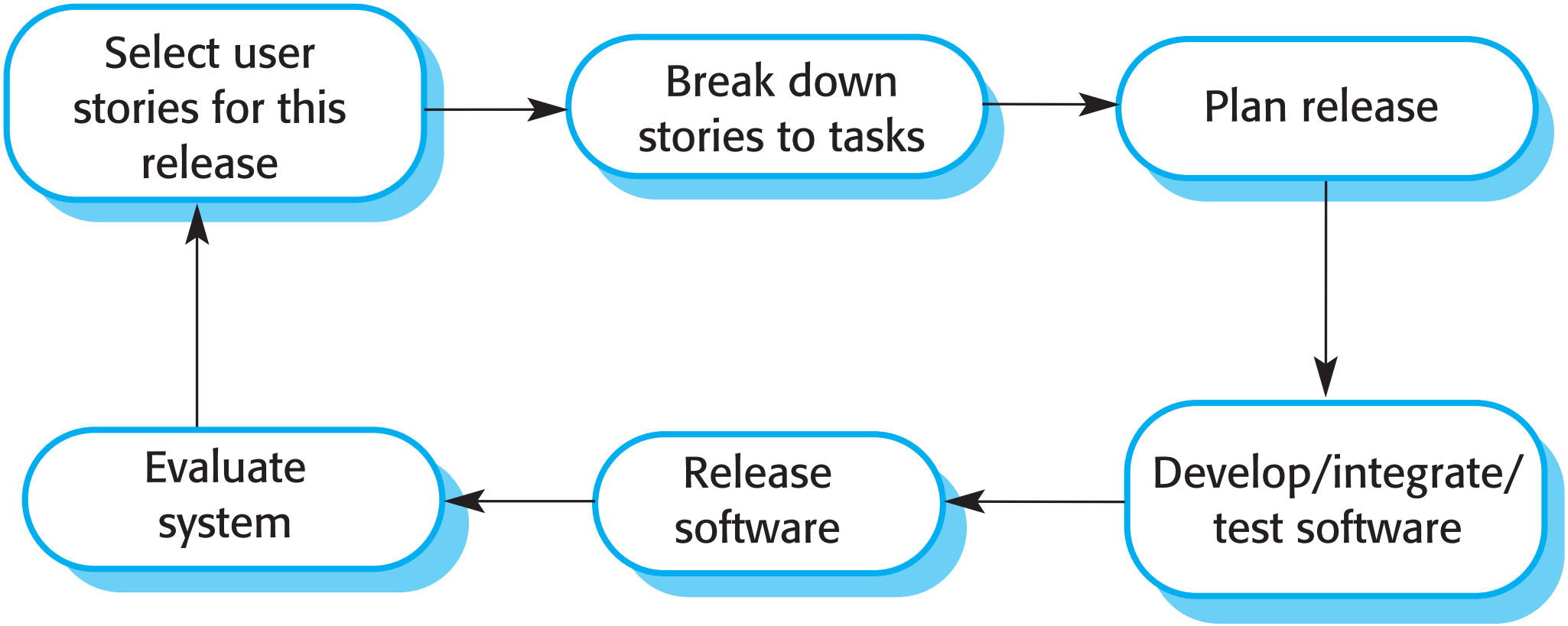
2. Kỹ thuật phát triển với Agile

2.1 Lập trình cực đoan (Extreme programming)

- Đây là phương pháp phát triển Agile rất có ảnh hưởng trong cuối những năm 90, trong đó một loạt các kỹ thuật được giới thiệu.
- Extreme Programming (XP) có một cách tiếp cận 'cực đoan' (extreme) để lặp đi lặp lại quá trình phát triển.
  - Các phiên bản mới có thể được xây dựng nhiều lần mỗi ngày;
  - Phần gia tăng được giao cho khách hàng 2 tuần một lần;
  - Tất cả các thử nghiệm (test) phải được chạy cho mọi bản dựng (build) và bản dựng chỉ được chấp nhận nếu các thử nghiệm chạy thành công.

