

# Nhập môn Công nghệ phần mềm

## Tuần 14: Phát triển phần mềm linh hoạt



KHOA CÔNG NGHỆ THÔNG TIN  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung của slide này được dịch và hiệu chỉnh dựa vào các slides của Ian Sommerville

# Nội dung

1. Các phương pháp linh hoạt
2. Extreme programming
3. Quản trị dự án linh hoạt
4. Mở rộng quy mô các phương pháp linh hoạt

# Nội dung

1. Các phương pháp linh hoạt
2. Extreme programming
3. Quản trị dự án linh hoạt
4. Mở rộng quy mô các phương pháp linh hoạt

# Phát triển phần mềm nhanh

- ☐ Phát triển và phân phối phần mềm nhanh thường là yêu cầu quan trọng nhất đối với hệ thống phần mềm hiện nay
- ☐ Phát triển phần mềm theo phương pháp hoạch định sẵn thường cần thiết cho một số hệ thống nhưng không đáp ứng được yêu cầu của các tác vụ thương mại.
- ☐ Phương pháp phát triển phần mềm linh hoạt nổi lên từ cuối những năm 90 để giúp giảm thời gian phân phối các hệ thống phần mềm sử dụng được.

# Phát triển linh hoạt

- ☐ Đặc tả, thiết kế và cài đặt đan xen nhau.
- ☐ Hệ thống được phát triển như là một chuỗi các phiên bản trong đó stakeholder tham gia vào việc đánh giá các phiên bản.
- ☐ Phân phối phiên bản mới thường xuyên để đánh giá
- ☐ Giao diện người dùng thường được phát triển sử dụng IDE và các công cụ đồ họa.
- ☐ Giảm thiểu tài liệu, tập trung vào mã nguồn hoạt động được

## □ Phát triển hoạch định sẵn

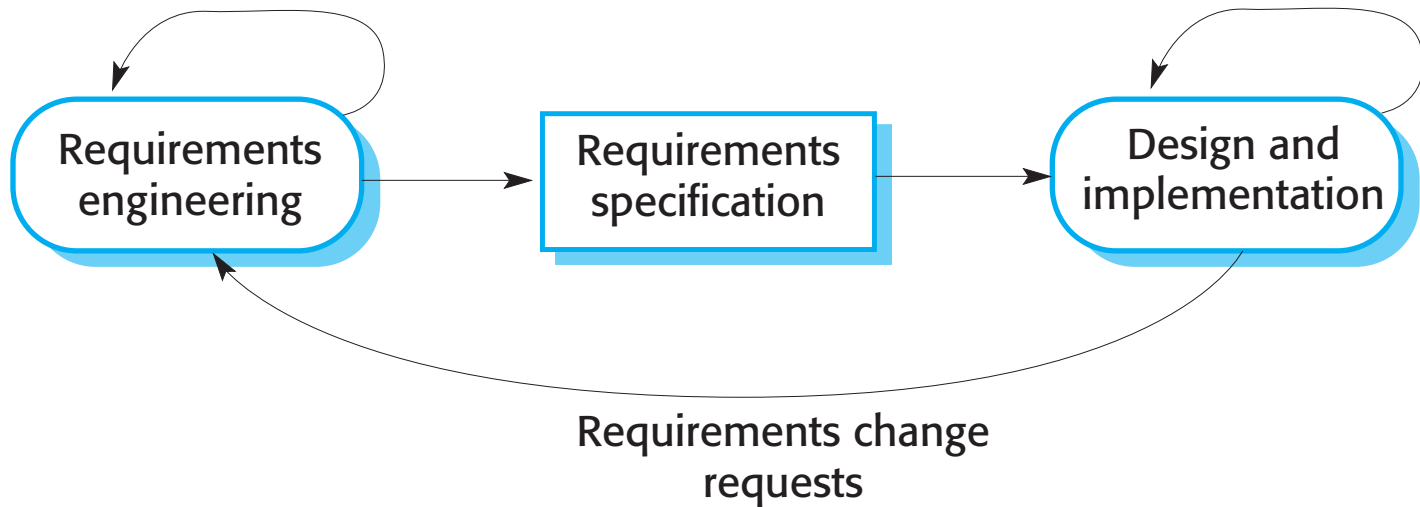
- Dựa vào các giai đoạn phát triển tách biệt,
- Đầu ra ở mỗi giai đoạn đã được lên kế hoạch trước,
- Không nhất thiết phải là mô hình thác nước, có thể là phương pháp phát triển dần dần,
- Vòng lặp xảy ra bên trong các hoạt động.

## □ Phát triển linh hoạt

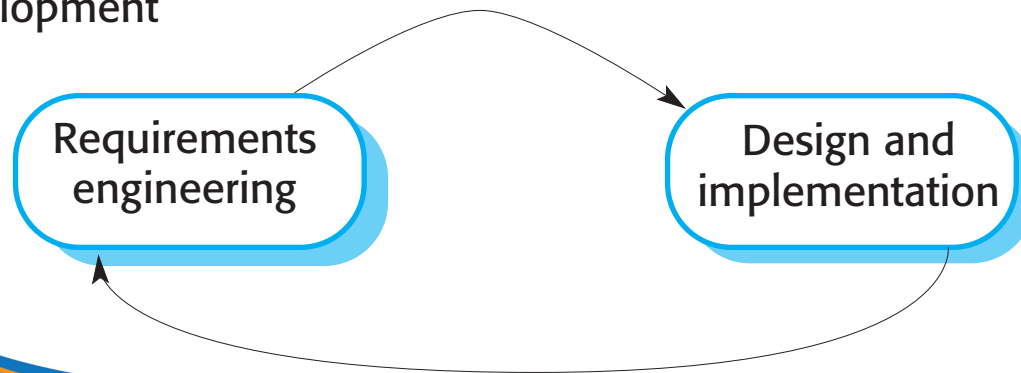
- Đặc tả, thiết kế, cài đặt và kiểm thử đan xen nhau và
- Đầu ra từ quy trình phát triển được quyết định thông qua việc thương lượng trong suốt quá trình phát triển phần mềm.

# Đặc tả linh hoạt và hoạch định sẵn

## Plan-based development



## Agile development



# Bối cảnh và mục tiêu

- Vào những năm 80 và 90, việc không thỏa mãn với các phụ phí trong các phương pháp thiết kế phần mềm dẫn đến việc tạo ra các phương pháp linh hoạt:
  - Tập trung vào mã nguồn hơn là thiết kế.
  - Dựa vào phương pháp phát triển phần mềm theo kiểu vòng lặp.
  - Với mục đích phân phối sản phẩm phần mềm nhanh và cải tiến nhanh để đáp ứng các yêu cầu thay đổi.
- Mục tiêu: giảm các phụ phí trong quy trình phần mềm
  - Bằng việc hạn chế việc viết tài liệu và cho phép trả lời nhanh các thay đổi về yêu cầu mà không cần làm lại quá nhiều.



# Tuyên ngôn

- Chúng tôi đang tìm ra những cách tốt hơn để phát triển phần mềm bằng cách tự tay phát triển nó và giúp đỡ người khác làm việc đó. Thông qua việc này, chúng tôi đã đi đến chỗ đánh giá cao:
  - **Các cá nhân và tương tác** hơn là quy trình và công cụ
  - **Phần mềm hoạt động được** hơn là tài liệu đầy đủ
  - **Sự cộng tác của khách hàng** hơn là thương lượng hợp đồng
  - **Trả lời nhanh sự thay đổi** hơn là làm theo kế hoạch
- Đó là, dù là các điểm bên phải có giá trị, nhưng chúng tôi đánh giá cao các điểm **bên trái** hơn.

