chiếu tới "ảnh" đó
* Git có khả năng tách nhánh, hỗ trợ team chia task hay tổng hợp code
+ tập trung gquyet từng task ko lo a.hưởng đến task khác
+ linh hoạt hơn khi dùng nhiều task
+ git free
* GIT hệ thống kiểm soát phiên bản phân tán
+ Hệ thống giúp lưu trữ tất cả thay đổi thay đổi của soure code
+ Hỗ trợ nhiều ng làm việc cùng lúc
+ Phát hiện ai là ng sửa code, gây lỗi +khôi pục tệp lỗi, xóa nhầm
***luồng công việc GIT**
Clone dữ liệu từ Repo trên máy chủ như Github, gitlab về máy local để thực hiện chỉnh sửa, thay đổi… Sau đó add và commit. Và push lên máy chủ để đồng bộ với repo trên đó
Hoặc khi có ng khác sửa trên server thì ta chạy lệnh pull hoặ fetch để đồng bộ dữ liệu từ repo trên máy chủ về máy local
- Tạo nhánh để tạo bản sao of code chung về máy, sau đó chỉnh sửa mà ko anh hưởng đến các code khác. Sau khi sửa xong tạo pull request để cấp trên review, nếu ổn xếp sẽ merge vào kho chung
- Workingspace: bản sao phiên bản của 1 dự án nơi ta đang thao tác thay đổi dữ liệu trên local
- staging area (index) : vùng tổ chức các file thay đổi, đánh dấu các file thay đổi sẽ lưu lại thành 1 version khi commit. Để đưa vào vùng này dùng lệnh (git add file)
4. Trình bày được định nghĩa về yêu cầu phần mềm, các kỹ thuật thu thập yêu cầu và đặc tả yêu cầu.
- Yêu cầu phần mềm: thiết lập các dịch vụ mà khách hàng yêu cầu từ hệ thống phần mềm. Phải thực hiện hết n ràng buộc mà theo đó hệ thống vận hành và được phát triển
-*Các kỹ thuật thu thập yêu cầu: Là giai đoạn đầu tiên trong việc xây dựng sự hiểu biết về sản phẩm phần mềm và các vấn đề cần thiết phải giải quyết . Thiết lập các mối quan hệ giữa các nhóm phát triển và khách hàng. Cần xác định dc các yêu cầu như:

+ Yêu cầu nghiệp vụ (business requirements)
+Yêu cầu của các bên liên quan (stakeholder requirements
+Yêu cầu chức năng (Functional requirements): Các yêu cầu xác định các khía cạnh chức năng của phần mềm +Yêu cầu phi chức năng (Nonfunctional requirements):Lquan đến thuộc tính phi chức năng of hệ thống, chất lượng hệ thống vdu: độ bảo mật, hiệu suất, chi phí, khả năng thích ứng, tương tác...
-yêu cầu người dùng và hệ thống:
+yêu cầu ng dùng: viết cho khách hàng, viết bằng ngôn ngữ tự nhiên, ko chứa chi tiết kthuat
+ ycau hệ thống: viết cho các nhà phát triển, chứa các ycau chức năng và phi chức năng
* Lợi ích của việc thu thập yêu cầu là:
-Tạo được niềm tin của khách hàng khi họ được tham gia vào giai đoạn thu thập yêu cầu.
-Giảm việc phải làm lại trong quá trình phát triển
-Quá trình phát triển sẽ nhanh hơn, giảm được những chi phí cho những yêu cầu không cần thiết.
*** đặc tả yêu cầu:** Đặc tả yêu cầu là một mô tả của hệ thống phần mềm được phát triển, đưa ra các yêu cầu chức năng và phi chức năng, và có thể bao gồm một tập hợp các ca sử dụng (use cases) để mô tả tương tác giữa người dùng với phần mềm.
- SRS là tài liệu đặc tả có vai trò:
+ Giúp cho các bên thứ ba – stakeholders đều hiểu được hệ thống theo cùng một hướng, tránh trường hợp mỗi người một ý.
+Giúp cho đội phát triển xây dựng hệ thống một cách chính xác, đặc tả được các tính năng đáp ứng ycau của khách
+SRS giúp nhà kiểm thử hệ thống đọc hiểu từ đó xây dựng nên kịch bản kiểm thử chi tiết nhất.
+Giúp cho việc bảo trì hệ thống và cải tiến những chức năng của hệ thống một cách nhanh chóng và dễ dàng.
*** các bước viết tài liệu đặc tả**
- Giới thiệu: nói về mục đích, bối cảnh và mục tiêu của dự án
+ nên biểu diễn sơ đồ ngữ cảnh để thể hiện mối quan hệ của spham với các hthong khác
-*Yêu cầu của người dùng:*
+ lớp ng dùng: xác định lớp người dùng khác nhau sẽ sử dụng spham này và mô tả các đặc điểm thích hợp cho họ
+ các tính năng chính: nhấn mạnh tính năng khác biệt với spam cạnh tranh, dùng tính năng đó xdinh ycau cụ thể of ng dùng. VDU: tạo, xem sửa xóa thực đơn. Đặt hàng thành toán…
- *Sơ đồ ca sử dụng:* biểu đồ use case là một tập hợp các tác nhân, các use case và các mối quan hệ giữa chúng
+ trả lời câu hỏi ai sử dụng hệ thống, hệ thống nào tương tác với hthong nào. Xdinh quan hệ giữa các actor, cac actor sdung chức năng gì...
- *TỔng quan về hệ thống:*
+ vai trò của ng dùng; xdinh ai có vtro gì
+ chức năng hthong:
+ luồng màn hình biểu diễn các màn hình hệ thống và mối quan hệ giữa các màn hình.vẽ dạng sơ đồ
+ mô tả màn hình
+ chức năng ko lquan màn hình