2.1.1 Chu kỳ phát hành của lập trình cực đoan



2.1.2 Thực hành lập trình cực đoan (a)

Nguyên tắc hoặc thực hành	Mô tả
Lập kế hoạch gia tăng (Incremental planning)	Các yêu cầu được ghi trên các thẻ câu chuyện 'story card' và các câu chuyện được đưa vào các bản phát hành được xác định theo thời gian định sẵn và mức độ ưu tiên tương đối. Người phát triển sẽ chia những câu chuyện này thành Nhiệm vụ (Task) phát triển. Xem hình 3.5 và 3.6 trong sách.
Bản phát hành nhỏ (Small releases)	Bộ chức năng hữu dụng tối thiểu để cung cấp giá trị kinh doanh được phát triển trước tiên. Các bản phát hành của hệ thống một cách thường xuyên và dần dần bổ sung chức năng cho bản phát hành đầu tiên.
Thiết kế đơn giản (Simple design)	Đủ thiết kế để đáp ứng nhu cầu hiện tại và không cần hơn.
Phát triển thử nghiệm đầu tiên (Test-first development)	Framework kiểm tra đơn vị tự động (automated unit test) được sử dụng để viết các bài kiểm tra cho một phần chức năng mới trước khi chính chức năng đó được triển khai.
Tái cấu trúc (Refactoring)	Tất cả đội phát triển phải dự kiến sẽ cấu trúc lại mã code ngay khi tìm thấy các cải tiến có thể. Điều này giữ cho mã code đơn giản và dễ bảo trì.

2.1.2 Thực hành lập trình cực đoan (b)

Lập trình cặp (Pair programming)	Các thành viên trong đội phát triển làm việc theo cặp, để kiểm tra công việc của nhau và cung cấp hỗ trợ để công việc luôn tốt.
Sở hữu tập thể (Collective ownership)	Các cặp đôi phát triển làm việc trên tất cả các phần của hệ thống, do đó không có phần tách biệt phát triển và tất cả đội phát triển chịu trách nhiệm về mã code. Bất cứ ai cũng có thể thay đổi bất cứ điều gì ở mã code.
Tích hợp liên tục (Continuous integration)	Ngay sau khi một nhiệm vụ (task) được hoàn thành, nó sẽ được tích hợp vào toàn bộ hệ thống. Sau bất kì tích hợp nào như vậy, tất cả các kiểm tra đơn vị (unit test) trong hệ thống phải vượt qua.
Tiến bước vững vàng (Sustainable pace)	Số lượng làm ngoài giờ (overtime) quá lớn sẽ không được chấp nhận, vì nó (net effect) thường làm giảm chất lượng mã code và năng suất trung hạn (medium term productivity).
Khách hàng tại chỗ (On-site customer)	Một đại diện của người dung cuối của hệ thống (khách hàng) phải có mặt toàn thời gian để sử dụng nhóm XP. Trong quy trình lập trình cực đoan, khách hang là thành viên của nhóm phát triển và có trách nhiệm đưa các yêu cầu hệ thống đế nhóm thực hiện.

2.2 XP và các nguyên tắc Agile

- Phát triển gia tăng được hỗ trợ thông qua các bản phát hành nhỏ và thường xuyên.
- Khách hàng tham gia toàn thời gian với nhóm.
- Cả đội không giải quyết vấn đề theo cặp, đây là sở hữu tập thể, và quy trình cần phải tránh thời gian làm việc kéo dài.
- Thay đổi được hỗ trợ thông qua các bản phát hành hệ thống thông thường.
- Duy trì sự đơn giản thông qua việc cấu trúc lại mã liên tục.

2.3 Ứng dụng phương pháp XP

- Lập trình cực đoan (Extreme programming) tập trung vào kỹ thuật và không dễ dàng tích hợp với thực tế quản lý trong hầu hết các tổ chức, công ty.
- Do đó, trong khi phát triển linh hoạt (agile development) sử dụng cách từ XP, thì các phương pháp nguyên thủy ban đầu không được sử dụng rộng rãi.
- Các thực hành chính:
  - Câu chuyện người dùng (User stories) cho đặc tả.
  - Tái cấu trúc mã code (Refactoring).
  - Phát triển bản thử nghiệm đầu tiên (Test-first development).
  - Lập trình theo cặp (Pair programming).