# Nguyên lý

Nguyên lý	Mô tả
Sự tham gia của khách hàng	Khách hàng nên tham gia trực tiếp vào quy trình phát triển. Vai trò: - cung cấp các yêu cầu mới và xác định độ ưu tiên và - đánh giá các vòng lặp của hệ thống.
Phân phối dần dần	Phần mềm được phát triển từng phần (increment) trong đó khách hàng chỉ ra yêu cầu trong mỗi phần đó.
Chú trọng vào con người hơn là quy trình	Kỹ năng của nhóm phát triển nên được nhận diện và khai thác. Các thành viên của nhóm nên được tự do làm việc theo cách của họ mà không cần đến các quy trình định trước.
Chấp nhận thay đổi	Hiểu rằng yêu cầu hệ thống sẽ thay đổi, vì vậy hệ thống phải được thiết kế sao cho có thể chấp nhận các thay đổi đó.
Duy trì sự đơn giản	Tập trung vào tính đơn giản của toàn bộ phần mềm và quy trình phát triển. Bất cứ khi nào có thể, nên chủ động loại bỏ những điểm phức tạp khỏi hệ thống.

# Phạm vi ứng dụng

- ☐ Phát triển các sản phẩm nhỏ và vừa để bán.
- ☐ Phát triển các sản phẩm đặt hàng trong đó
  - ☐ việc khách hàng chấp nhận tham gia vào quy trình phát triển được xác nhận rõ ràng và
  - ☐ không có nhiều quy tắc và quy định bên ngoài ảnh hưởng lên phần mềm.
- ☐ Trọng tâm tập trung vào các nhóm phát triển nhỏ, gắn kết chặt chẽ với nhau → phát sinh nhiều vấn đề khi mở rộng phương pháp linh hoạt cho các hệ thống lớn.

# Nội dung

1. Các phương pháp linh hoạt
2. Phát triển hoạch định sẵn và linh hoạt
3. **Extreme programming**
4. Quản trị dự án linh hoạt
5. Mở rộng quy mô các phương pháp linh hoạt

# Các phương pháp linh hoạt

- ☐ Agile Modeling
- ☐ Agile Unified Process (AUP)
- ☐ Dynamic Systems Development Method (DSDM)
- ☐ Essential Unified Process (EssUP)
- ☐ **Extreme Programming (XP)**
- ☐ Feature Driven Development (FDD)
- ☐ Open Unified Process (OpenUP)
- ☐ **Scrum**
- ☐ Velocity tracking

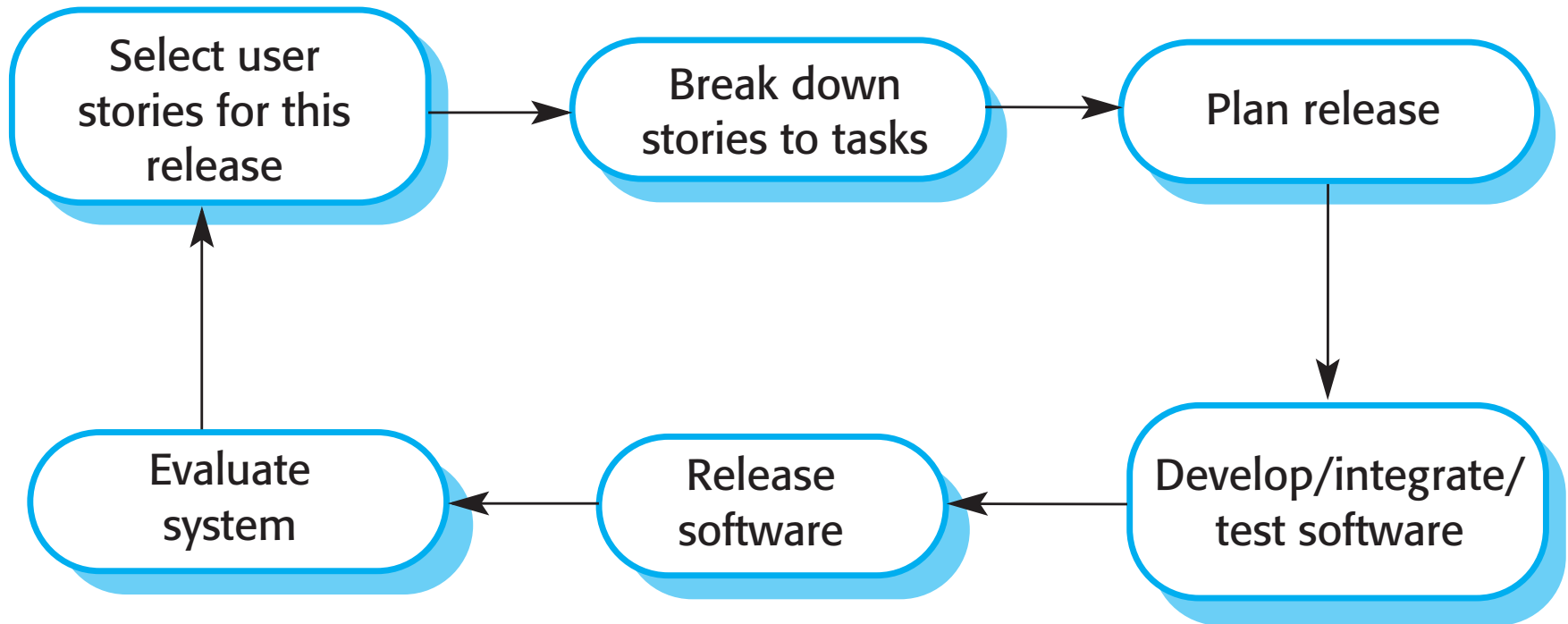
# Extreme programming (XP)

- Được xem là phương pháp linh hoạt nổi tiếng và được sử dụng rộng rãi nhất.
- Có cách tiếp cận “cực đoan” đối với việc phát triển vòng lặp.
  - Các phiên bản mới có thể được xây dựng **vài lần mỗi ngày**;
  - Các phần được phân phối đến khách hàng **hai tuần một lần**;
  - **Tất cả các test phải được chạy ở mỗi phiên bản** và phiên bản đó chỉ được chấp nhận nếu các test đều thành công.

# XP và phương pháp linh hoạt

- Việc phát triển từng phần được hỗ trợ thông qua các bản release nhỏ, thường xuyên.
- Sự tham gia của khách hàng đồng nghĩa với việc cam kết tham gia toàn thời gian với đội ngũ phát triển.
- Đặt nặng yếu tố con người hơn là quy trình thông qua lập trình cặp, sở hữu tập thể và một quy trình hạn chế làm việc nhiều giờ.
- Các thay đổi được hỗ trợ thông qua các bản release thường xuyên.
- Duy trì tính đơn giản thông qua việc cải thiện thường xuyên mã nguồn.

# Vòng lặp tạo ra các bản release trong XP





# Các practice của XP[1]

Nguyên tắc	Mô tả
Lập kế hoạch tăng dần	<ul style="list-style-type: none"> <li>Các yêu cầu được ghi lại trên các story card,</li> <li>việc lựa chọn các story cho một bản release là tùy thuộc vào thời gian và mức độ ưu tiên tương đối giữa chúng,</li> <li>Người phát triển chia các story thành các tác vụ.</li> </ul>
Các bản release nhỏ	<ul style="list-style-type: none"> <li>Một tập tối thiểu các chức năng hữu ích mang lại giá trị công việc được phát triển đầu tiên.</li> <li>Các bản release được đưa ra thường xuyên và thêm tính năng dần dần vào bản release đầu tiên.</li> </ul>
Thiết kế đơn giản	<ul style="list-style-type: none"> <li>Chỉ thiết kế vừa đủ để thỏa mãn các yêu cầu hiện tại và không hơn.</li> </ul>
Phát triển test trước	<ul style="list-style-type: none"> <li>Sử dụng framework cho kiểm thử đơn vị để viết các test cho một chức năng trước khi cài đặt tính năng đó.</li> </ul>
Cải tiến mã nguồn	<ul style="list-style-type: none"> <li>Cả nhóm phát triển tham gia vào việc cải thiện liên tục mã nguồn bất cứ khi nào có điểm cần cải tiến.</li> <li>Việc này làm cho mã nguồn trở nên đơn giản và dễ bảo trì.</li> </ul>