- hệ thống cập quyền màn hình
- Yêu cầu chức năng
- yêu cầu phi chức năng như: độ bảo mật, hiệu suất, chi phí, khả năng thích ứng, tương tác...

5. **So sánh được cách lập sơ đồ lớp UML với các loại biểu đồ UML thông dụng khác.**

SƠ đồ lớp class diagram thể hiện 1 cái nhìn mang tính cấu trúc, tĩnh về hệ thống. Nó mô tả các lớp và cấu trúc of chúng. Mối quan hệ giữa các lớp of hệ thống
- biểu diễn hình chữ nhật:
* class name: các lớp đặt tên bằng từ vựng of tên miền
+ là những danh từ số ít dc viết bắt đầu bằng chữ in hoa
* thuộc tính of lớp:+ trạng thái cho lớp (là private)
+ liệt kê những thuộc tính đơn giản qua tên
* hoạt động of lớp(phương thức): theo tên với danh sách các đối số (public)
- dấu – trong thuộc tính đại diện cho tư nhân
- dấu + cho công chúng và các lớp khác có thể truy cập các thành viên đó

Biểu đồ đối tượng: Nó tương tự như một biểu đồ lớp, nhưng nó tập trung vào các đối tượng, giúp hiểu hành vi của đối tượng và mối quan hệ của chúng

khác nhau:
- Sơ đồ lớp: xác định lớp và chỉ ra cách chúng lquan đến nhau
+ các lớp là bản thiết kế
+ tên lớp bắt đầu bằng chữ hoa
- Biểu đồ đối tượng cho thấy các đối tượng và mối quan hệ
+ các đối tượng là các thể hiện của of lớp
+ tên đối tượng viết thường và dc gạch dưới

VÍ dụ sơ đồ lớp:
Tên lớp: Users
- thuộc tính: UserID, password,regist date
+ hoạt động: login

Sơ đồ đối tượng:
Tên : Khóa học
các đối tượng: Sinh viên 1, sinh viên 2,svien 3, giảng viên.

- *Sơ đồ thành phần*: (component diagram) biểu đồ mô tả các thành phần và sự phụ thuộc của chúng trong hethong
+ chia hệ thống thành các subsystem
+ xác định quan hệ các thành phần và vẽ
- *Biểu đồ triển khai* thể hiện kiến trúc thực thi của hệ thống, bao gồm các nút như môi trường thực thi phần cứng hoặc phần mềm và phần mềm trung gian kết nối chúng.

Bước 1: Xác định mục đích của sơ đồ triển khai của bạn.

Bước 2: Tìm ra mối quan hệ giữa các nút và thiết bị.

Bước 3: Xác định những yếu tố khác như các thành phần, các đối tượng hoạt động mà bạn cần thêm vào để hoàn thành sơ đồ.

- *Biểu đồ tuần tự* là biểu đồ dùng để xác định các trình tự diễn ra sự kiện của một nhóm đối tượng nào đó

VD:

- *Biểu đồ trạng thái* là dạng biểu đồ mô tả các trạng thái có thể có và sự chuyển đổi giữa các trạng thái đó khi có các sự kiện tác động của một đối tượng.

- *Biểu đồ ca sử dụng:*

- tìm actor (Ai sử dụng hệ thống, Hệ thống nào tương tác với hệ thống này)

- tìm use case (Actor sử dụng chức năng gì trong hệ thống)

- tìm quan hệ (xdinh qhe giữa các actor, giữa các use case)

VD: actor là ng mua hàng.

Các usecase: đăng nhập, order, thanh toán