2.3.1 User story

- Trong XP, khách hàng hoặc người dùng là một phần của nhóm XP và chịu trách nhiệm đưa ra quyết định về các yêu cầu.
- Yêu cầu của người dùng được thể hiện dưới dạng câu chuyện (user story) hoặc kịch bản (scenario).
- Chúng được viết lên các thẻ (card) và nhóm phát triển chia nhỏ chúng thành các nhiệm vụ (task) để làm. Các nhiệm vụ (task) này là cơ sở của tiến độ dự án và dự toán chi phí.
- Khách hàng chọn các câu chuyện để đưa vào bản phát hành tiếp theo dựa trên mức độ ưu tiên (priority) và ước lượng thời gian (schedule estimate).

2.3.1.1 Câu chuyện ‘Kê đơn thuốc’ (‘prescribing medication’ story)

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.

2.3.1.2 Task card

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.

2.3.2 Tái cấu trúc (Refactoring)

- Sự khôn ngoan thông thường trong kỹ thuật phần mềm là ***thiết kế để thay đổi***. Bạn dành thời gian và nỗ lực để lường trước những thay đổi vì điều này làm giảm chi phí sau này.
- Tuy nhiên, XP khẳng định điều này không đáng giá vì những thay đổi không thể lường trước ở mức đáng tin cậy.
- Thay vào đó, XP đề xuất cải tiến mã code liên tục (tái cấu trúc) để thực hiện các thay đổi dễ dàng hơn khi chúng bắt buộc phải thực hiện.

2.3.2 Tái cấu trúc (Refactoring)

- Nhóm lập trình tìm kiếm những cải tiến phần mềm có thể có và thực hiện những cải tiến này ngay cả khi không cần chúng ngay lập tức.
- Điều này cải thiện khả năng hiểu của phần mềm và do đó giảm nhu cầu làm tài liệu.
- Thay đổi dễ thực hiện hơn vì mã có cấu trúc tốt và rõ ràng.
- Tuy nhiên, một số thay đổi yêu cầu cấu trúc lại kiến trúc và điều này gây nên tốn kém hơn nhiều.

2.3.2.1 Ví dụ

- Tổ chức lại hệ thống phân cấp lớp (class hierarchy) để loại bỏ mã trùng lắp.
- Thu dọn, đổi tên các thuộc tính và phương thức để dễ hiểu hơn.
- Thay thế mã nội tuyến bằng các lệnh gọi đến các phương thức đã đưa vào thư viện chương trình.

2.3.3 Test-first development

- Kiểm tra (Testing) là trọng tâm của (XP). XP đã phát triển một cách tiếp cận mà chương trình được kiểm tra sau mỗi lần thay đổi được thực hiện.
- Các tính năng thử nghiệm XP:
  - Phát triển thử nghiệm đầu tiên.
  - Phát triển thử nghiệm gia tăng từ các kịch bản.
  - Người dùng tham gia vào việc phát triển và xác nhận thử nghiệm.
  - Kiểm thử tự động được sử dụng để chạy tất cả các kiểm tra thành phần mỗi khi bản phát hành mới được xây dựng.

2.3.3.1 Test-driven development

- Viết các bài kiểm tra trước khi viết mã làm rõ các yêu cầu cần thực hiện.
- Các bài kiểm tra được viết dưới dạng chương trình chứ không phải dữ liệu để chúng có thể được thực thi tự động. Kiểm tra bao gồm thử xem chương trình đã thực thi chính xác chưa.
  - Thường dựa trên các framework thử nghiệm như Junit.
- Tất cả các kiểm tra đã có và thêm bài kiểm tra phần mới được chạy tự động khi chức năng mới thêm vào, do đó, kiểm tra được chức năng mới không gây lỗi.