# Các practice của XP[2]

Nguyên tắc	Mô tả
Lập trình cặp	<ul style="list-style-type: none"> <li>Người phát triển làm việc theo cặp,</li> <li>người này kiểm tra công việc của người kia và hỗ trợ để đảm bảo công việc luôn luôn tốt.</li> </ul>
Sở hữu tập thể	<ul style="list-style-type: none"> <li>Các cặp phát triển làm việc trong mọi khía cạnh của hệ thống, để không xảy ra tình trạng mỗi người chỉ thông thạo một vùng và tất cả các thành viên của nhóm phát triển chịu trách nhiệm cho toàn bộ mã nguồn.</li> <li>Bất cứ ai cũng có thể sửa bất cứ cái gì.</li> </ul>
Tích hợp liên tục	<ul style="list-style-type: none"> <li>Khi một tác vụ được hoàn thành → tích hợp ngay</li> <li>Sau mỗi lần tích hợp, tất cả các unit test phải được chạy thành công.</li> </ul>
Tiến độ bền vững	<ul style="list-style-type: none"> <li>Không chấp nhận làm việc quá giờ quá nhiều do hệ quả thường là giảm chất lượng mã nguồn và giảm năng suất làm việc.</li> </ul>
Khách hàng tại chỗ	<ul style="list-style-type: none"> <li>Một người đại diện của người dùng cuối (khách hàng) sẽ luôn luôn sẵn sàng tham gia.</li> <li>Khách hàng là một thành viên của nhóm phát triển và có trách nhiệm đưa ra các yêu cầu để nhóm cài đặt.</li> </ul>

# Các practice XP có ảnh hưởng

- ☐ XP tập trung vào kỹ thuật và không dễ tích hợp với các practice về quản lý trong hầu hết các tổ chức
- ☐ Một số practice chính
  - ☐ Sử dụng các user story để đặc tả
  - ☐ Refactor mã nguồn
  - ☐ Sử dụng phương pháp TDD
  - ☐ Lập trình cặp

# Kịch bản yêu cầu

- Trong XP, khách hàng là một phần của nhóm phát triển và có trách nhiệm đưa ra các quyết định về yêu cầu.
- Yêu cầu người dùng được biểu diễn dưới dạng các kịch bản hoặc user story.
  - ▣ Được viết trên các story card và
  - ▣ Chia nhỏ các story thành các tác vụ để cài đặt → cơ sở để lập kế hoạch và ước lượng chi phí.
- Khách hàng chọn các story cần được đáp ứng trong bản release tiếp theo
  - ▣ Dựa vào độ ưu tiên và ước lượng về kế hoạch.

# Một story về kê đơn thuốc

## Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

# Ví dụ về card tác vụ cho việc kê đơn thuốc

## **Task 1: Change dose of prescribed drug**

## **Task 2: Formulary selection**

## **Task 3: Dose checking**

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.

Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.

Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low. If within the range, enable the 'Confirm' button.

# Cải tiến mã nguồn

- Nhóm lập trình tìm kiếm các phần có thể thực hiện cải tiến được và tiến hành cải tiến
  - Ngay cả khi những cải tiến này không cần ngay.
  - Làm tăng tính dễ hiểu của phần mềm → giảm được tài liệu.
  - Các thay đổi dễ thực hiện hơn do mã nguồn có cấu trúc và rõ ràng.
- Tuy nhiên, một số thay đổi đòi hỏi phải cải tiến kiến trúc hệ thống sẽ đòi hỏi chi phí lớn.

# Ví dụ về cải tiến mã nguồn

- ☐ Tổ chức lại cây phân cấp lớp
- ☐ Thay đổi tên các thuộc tính và phương thức để làm cho mã nguồn dễ hiểu hơn.
- ☐ Thay thế các inline code bằng việc gọi các phương thức đã có sẵn trong thư viện chương trình.
- ☐ ...



# XP và sự thay đổi

- Nguyên tắc chung cho CNPM truyền thống: thiết kế cho sự thay đổi. Đó là
  - Dự đoán các thay đổi trong tương lai,
  - Thiết kế sao cho các thay đổi này có thể được cài đặt dễ dàng
  - Đó là dành thời gian và nỗ lực cho việc dự đoán các thay đổi vì điều này sẽ làm giảm chi phí sau này.
- Với XP: nguyên tắc trên sẽ không có giá trị nếu các thay đổi không được dự đoán một cách đáng tin.
  - Thay vào đó: việc cải tiến mã nguồn liên tục giúp thay đổi sau này dễ dàng hơn.

# Kiểm thử trong XP

- XP nhấn mạnh tầm quan trọng của việc kiểm thử chương trình
  - chương trình được kiểm thử cho mỗi thay đổi.
- Đặc điểm của kiểm thử trong XP:
  - Phát triển test trước.
  - Phát triển test tăng dần từ các kịch bản.
  - Người dùng tham gia vào quá trình phát triển test và thẩm định.
  - Sử dụng các framework kiểm thử tự động.

# Phát triển theo hướng test

- Việc viết test trước khi viết mã nguồn làm rõ các yêu cầu cần cài đặt.
- Các test được viết như là chương trình hơn là dữ liệu để có thể chạy tự động.
  - ▣ Thường sử dụng một framework để kiểm thử: JUnit chẳng hạn.
- Tất cả các test cũ và mới được chạy tự động khi một tính năng mới được cài đặt
  - ➔ kiểm tra được tính năng mới không gây ra lỗi.

# Sự tham gia của khách hàng

## □ Vai trò:

- hỗ trợ phát triển các acceptance test cho các kịch bản sẽ được cài đặt trong bản release tiếp theo.
- Viết test cho nhóm phát triển. Tất cả các mã nguồn mới phải được thẩm định để đảm bảo rằng nó đáp ứng được nhu cầu của khách hàng.

## □ Tuy nhiên:

- những người đóng vai trò khách hàng thường có ít thời gian rỗi và họ không thể tham gia toàn phần vào nhóm phát triển.
- Họ có thể cảm thấy việc cung cấp yêu cầu là đủ và có thể sẽ không hào hứng với việc tham gia quy trình test.

# Khó khăn của kiểm thử trong XP

- ☐ Người lập trình thích lập trình hơn là kiểm thử
  - ☐ Thường đi tắt trong việc viết test
- ☐ Một số test khó có thể viết theo kiểu tăng dần
  - ☐ Ví dụ: trong giao diện người dùng phức tạp, thường khó viết các unit test cho đoạn mã cài đặt việc hiển thị và các luồng chuyển tiếp giữa các màn hình.
- ☐ Khó đánh giá được tính đầy đủ của bộ test
  - ☐ Bộ test cũng không thể bao phủ hết được tất cả.

# Lập trình cặp

- ☐ Người lập trình làm việc theo cặp, ngồi cùng nhau tại một nơi để phát triển mã nguồn.
- ☐ Khuyến khích cải tiến mã nguồn vì toàn bộ nhóm phát triển có thể hưởng lợi từ việc này.
- ☐ Năng suất của việc lập trình cặp tương đương với năng suất của hai người làm việc độc lập.
- ☐ Các cặp được tạo ra một cách linh động
- ☐ Việc chia sẻ kiến thức xảy ra khi làm việc cặp là quan trọng → Giảm thiểu các nguy cơ cho dự án khi có thành viên rời khỏi nhóm.

# Lợi ích của lập trình cặp

- ☐ Hỗ trợ ý tưởng sở hữu nhóm và trách nhiệm chung đối với hệ thống.
- ☐ Hoạt động như một quy trình duyệt không chính thức
- ☐ Hỗ trợ cải tiến mã nguồn

# Nội dung

1. Các phương pháp linh hoạt
2. Phát triển hoạch định sẵn và linh hoạt
3. Extreme programming
4. Quản trị dự án linh hoạt
5. Mở rộng quy mô các phương pháp linh hoạt



# Quản trị dự án linh hoạt

- Trách nhiệm chính của một người quản trị dự án phần mềm là quản lý dự án sao cho phần mềm được phân phối đúng hạn và trong hạn mức tài chính cho phép đã được lên kế hoạch trước.
  - ▣ Phương pháp chuẩn của quản trị dự án là hoạch định sẵn
  - ▣ Người quản lý lên kế hoạch cho dự án, chỉ ra rằng cái gì nên được phân phối, khi nào nó được phân phối và ai sẽ tham gia vào việc phát triển các phần sẽ được phân phối đó.
- Việc quản trị dự án linh hoạt yêu cầu một cách tiếp cận khác, thích nghi với việc phát triển tăng dần và các điểm mạnh của phương pháp linh hoạt.



# Scrum

- ☐ Là phương pháp tổng quát.
- ☐ Tập trung vào quản lý việc phát triển vòng lặp hơn là các nguyên lý về phương pháp linh hoạt.

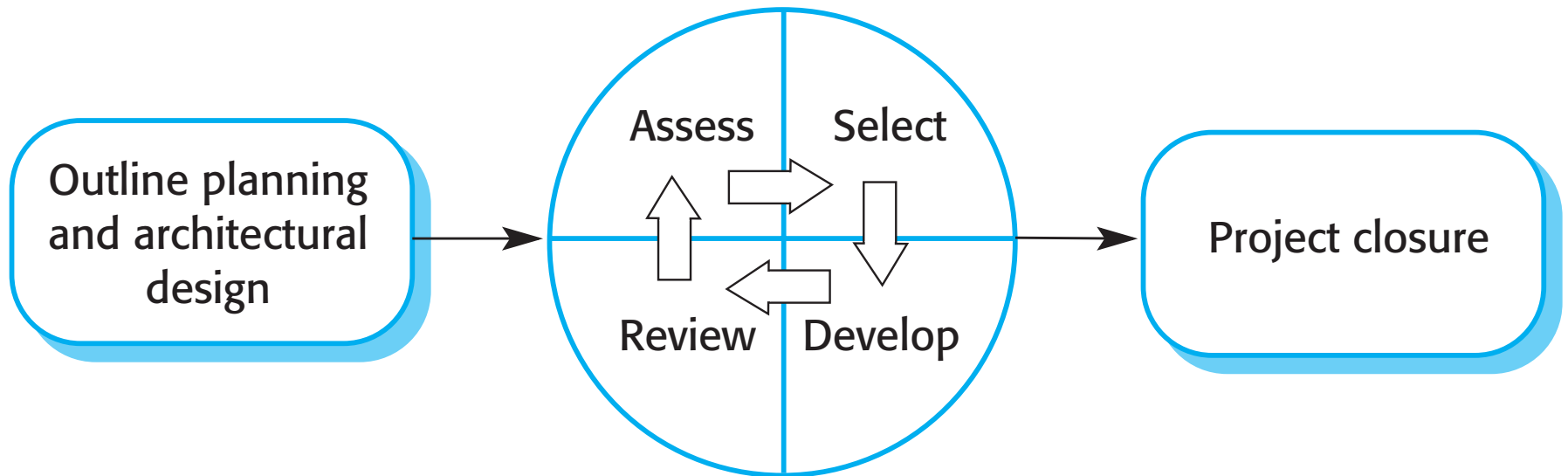
# Thuật ngữ Scrum (a)

Scrum term	Definition
Development team	Một nhóm tự tổ chức các lập trình viên, không nên nhiều hơn 7 người. Nhóm chịu trách nhiệm cho việc phát triển phần mềm và các tài liệu cần thiết cho dự án.
Potentially shippable product increment	Các phần nhỏ được phát triển từ một sprint. Ý tưởng là phần này nên là 'potentially shippable', nghĩa là nó ở trạng thái hoàn thành, không có thêm tác vụ cần xử lý như kiểm thử chẳng hạn, cần tích hợp vào sản phẩm cuối. Thực tế, việc này khó đạt được.
Product backlog	Đây là 'to do' list mà nhóm Scrum phải giải quyết. Danh sách này có thể là các định nghĩa về đặc tính của phần mềm, các yêu cầu phần mềm, các user story hay mô tả các tác vụ bổ sung cần thiết, ví dụ như định nghĩa về kiến trúc hay tài liệu người dùng.
Product owner	Một cá nhân (có thể một nhóm nhỏ) chịu trách nhiệm nhận diện các đặc tính, chức năng của hệ thống, xếp độ ưu tiên, và review lại các product backlog để đảm bảo rằng dự án đáp ứng được các yêu cầu nghiệp vụ. Product Owner có thể là khách hàng nhưng cũng có thể là một product manager trong một công ty phần mềm hoặc các đại diện stakeholder khác.

# Thuật ngữ Scrum (b)

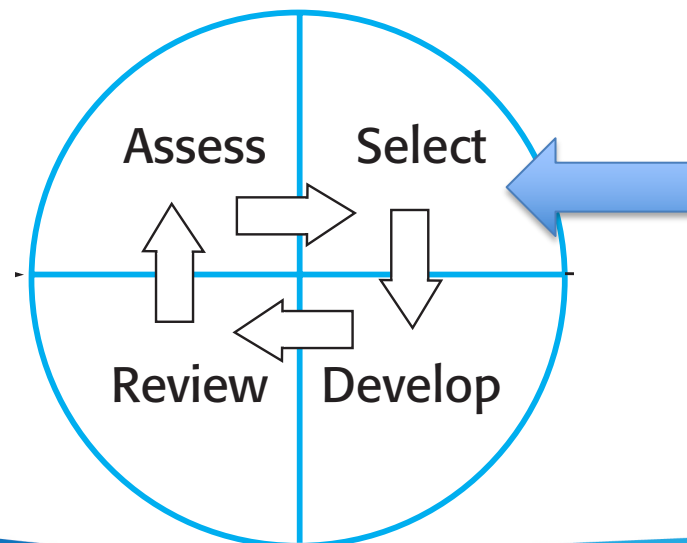
Scrum terms	Definition
Scrum	Một cuộc họp ngắn hàng ngày của nhóm Scrum để review lại tiến độ và xếp độ ưu tiên các việc phải làm trong ngày hôm đó. Lý tưởng, đây nên là một cuộc họp trực tiếp có tất cả các thành viên trong nhóm.
ScrumMaster	ScrumMaster chịu trách nhiệm đảm bảo rằng quy trình Scrum được tuân thủ và chỉ dẫn nhóm sử dụng Scrum một cách hiệu quả. Người này chịu trách nhiệm là “giao diện” của nhóm, giao tiếp với phần còn lại của công ty để đảm bảo rằng nhóm Scrum không bị chi phối bởi yếu tố bên ngoài. Nhóm phát triển Scrum quả quyết rằng không nên nghĩ ScrumMaster là người quản trị dự án. Tuy nhiên, một số người lập luận rằng không dễ dàng để nhận ra sự khác nhau.
Sprint	Là một vòng lặp phát triển. Sprint thường kéo dài 2-4 tuần.
Velocity	Một ước lượng về nỗ lực mà nhóm thực hiện trong một sprint. Việc hiểu velocity của nhóm giúp cho việc ước lượng cái gì có thể thực hiện trong một sprint và cung cấp cơ sở để đo việc cải thiện hiệu năng

# Quy trình Scrum



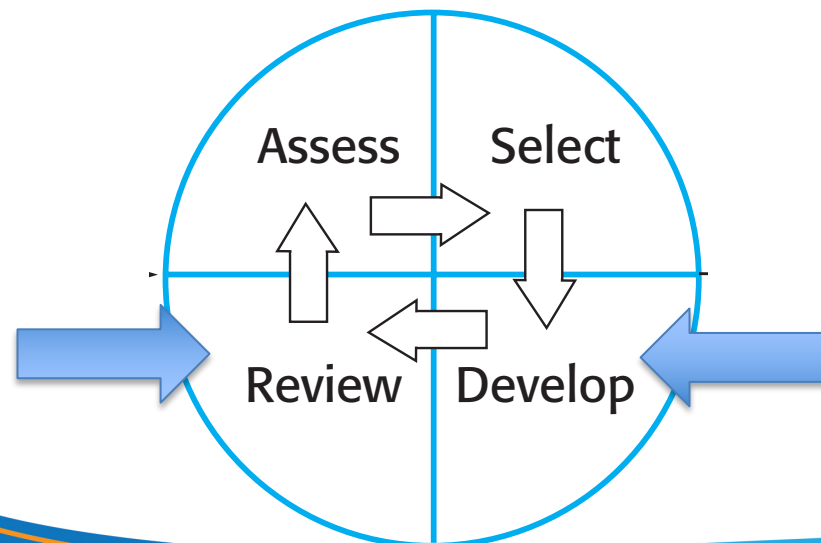
# Chu trình Sprint [1]

- Độ dài Sprint cố định, thường từ 2-4 tuần  $\Leftrightarrow$  sự phát triển của một bản release của hệ thống.
- Điểm bắt đầu cho kế hoạch là product backlog, là danh sách các công việc phải làm trong dự án.
- Pha chọn: cả nhóm phát triển dự án làm việc với khách hàng để chọn ra các đặc tính và chức năng được cài đặt trong sprint.