2.3.3.2 Khách hàng tham gia

- Vai trò của khách hàng trong quá trình thử nghiệm là giúp phát triển các thử nghiệm chấp nhận (acceptance testing) cho các câu chuyện sẽ được thực hiện trong bản phát hành tiếp theo của hệ thống.
- Khách hàng là một phần của nhóm viết các bài kiểm tra khi quá trình phát triển diễn ra. Do đó, tất cả các mã code mới đều được xác thực để đảm bảo rằng đó là những gì khách hàng cần.
- Tuy nhiên, những người đóng vai trò khách hàng có thời gian hạn chế và do đó không thể làm việc toàn thời gian với nhóm phát triển. Họ có thể nghĩ rằng cung cấp các yêu cầu là đủ rồi và miễn cưỡng tham gia vào thử nghiệm mà thôi.

2.3.3.4 Ví dụ Test-case

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.

2.3.3.5 Tự động hóa kiểm thử

- Tự động hóa kiểm thử (Test automation) có nghĩa là các bài kiểm tra được viết dưới dạng các thành phần có thể thực thi trước khi nhiệm vụ (task) được thực hiện.
  - Các thành phần kiểm tra này nên độc lập, phải mô phỏng việc gửi đầu vào để kiểm tra và phải kiểm tra xem kết quả có đáp ứng đặc tả ở đầu ra không. Framework kiểm thử tự động (như Junit) là một hệ thống giúp dễ dàng viết các bài kiểm tra thực thi và gửi một tập hợp các bài kiểm tra để thực thi.
- Vì thử nghiệm được tự động hóa, luôn có một tập hợp các thử nghiệm có thể được thực hiện nhanh chóng và dễ dàng.
  - Bất cứ khi nào chức năng mới được thêm vào hệ thống, các bài kiểm tra có thể được chạy và các vấn đề mà mã mới đã đưa vào có thể được khắc phục ngay lập tức.

2.3.3.6 Vấn đề?

- Các lập trình viên thích lập trình hơn là kiểm thử và đôi khi họ đi tắt đón đầu khi viết các bài kiểm tra.
  - Ví dụ, họ có thể viết các bài kiểm tra không đầy đủ không kiểm tra tất cả các trường hợp ngoại lệ có thể xảy ra..
- Một số bài kiểm tra có thể rất khó để viết tăng dần.
  - Ví dụ, trong một giao diện người dùng phức tạp, rất khó để viết các bài kiểm tra đơn vị cho mã code hiện thực 'logic hiển thị' và quy trình làm việc giữa các màn hình.
- Rất khó để đánh giá mức độ hoàn chỉnh của một bài kiểm tra. Mặc dù có thể có nhiều thử nghiệm hệ thống diễn ra, nhưng không đảm bảo tập hợp các thử nghiệm này đầy đủ.

2.3.4 Lập trình cặp đôi

- Lập trình cặp (Pair programming) là các lập trình viên làm việc theo cặp, phát triển mã code cùng nhau.
- Điều này giúp phát triển quyền sở hữu chung đối với mã code và hỗ trợ kiến thức trong nhóm.
- Nó giống như là một quá trình xem xét (review) không chính thức vì mỗi dòng mã code được nhiều hơn 1 người xem.
- Nó khuyến khích tái cấu trúc vì cả nhóm có thể hưởng lợi từ việc cải thiện mã hệ thống.

2.3.4 Lập trình cặp đôi

- Trong lập trình cặp, các lập trình viên ngồi cùng nhau trên cùng một máy tính để phát triển phần mềm.
- Các cặp được tạo ra một cách tự động để tất cả các thành viên trong nhóm làm việc với nhau trong quá trình phát triển.
- Việc chia sẻ kiến thức xảy ra trong quá trình lập trình theo cặp là rất quan trọng vì nó làm giảm rủi ro tổng thể cho một dự án khi các thành viên trong nhóm rời đi.
- Lập trình theo cặp không hẳn là không hiệu quả và có một số bằng chứng cho thấy rằng một cặp làm việc cùng nhau sẽ hiệu quả hơn 2 lập trình viên làm việc riêng lẻ.