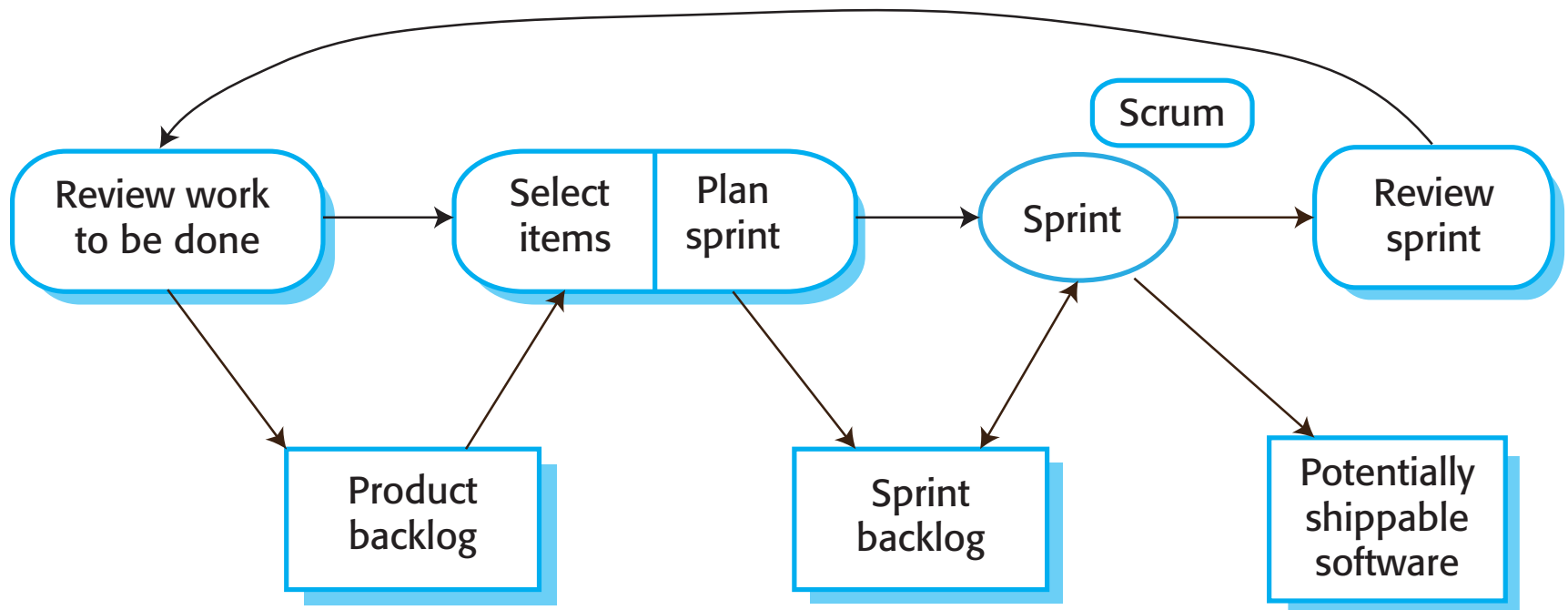


# Chu trình Sprint [2]

- Pha phát triển: Một khi các chức năng được lựa chọn, nhóm tự tổ chức để phát triển phần mềm.
  - ▣ Người khách hàng bị tách ra khỏi nhóm và tất cả các liên lạc đều thông qua '**Scrum master**'.
  - ▣ Vai trò của 'Scrum master': bảo vệ nhóm phát triển khỏi sự phân tán bên ngoài.
- Pha duyệt: công việc đã thực hiện được duyệt lại và chuyển cho stakeholder. Chu trình sprint tiếp theo lại bắt đầu.



# Scrum sprint cycle





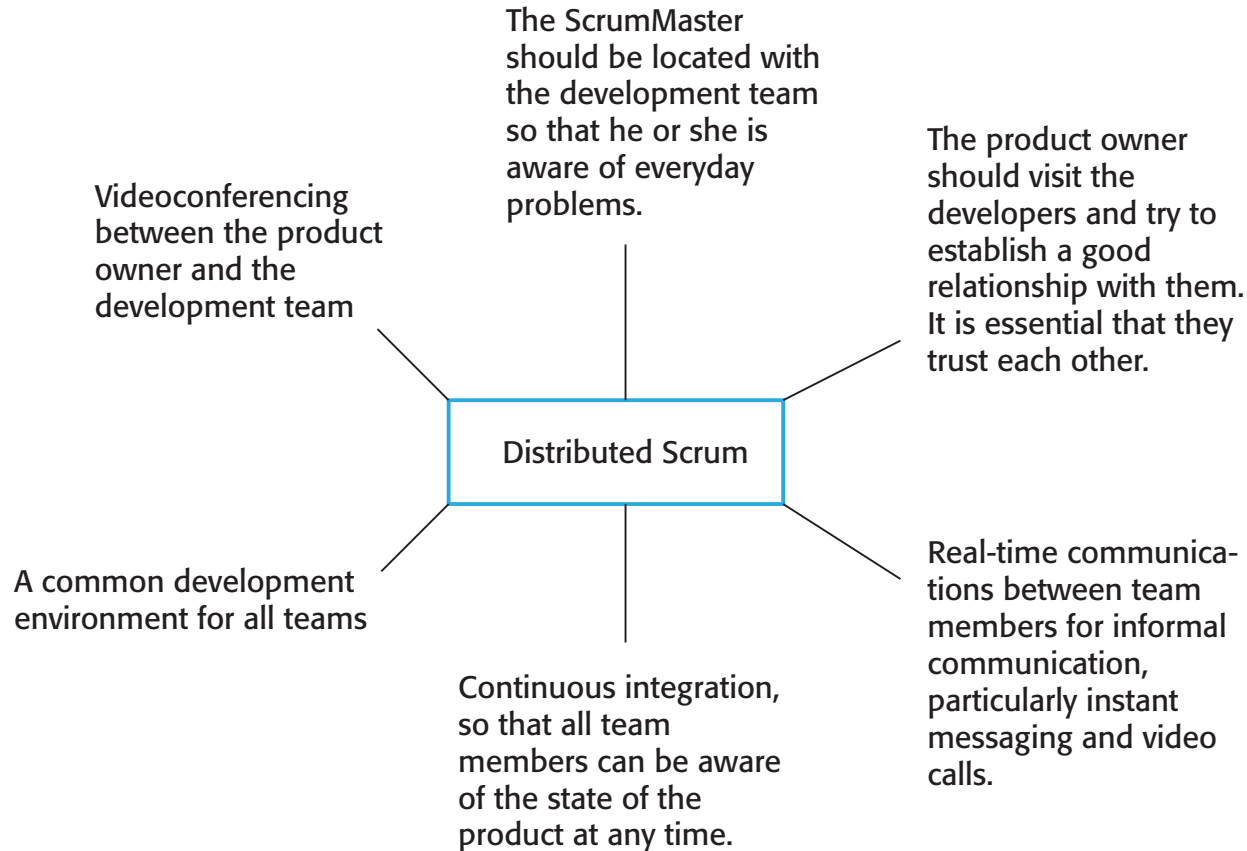
# Nhóm làm việc trong Scrum

- 'Scrum master' là người tổ chức họp hàng ngày, theo dõi tiến độ công việc, giao tiếp với khách hàng và quản lý bên ngoài nhóm.
- Toàn đội tham dự một cuộc họp ngắn hàng ngày
  - ▣ Các thành viên chia sẻ thông tin, mô tả tiến độ của họ, các vấn đề phát sinh và lên kế hoạch cho ngày tiếp theo.
  - ▣ Mọi người đều biết cái gì đang diễn ra, vấn đề phát sinh và lên kế hoạch ngắn hạn để đáp ứng sự thay đổi.

# Lợi ích của Scrum

- ☐ Sản phẩm được chia thành các phần nhỏ dễ hiểu và dễ quản lý.
- ☐ Các yêu cầu không ổn định sẽ không làm chậm trễ tiến độ.
- ☐ Toàn đội thấy được mọi thứ → giao tiếp nhóm được cải thiện.
- ☐ Khách hàng nhận từng phần đúng hạn và gửi phản hồi về sản phẩm.
- ☐ Niềm tin giữa khách hàng và đội phát triển tăng lên và một văn hóa tích cực được tạo ra trong đó mỗi người đều mong muốn dự án thành công.

# Distributed Scrum



# Nội dung

1. Các phương pháp linh hoạt
2. Phát triển hoạch định sẵn và linh hoạt
3. Extreme programming
4. Quản trị dự án linh hoạt
5. Mở rộng quy mô của phương pháp linh hoạt

# Mở rộng quy mô

- Các phương pháp linh hoạt được chứng minh là một phương pháp thành công
  - cho các dự án vừa và nhỏ,
  - được phát triển bởi nhóm nhỏ làm việc tại cùng một nơi.
  - Thành công do việc giao tiếp giữa các thành viên được cải thiện khi mọi người làm việc cùng nhau.
- Việc mở rộng quy mô của phương pháp linh hoạt gồm
  - điều chỉnh phương pháp này để phù hợp với những dự án lớn hơn,
  - các tổ chức lớn sử dụng phương pháp này để phát triển ứng dụng.

# Scaling out và scaling up

- 'Scaling up'
  - ▣ sử dụng phương pháp linh hoạt để phát triển các hệ thống lớn.
- 'Scaling out':
  - ▣ cách các phương pháp linh hoạt được giới thiệu đến các tổ chức đã có nhiều năm kinh nghiệm về sản xuất phần mềm.
- Khi mở rộng quy mô các phương pháp linh hoạt cần giữ lại các nền tảng linh hoạt
  - ▣ Kế hoạch linh động,
  - ▣ các bản release ra thường xuyên,
  - ▣ tích hợp liên tục,
  - ▣ phát triển theo hướng kiểm thử và
  - ▣ giao tiếp nhóm tốt.

# Các vấn đề gặp phải

- Tính phi hình thức của việc phát triển linh hoạt không tương thích với cách tiếp cận pháp lý liên quan đến việc định nghĩa hợp đồng, được sử dụng phổ biến trong các công ty lớn.
- Các phương pháp linh hoạt phfu hợp nhất với việc phát triển phần mềm mới hơn là bảo trì phần mềm. Đa phần chi phí phần mềm trong các công ty lớn đến từ việc bảo trì hệ thống phần mềm có sẵn.
- Các phương pháp linh hoạt được thiết kế cho các team làm việc cùng nhau, vẫn chưa thật sự phù hợp với các team phân tán toàn cầu.

# Vấn đề liên quan đến hợp đồng

- ☐ Hầu hết các hợp đồng phần mềm cho các hệ thống đặt hàng dựa vào đặc tả, thiết lập ra cái gì phải cài đặt bởi người phát triển cho khách hàng hệ thống.
- ☐ Tuy nhiên, điều này hạn chế việc đan xen phát triển và đặc tả, vốn là chuẩn của phát triển linh hoạt.
- ☐ Một hợp đồng trả cho người phát triển dựa trên thời gian hơn là dựa vào chức năng được yêu cầu.
  - ☐ Tuy nhiên, đây được nhìn nhận là rủi ro lớn về mặt pháp lý do những gì phải phân phối cho khách hàng không được đảm bảo.





# Pp linh hoạt và bảo trì phần mềm

- Hầu hết các tổ chức dành nhiều thời gian để bảo trì hệ thống đang có hơn là phát triển mới hoàn toàn → nếu phương pháp linh hoạt thành công, họ phải hỗ trợ việc bảo trì như việc phát triển bản gốc.
- Hai câu hỏi chính cần được xem xét:
  - ▣ Các hệ thống được phát triển bằng phương pháp linh hoạt có bảo trì được không, nhấn mạnh rằng trong quy trình phát triển ta giảm thiểu tài liệu mang tính hình thức?
  - ▣ Các phương pháp linh hoạt có hiệu quả cho việc cải tiến một hệ thống để trả lời các yêu cầu thay đổi của khách hàng không?
- Các vấn đề có thể nảy sinh nếu nhóm phát triển ban đầu không được duy trì.

# Bảo trì phần mềm

- Các vấn đề chính:
  - Thiếu tài liệu liên quan đến sản phẩm
  - Đảm bảo được việc tham gia của khách hàng vào quy trình phát triển
  - Duy trì tính liên tục của nhóm phát triển
- Việc phát triển linh hoạt dựa vào việc hiểu biết của nhóm phát triển và hiểu được cái gì cần phải làm.
- Với những hệ thống được sử dụng lâu dài, đây là vấn đề thực sự do nhóm phát triển ban đầu không còn làm việc trên hệ thống.

# Linh hoạt vs hoạch định sẵn

- Đa số dự án chứa các phần tử của của quy trình phát triển linh hoạt và hoạch định sẵn. Việc cân bằng dựa vào:
  1. Có cần đặc tả và thiết kế chi tiết trước khi chuyển sang cài đặt hay không?
    - Nếu có, ta cần sử dụng phương pháp hoạch định sẵn.
  2. Chiến lược chuyển giao dần dần có thực tế không?
    - Nếu có, xem xét việc sử dụng phương pháp linh hoạt.
  3. Hệ thống sẽ phát triển lớn đến mức nào?
    - Phương pháp linh hoạt hiệu quả nhất khi hệ thống được phát triển với một đội ngũ nhỏ làm việc cùng một nơi và giao tiếp thân mật với nhau.
    - Hệ thống lớn đòi hỏi đội ngũ phát triển lớn hơn → có thể sử dụng phương pháp hoạch định sẵn.

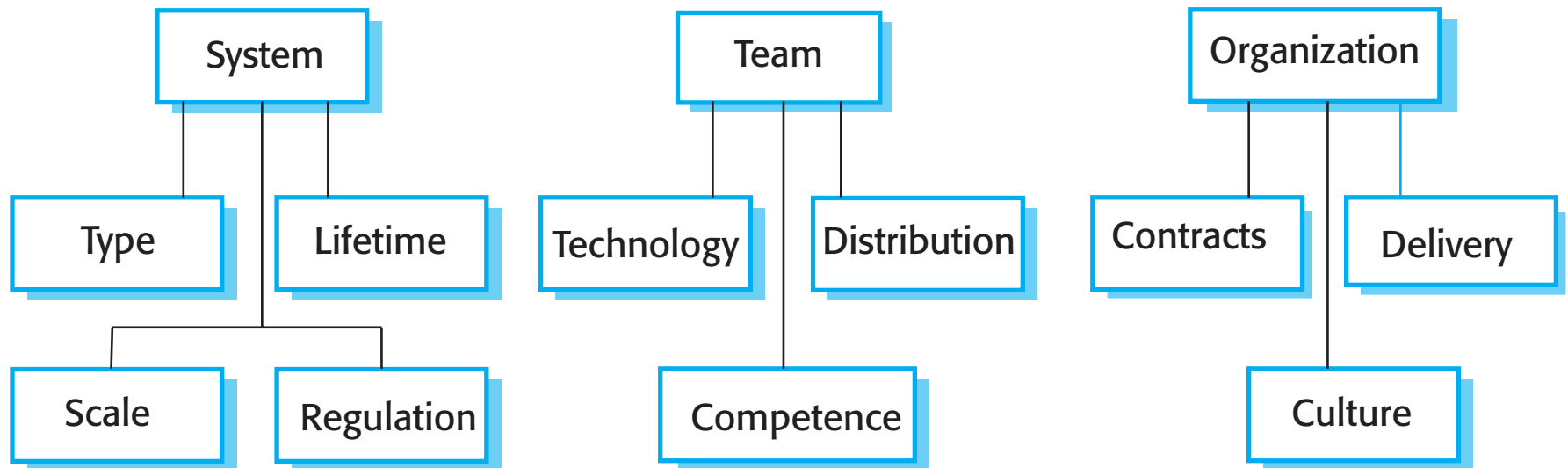
# Các nguyên lý của agile và practice về mặt tổ chức

Principle	Practice
Customer involvement	<p>This depends on having a customer who is willing and able to spend time with the development team and who can represent all system stakeholders. Often, customer representatives have other demands on their time and cannot play a full part in the software development.</p> <p>Where there are external stakeholders, such as regulators, it is difficult to represent their views to the agile team.</p>
Embrace change	<p>Prioritizing changes can be extremely difficult, especially in systems for which there are many stakeholders. Typically, each stakeholder gives different priorities to different changes.</p>
Incremental delivery	<p>Rapid iterations and short-term planning for development does not always fit in with the longer-term planning cycles of business planning and marketing. Marketing managers may need to know what product features several months in advance to prepare an effective marketing campaign.</p>

# Các nguyên lý của agile và practice về mặt tổ chức

Principle	Practice
Maintain simplicity	Under pressure from delivery schedules, team members may not have time to carry out desirable system simplifications.
People not process	Individual team members may not have suitable personalities for the intense involvement that is typical of agile methods, and therefore may not interact well with other team members.

# Các yếu tố liên quan đến linh hoạt và hoạch định sẵn



# Vấn đề về hệ thống

- Hệ thống được phát triển lớn tới mức nào?
  - ▣ Phương pháp linh hoạt hiệu quả nhất khi team khá nhỏ, làm việc cùng nhau, và giao tiếp phi hình thức
- Loại hệ thống nào được phát triển?
  - ▣ Phương pháp hoạch định sẵn thích hợp với các hệ thống đòi hỏi một lượng phân tích lớn trước khi cài đặt
- Thời gian sử dụng hệ thống?
  - ▣ Các hệ thống được mong đợi sử dụng càng lâu → cần nhiều tài liệu thiết kế để truyền đạt ý định ban đầu của người phát triển hệ thống cho nhóm hỗ trợ.
- Hệ thống có chịu sự chi phối bởi các quy định từ bên ngoài không?
  - ▣ Nếu một hệ thống phải được duyệt bởi một nhân tố bên ngoài → cần tài liệu chi tiết.

# Con người và nhóm

- Trình độ của người thiết kế và người lập trình trong nhóm phát triển tốt đến đâu?
  - ▣ Phương pháp linh hoạt đòi hỏi kỹ năng cao hơn phương pháp hoạch định sẵn.
  - ▣ Trong phương pháp hoạch định sẵn: người lập trình chỉ việc chuyển các thiết kế chi tiết thành mã nguồn.
- Đội ngũ phát triển được tổ chức như thế nào?
  - ▣ Đội ngũ phát triển phân tán hoặc một phần của việc phát triển được gia công bên ngoài → cần phát triển các tài liệu thiết kế để giao tiếp giữa các nhóm với nhau.
- Công nghệ nào sẵn có để hỗ trợ sự phát triển hệ thống?
  - ▣ Phương pháp linh hoạt dựa vào các công cụ tốt để ghi lại dấu vết của việc thiết kế liên tục thay đổi.



# Vấn đề về tổ chức

- ☐ Các tổ chức công nghệ truyền thống có văn hoá của việc phát triển dựa vào kế hoạch, đây được xem là chuẩn công nghệ.
- ☐ Có phải practice về tổ chức chuẩn để phát triển đặc tả hệ thống ở mức chi tiết?
- ☐ Đại diện khách hàng có sẵn lòng cung cấp feedback cho các phần nhỏ của hệ thống không?
- ☐ Phát triển linh hoạt phi hình thức có phù hợp với văn hoá tổ chức gắn với tài liệu chi tiết không?



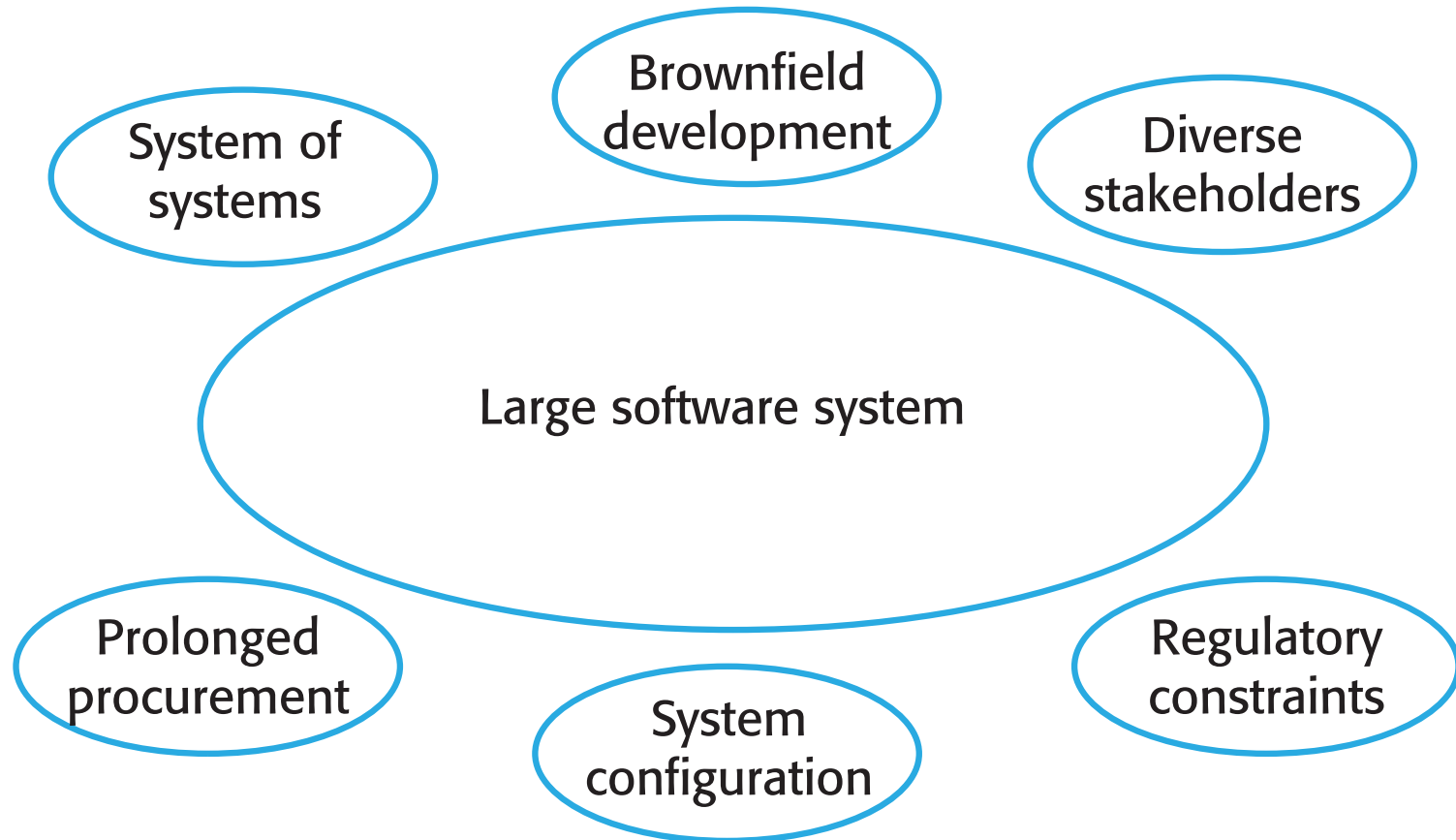
# Việc phát triển các hệ thống lớn[1]

- Các hệ thống lớn thường là tập hợp các hệ thống rời rạc, tương tác với nhau.
  - ▣ Có các nhóm phát triển riêng cho từng hệ thống con.
  - ▣ Các nhóm thường làm việc ở nhiều nơi khác nhau, nhiều múi giờ khác nhau.
- Các hệ thống này thường là ‘brownfield systems’
  - ▣ Gồm có một số các hệ thống con có sẵn và tương tác với các hệ thống này.
  - ▣ Nhiều yêu cầu hệ thống liên quan đến tương tác → không thật sự cần tính linh động và phát triển tăng dần.
- Vài hệ thống con được tích hợp để tạo ra một hệ thống lớn → liên quan nhiều đến cấu hình hệ thống hơn là phát triển mã nguồn mới.

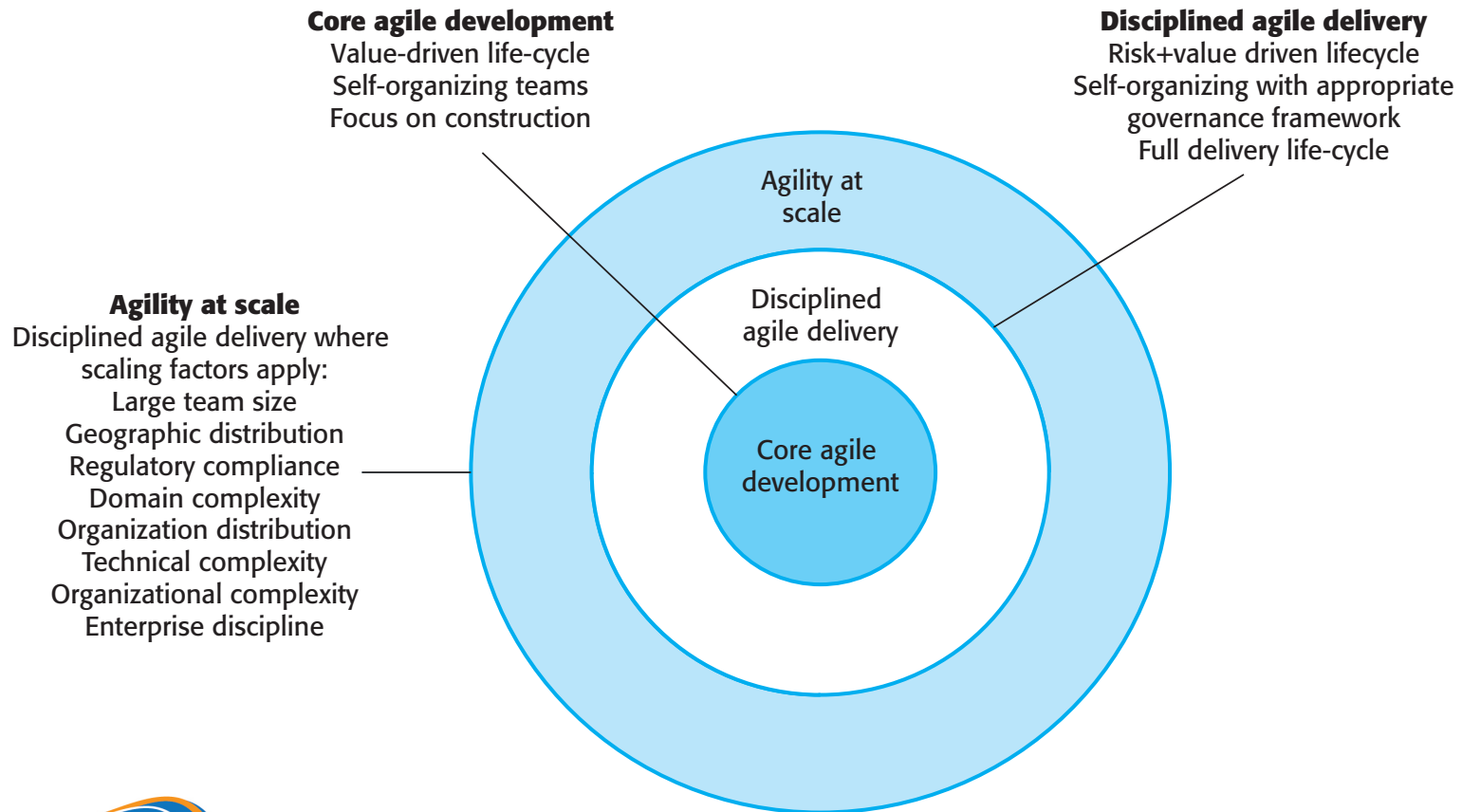
# Việc phát triển các hệ thống lớn[2]

- Các hệ thống lớn và quy trình phát triển của chúng thường ràng buộc bởi các quy tắc và quy định bên ngoài
  - ▣ giới hạn cách mà hệ thống được phát triển.
- Các hệ thống lớn có thời gian phát triển và sử dụng dài.
  - ▣ Khó duy trì các nhóm kết nối có các thành viên hiểu về hệ thống trong một thời gian dài.
- Các hệ thống lớn thường có một tập đa dạng các stakeholder.
  - ▣ Sẽ không thực tế nếu đưa tất cả các stakeholder này vào trong quy trình phát triển

# Các nhân tố trong hệ thống lớn



# IBM's agility at scale model



# Scaling up lên các hệ thống lớn

- ☐ Sử dụng phương pháp phát triển dần dần hoàn toàn cho việc lấy yêu cầu là không thể.
- ☐ Không thể chỉ có 1 product owner hay đại diện khách hàng.
- ☐ Không thể chỉ tập trung vào mã nguồn.
  - ☐ Ta cần thêm nhiều thiết kế và tài liệu hệ thống.
- ☐ Cơ chế giao tiếp giữa các nhóm phải được thiết kế và sử dụng.
- ☐ Tích hợp liên tục trên thực tế là không thể
  - ☐ Duy trì việc build hệ thống thường xuyên và tạo ra các bản release đều đặn là cần thiết.

# Multi-team Scrum

- Nhân rộng vai trò
  - ▣ Mỗi nhóm có một Product Owner và một Scrum Master để phát triển component của họ.
- Kiến trúc sư sản phẩm
  - ▣ Mỗi nhóm chọn một kiến trúc sư sản phẩm và những người này sẽ cộng tác để thiết kế và phát triển toàn bộ kiến trúc hệ thống.
- Release alignment
  - ▣ Ngày release sản phẩm từ từng nhóm phải khớp với nhau sao cho có thể tạo ra được một hệ thống hoàn chỉnh và có thể demo được.
- Scrum of Scrums
  - ▣ Tổ chức Scrum of Scrums hàng ngày tại đó các đại diện của mỗi nhóm sẽ gặp nhau để thảo luận về tiến độ và kế hoạch các công việc cần phải làm.

# Scaling out cho các công ty lớn

- ☐ Người quản trị dự án không có kinh nghiệm với phương pháp linh hoạt có thể sẽ miễn cưỡng chấp nhận phương pháp mới.
- ☐ Các tổ chức lớn thường có các thủ tục và chuẩn về chất lượng cần phải tuân theo.
  - ☐ Thường khó tương thích với các phương pháp linh hoạt.
- ☐ Các phương pháp linh hoạt được xem là hiệu quả khi các thành viên có kỹ năng tương đối cao.
  - ☐ Trong các tổ chức lớn, kỹ năng và khả năng thường không đồng đều.
- ☐ Văn hóa của công ty có thể là một trở ngại cho việc sử dụng phương pháp linh hoạt



# Câu hỏi?



KHOA CÔNG NGHỆ THÔNG TIN  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN