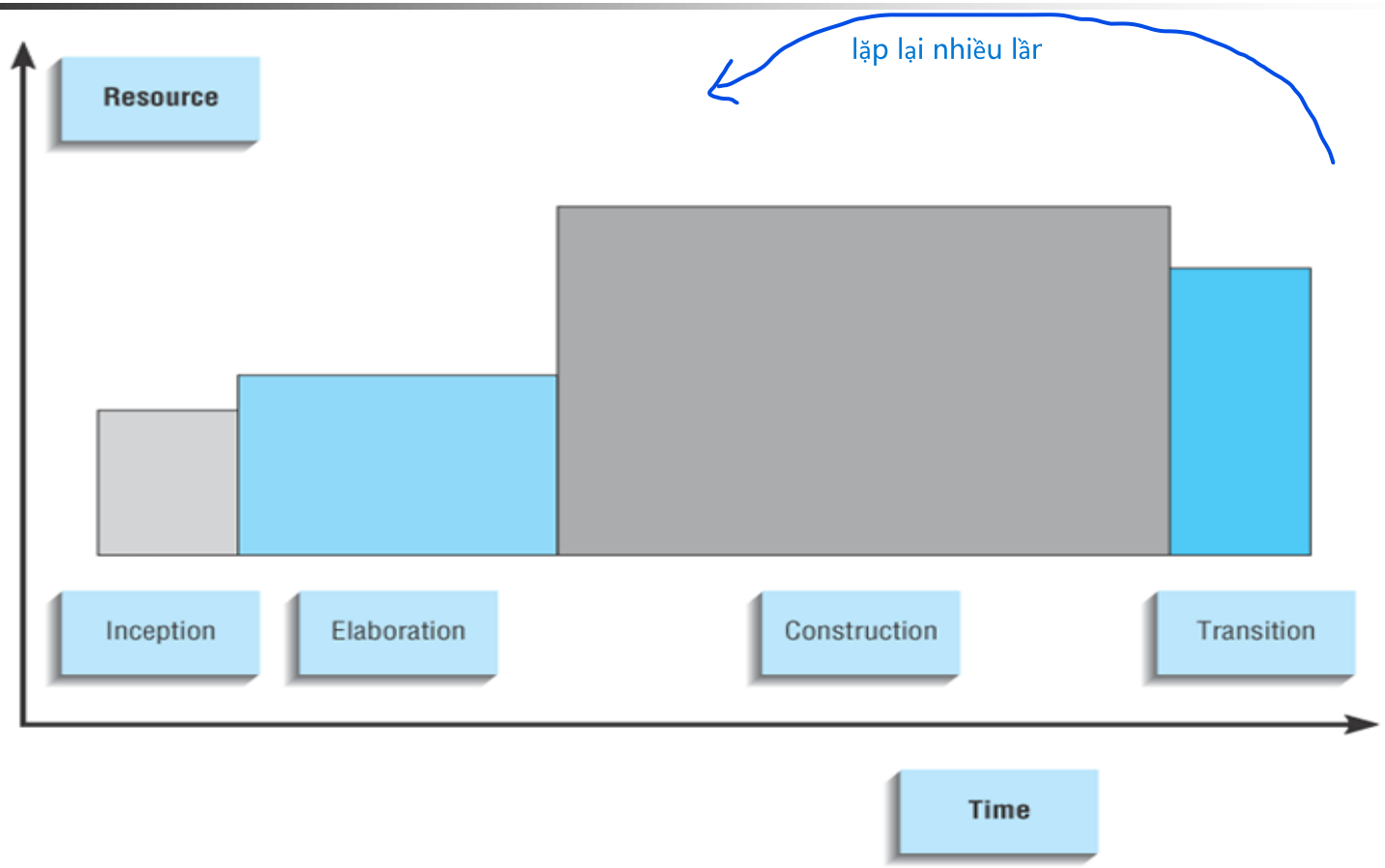
Chương 1

Tổng quan về Kiểm thử phần mềm

Nội dung

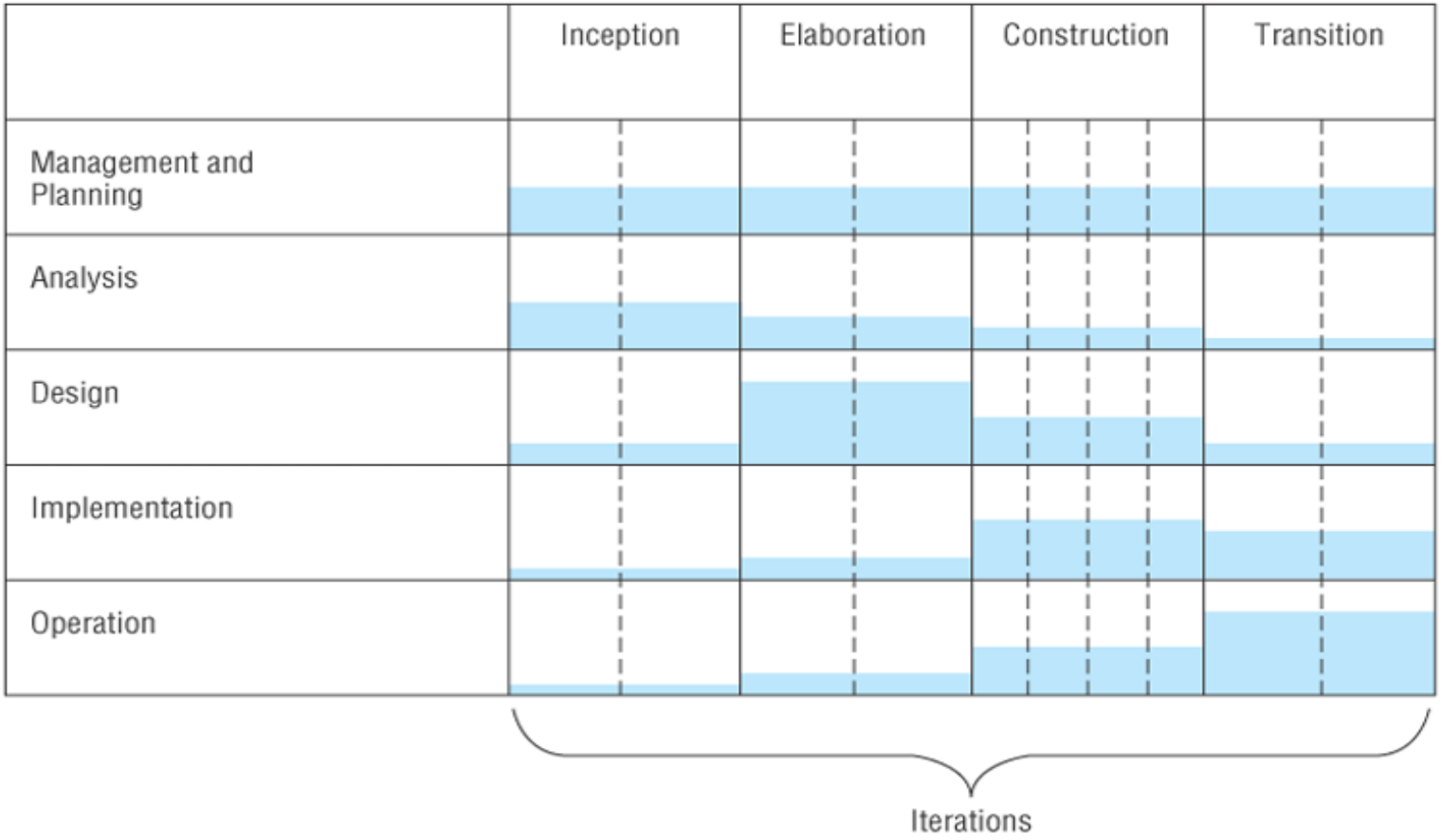
- Quy trình phát triển phần mềm *RUP* (*Rational Unified Process*)
- Kiểm thử phần mềm là gì?
- Các nguyên tắc kiểm thử phần mềm
- Các hạn chế của kiểm thử phần mềm
- Mô hình V

Quy trình phát triển phần mềm RUP



Hình 1.1. Các giai đoạn phát triển RUP (*Rational Unified Process*)

Quy trình phát triển phần mềm RUP

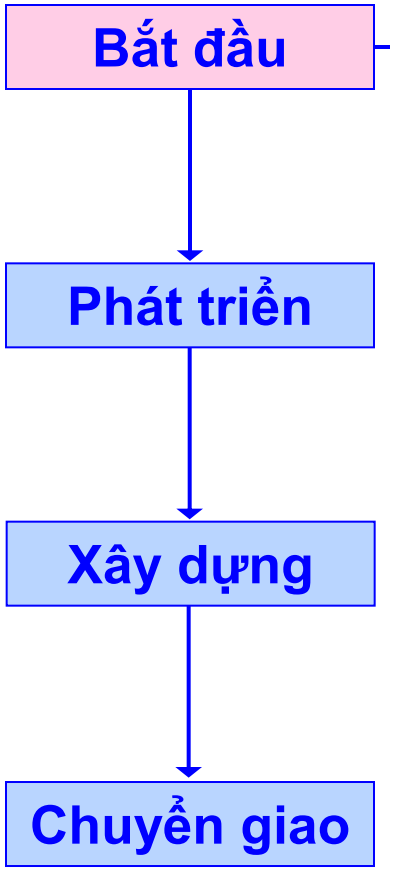


Hình 1.2. Phát triển lặp: các công việc của mỗi giai đoạn

Quy trình phát triển phần mềm RUP

- **Chu kỳ phần mềm (*software cycle*)**
  - ***Bắt đầu*: các yêu cầu của người sử dụng**
  - ***Kết thúc*: phần mềm đáp ứng đúng các yêu cầu của người sử dụng.**
- **Các giai đoạn của chu kỳ phần mềm**
  - **Bắt đầu (*inception*)**
  - **Phát triển (*elaboration*)**
  - **Xây dựng (*construction*)**
  - **Chuyển giao (*transition*)**
- **Mỗi giai đoạn thường được thực hiện lặp nhiều lần để kết quả ngày càng tốt hơn.**

Quy trình phát triển phần mềm RUP



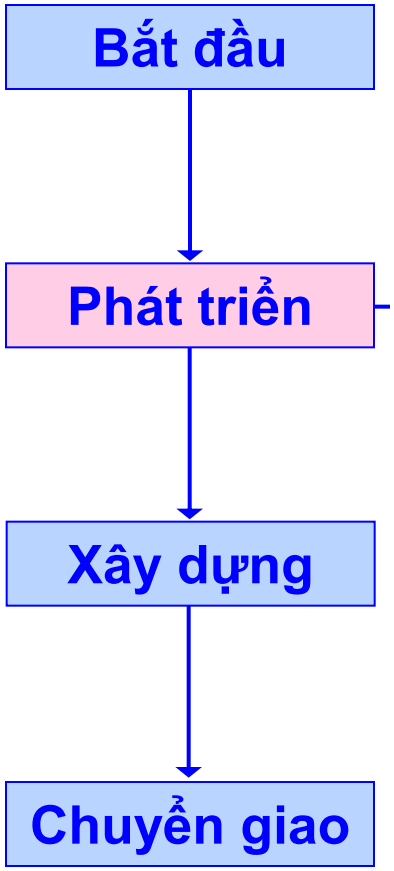
Giai đoạn 1. Bắt đầu (*inception*)

- ***Định nghĩa phạm vi*** **(*scope*).**
- ***Xác định tính khả thi*** **(*feasibility*).**
- ***Hiểu các yêu cầu của người sử dụng*** **(*user requirement*).**
- ***Chuẩn bị kế hoạch phát triển phần mềm*** **(*software development plan*).**
- **Các yêu cầu về tài nguyên tương đối ít.**
- **Thời gian của giai đoạn này ngắn.**
- **Tập trung vào *lập kế hoạch và phân tích*.**

thiên về business người dùn

Quy trình phát triển phần mềm RUP

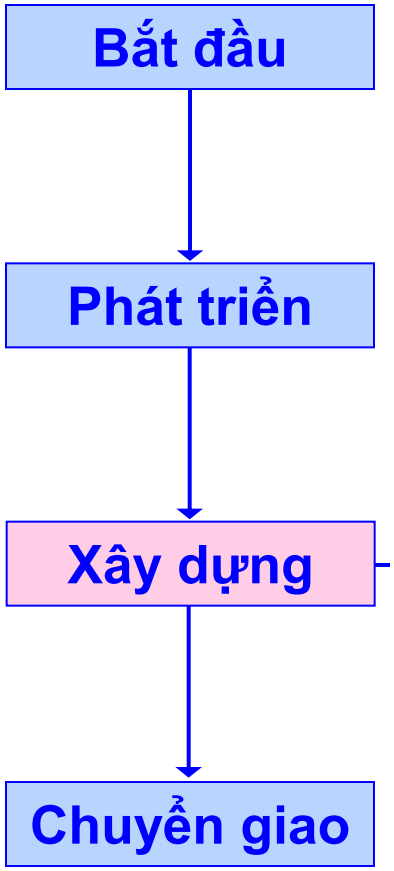
system design



Giai đoạn 2. Phát triển (*elaboration*)

- ***Chi tiết hóa các yêu cầu của người sử dụng*.**
- ***Thiết kế kiến trúc.***
- **Không cần nhiều tài nguyên.**
- **Thời gian của giai đoạn này tương đối dài.**
- **Tập trung vào *phân tích và thiết kế*.**

Quy trình phát triển phần mềm RUP

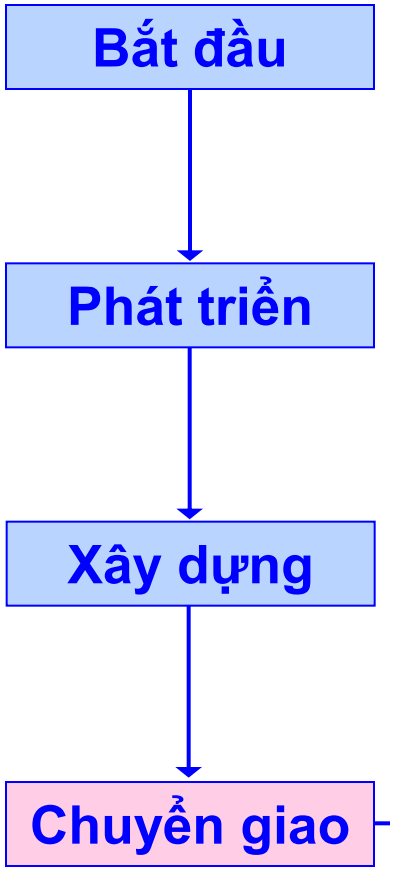


Giai đoạn 3. Xây dựng (*construction*)

- ***Lập trình*** **(*coding*).**
- ***Kiểm tra*** **(*testing*).**
- ***Lập tài liệu lập trình*****.**
- **Cần nhiều tài nguyên.**
- **Thời gian của giai đoạn này là dài nhất.**
- **Tập trung vào *thực hiện các công việc*.**

tập trung viết theo chuẩn s
Viết theo chuẩn C

Quy trình phát triển phần mềm RUP



Giai đoạn 4. Chuyển giao (*transition*)

- ***Triển khai hệ thống*** **(*system deployment*).**
- ***Đào tạo và hỗ trợ người sử dụng*.**
- **Cần nhiều tài nguyên.**
- **Thời gian của giai đoạn này ngắn.**
- **Tập trung vào *cài đặt, đào tạo và hỗ trợ*.**

Các vấn đề thường gặp

Các vấn đề thường gặp trong phát triển phần mềm

- *Tính toán* bị sai.
- *Hiệu chỉnh dữ liệu* bị sai.
- *Kết hợp dữ liệu* bị sai.
- *Tìm kiếm dữ liệu* bị sai so với yêu cầu.
- Không thỏa mãn các *ràng buộc của nghiệp vụ*.
- *Hiệu suất của phần mềm* còn thấp.
- *Kết quả* không đáng tin cậy.
- Chưa đáp ứng đầy đủ các *yêu cầu nghiệp vụ*.
- Chưa *giao tiếp* được với các hệ thống khác.
- *Bảo mật* chưa tốt.

sai bản chất nghiệp vụ

Kiểm thử phần mềm là gì?

- **Mục đích của kiểm thử phần mềm**
  - Phần mềm thực hiện đúng *các yêu cầu của người sử dụng*.
- **Kiểm thử phần mềm cho thấy:**
  - Phần mềm *thực hiện đúng các yêu cầu*.
  - *Phát hiện các khiếm khuyết* trước khi đưa vào sử dụng.
- **Kiểm thử phần mềm là:**
  - Quy trình chứng minh phần mềm *không có lỗi sai*.
  - Quy trình chạy phần mềm để *tìm kiếm các lỗi sai*.

Kiểm thử phần mềm là gì?

- **Kiểm tra các kết quả của lần chạy thử để:**
  - **Phát hiện lỗi sai**
  - **Phát hiện các bất thường**
  - **Có thông tin về các thuộc tính phi chức năng của chương trình.**

phát hiện các trường hợp khác không chạy được ngoài trường hợp đang làm

Mục tiêu của kiểm thử phần mềm

- Chứng minh cho người phát triển và khách hàng rằng phần mềm thoả mãn các yêu cầu.
  - **Đối với phần mềm của khách hàng,** điều này có nghĩa là phải có ít nhất một lần kiểm tra cho mọi yêu cầu trong tài liệu các yêu cầu.
  - **Đối với các sản phẩm phần mềm nói chung,** điều này có nghĩa là phải có nhiều lần kiểm tra cho tất cả các tính năng của hệ thống, các tổ hợp của các tính năng này mà chúng kết hợp chặt chẽ với nhau trong phiên bản sản phẩm.
- Phát hiện các trường hợp mà phần mềm chạy sai, không mong muốn hoặc không đúng theo đặc tả của nó.

Mục tiêu của kiểm thử phần mềm

- **Mục tiêu đầu tiên là kiểm tra kiểm chứng (*validation testing*)**
  - **Chứng minh cho người phát triển và khách hàng rằng phần mềm thoả mãn các yêu cầu của nó.**
  - **Mong muốn hệ thống thực hiện đúng bằng cách sử dụng một tập các *test-case* để phản ánh hệ thống thực hiện đúng như mong muốn.**
  - **Một kiểm tra thành công cho thấy hệ thống hoạt động như mong muốn.**

phát hiện khiếm khuyết khách hà

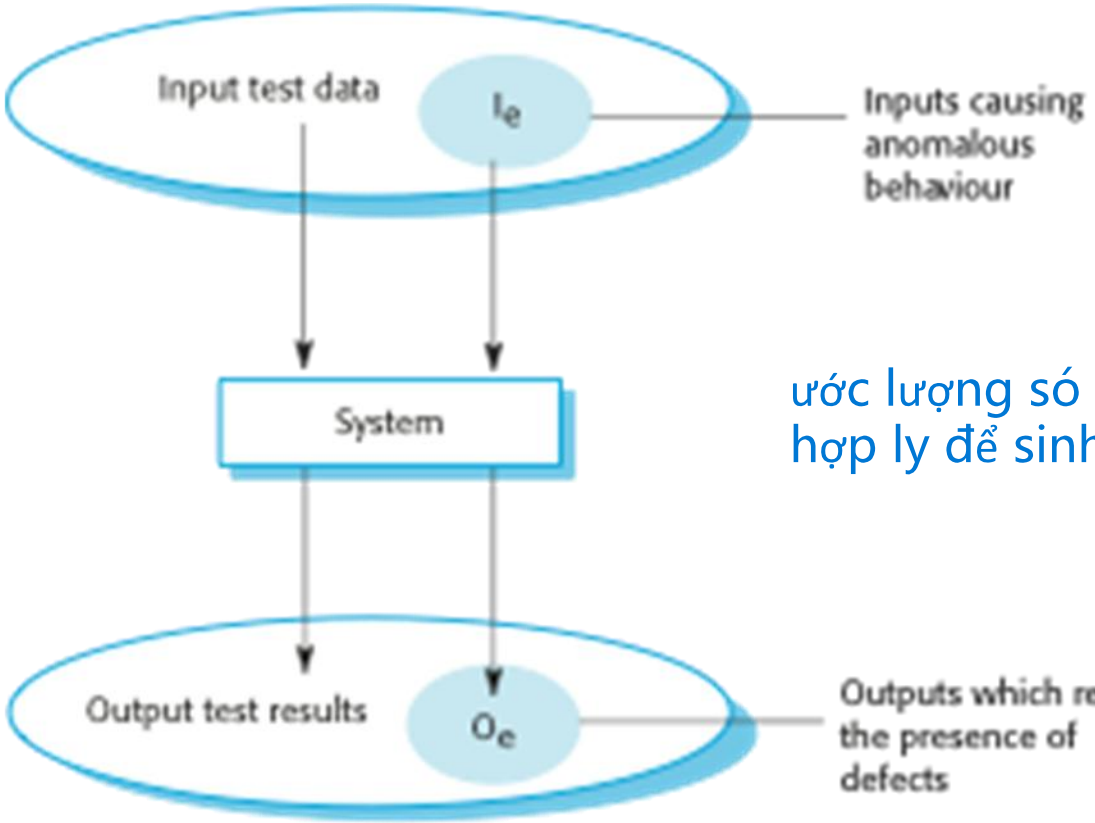
đưa các test case phủ hết toàn bộ cod

đưa ra các trường hợp biên để kiểm tra lỗi sai

Mục tiêu của kiểm thử phần mềm

- Mục tiêu thứ hai là kiểm tra khiếm khuyết (*defect testing*)
  - Phát hiện các lỗi sai hoặc các thiếu sót trong phần mềm mà cách hoạt động của nó bị sai hoặc không đúng với đặc tả của nó.
  - Thiết kế các *test-case* để phát hiện các khiếm khuyết. Các *test-case* có thể không rõ ràng và không cần phản ánh cách hệ thống thường được sử dụng.
  - Một kiểm tra thành công là kiểm tra làm cho hệ thống thực hiện sai và bộc lộ khiếm khuyết của hệ thống.
  - Trừ khử hoạt động không mong muốn: phần mềm không chạy, các tương tác không mong muốn với các hệ thống khác, tính toán bị sai, dữ liệu bị sai.

Mục tiêu của kiểm thử phần mềm



ước lượng số lượng test case hợp ly để sinh ra lỗi

Kiểm định (*Verification*)

chạy đúng

- **Phần mềm phải làm đúng với đặc tả của nó.**

  Are we building the product right?

- ***Kiểm định (xác minh)*** **là quy trình đánh giá phần mềm để xác định các sản phẩm của một giai đoạn phát triển cụ thể có thỏa mãn các điều kiện bắt buộc được xác định ở đầu giai đoạn này.**
- **Kiểm định là kiểm tra tĩnh (*static testing*)**
- **Các phương pháp kiểm định**
  - **Walkthrough**
  - **Code Inspection**

Kiểm chứng (*Validation*) dúng quy trình, đúng kết quả

- **Phần mềm phải làm điều mà khách hàng thật sự cần.**

  Are we building the right product?
- ***Kiểm chứng (xác thực)*** **là quy trình đánh giá phần mềm trong lúc hoặc cuối giai đoạn phát triển phần mềm để xác định phần mềm có thỏa mãn các yêu cầu quy định.**
- **Kiểm chứng là quy trình đánh giá sản phẩm cuối cùng để kiểm tra phần mềm thỏa mãn yêu cầu mong muốn của khách hàng.**
- **Kiểm chứng là kiểm tra động (*dynamic testing*)**
- **Các phương pháp kiểm định**
  - **Testing**
  - **End Users**

Kiểm định và Kiểm chứng

	Verification	Validation
1	**Verification** is a static practice of verifying documents, design, code and program.	**Validation** is a dynamic mechanism of validating and testing the actual product.
2	It does not involve executing the code.	It always involves executing the code.
3	It is human based checking of documents and files.	It is computer based execution of program.
4	**Verification** uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	**Validation** uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5	**Verification** is to check whether the software conforms to specifications.	**Validation** is to check whether software meets the customer expectations and requirements.
6	It can catch errors that validation cannot catch. It is low level exercise.	*It can catch errors that verification cannot catch. It is High Level Exercise.*
7	Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8	**Verification** is done by QA team to ensure that the software is as per the specifications in the SRS document.	***Validation*** *is carried out with the involvement of testing team.*
9	It generally comes first-done before validation.	It generally follows after **verification**.

Các nguyên tắc kiểm thử phần mềm

- **Trường hợp kiểm tra (*test-case*)**
  - **Mô tả**: đặc tả các điều kiện cần có để tiến hành kiểm tra.
  - **Nhập**: đặc tả đối tượng hay dữ liệu cần thiết, được sử dụng làm đầu vào để thực hiện việc kiểm tra.
  - **Kết quả mong muốn**: kết quả trả về từ đối tượng kiểm tra, chứng tỏ đối tượng đạt yêu cầu.
- **Thiết kế đầy đủ các *test-case***
  - Dữ liệu nhập **hợp lệ**
  - Dữ liệu nhập **không hợp lệ**

Các nguyên tắc kiểm thử phần mềm

- **Kiểm tra kết quả**
  - Kết quả mong muốn
  - Kết quả không mong muốn

Các nguyên tắc kiểm thử phần mềm

- Việc kiểm thử đòi hỏi tính độc lập: người lập trình nên tránh việc kiểm thử các chương trình của mình viết.
- Kiểm thử phần mềm nên bắt đầu từ các thành phần đơn giản, rồi đến các thành phần phức tạp hơn.
- Nên lập kế hoạch và qui trình kiểm thử trước khi bắt đầu kiểm thử.

Các hạn chế của kiểm thử phần mềm

- Các đặc tả phần mềm có thể chưa đúng.
- Công cụ kiểm thử có thể chưa chắc đúng.
- Không có công cụ kiểm thử nào thích hợp cho mọi phần mềm.
- Kỹ sư kiểm thử có thể chưa hiểu đầy đủ về sản phẩm phần mềm.
- Không thể thực hiện kiểm thử phần mềm một cách đầy đủ.

Thuật ngữ

- **Defect (khiếm khuyết)**
  - Sự khác biệt giữa kết quả mong muốn và kết quả thực tế trong ngữ cảnh kiểm thử.
  - Sự khác biệt giữa sản phẩm phần mềm (*software product*) với yêu cầu của khách hàng (*user requirement*).
  - Một lỗi sau khi ứng dụng đã được triển khai trong thực tế.
  - Đề cặp đến một số rắc rối (*trouble*) của sản phẩm phần mềm, với hành vi (*behavior*) bên ngoài hoặc với các tính năng (*feature*) bên trong của nó.

Thuật ngữ

- **Phân loại defect**
  - **Wrong**
    - Khi yêu cầu được thực hiện không đúng.
    - Khiếm khuyết này do không đúng với các đặc tả.
  - **Missing**
    - Yêu cầu của khách hàng đã không được thực hiện.
    - Khiếm khuyết này do không đúng với các đặc tả, dấu hiệu cho thấy đặc tả không được thực hiện, hoặc yêu cầu của khách hàng không được ghi chép đúng.

Thuật ngữ

- Phân loại defect
  - **Extra**
    - **Một yêu cầu được đưa vào sản phẩm phần mềm nhưng không phải là yêu cầu của khách hàng.**
    - **Là điều khác biệt so với đặc tả, nhưng có thể là một thuộc tính mong muốn cho khách hàng.**
    - **Là khiếm khuyết bởi vì nó khác với các yêu cầu đề ra.**

Thuật ngữ

- Phân loại defect
  - Error
    - **Một lỗi là một sai lầm (*mistake*), quan niệm sai, hiểu sai của người phát triển phần mềm (*developer*): kỹ sư phần mềm, người lập trình, người phân tích, người kiểm thử. Kết quả: thay đổi chức năng (*functionality*) của phần mềm.**
    - **Là trạng thái bên trong (*internal state*) bị sai, là biểu hiện của *fault*.**
    - ***Ví dụ*: hiểu sai chức năng của phần mềm, tính toán bị sai, viết sai tên biến,…**

Thuật ngữ

- Phân loại defect
  - Các loại Error
    - User interface error (*Ex*: output errors, incorrect user messages)
    - Function errors
    - Hardware defects
    - Incorrect program version
    - Requirements errors
    - Design errors
    - Documentation errors
    - Architecture errors
    - Module interface errors
    - Performance errors
    - Boundary-related errors

Thuật ngữ

- Phân loại defect
  - **Bug**
    - **Là kết quả của lỗi lập trình (*coding error*).**
    - **Một lỗi được tìm thấy trong môi trường phát triển trước khi giao sản phẩm phần mềm cho khách hàng.**
    - **Lỗi lập trình làm cho chương trình hoạt động kém, tạo ra kết quả sai, hoặc bị sập (*crash*).**
    - **Một lỗi trong phần mềm hoặc phần cứng làm cho chương trình bị trục trặc (*malfunction*).**
    - ***Bug* là thuật ngữ của người kiểm thử.**

Thuật ngữ

- Phân loại defect
  - Fault
    - Khiếm khuyết tĩnh (*static*) trong phần mềm.
    - Định nghĩa dữ liệu (*data definition*), quá trình xử lý (*process*), hoặc một bước (*step*) bị sai trong chương trình máy tính làm cho chương trình thực hiện theo cách không dự kiến trước.
    - Là sự bất thường (*anomoly*) trong phần mềm, có thể làm cho phần mềm hoạt động sai, không đúng với đặc tả.
    - *Fault* là lỗi thiết kế (*design mistake*).
    - *Fault* là trạng thái của phần mềm được gây ra bởi *error*.

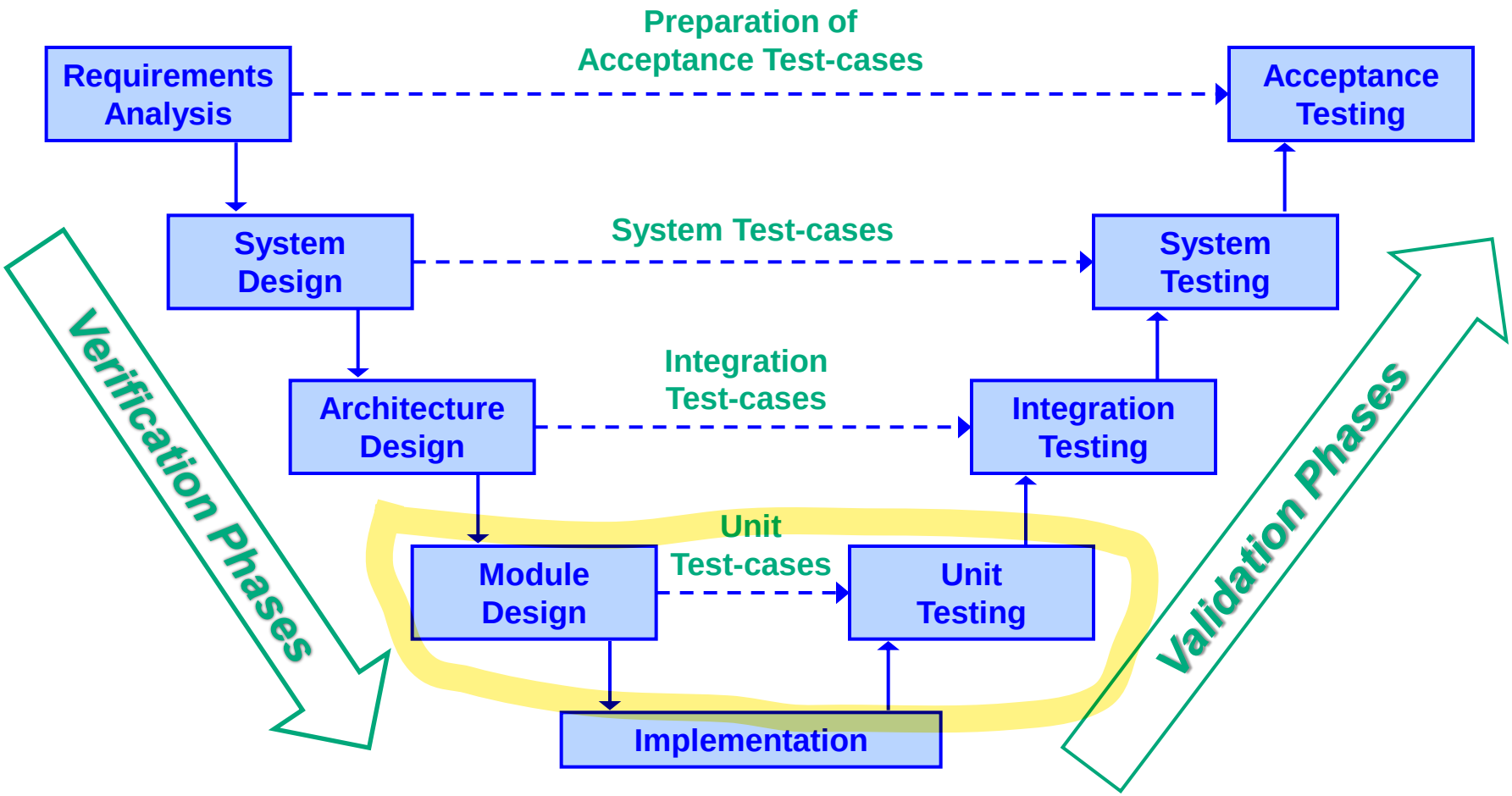
Thuật ngữ

- Phân loại defect
  - **Failure**
    - **Là hành vi bên ngoài (*external behavior*) bị sai đối với các yêu cầu hoặc đặc tả của hành vi mong muốn (*expected behavior*).**
    - **Là sự bất lực (*inability*) của thành phần hoặc hệ thống phần mềm để thực hiện các chức năng cần thiết ở trong các yêu cầu thực hiện được xác định.**
    - **Là khiếm khuyết đến với khách hàng.**
    - **Trong quá trình phát triển, *failure* thường được người kiểm thử theo dõi.**

Thuật ngữ

- **Testing**
  - Đánh giá phần mềm bằng cách theo dõi sự thực hiện của nó.
- **Test Failure**
  - Thực hiện phần mềm mà kết quả là *failure*.
- **Debugging**
  - Quá trình tìm kiếm một *fault* gây ra *failure*.

Mô hình V


Preparation of
Acceptance Test-cases
Requirements Analysis
Acceptance Testing
System Design
System Test-cases
System Testing
Verification Phases
Integration
Test-cases
Architecture Design
Integration Testing
Unit
Test-cases
Module Design
Unit Testing
Validation Phases
Implementation

Verification Phases

Requirement Analysis

- Các yêu cầu hệ thống được thu thập bằng cách phân tích nhu cầu của người sử dụng. Giai đoạn này liên quan đến việc *thiết lập những gì hệ thống phải thực hiện*.
- *Phỏng vấn khách hàng* và *lập tài liệu yêu cầu người sử dụng* (*user requirements document*): các yêu cầu về chức năng, giao diện, hiệu suất, dữ liệu, bảo mật …
- *Người phân tích nghiệp vụ* (*business analyst*) và người sử dụng phải đồng thuận tài liệu yêu cầu người sử dụng.

Verification Phases

- **Requirement Analysis**
  - ***Người thiết kế hệ thống*** **(*system designer*) sử dụng tài liệu yêu cầu người sử dụng trong giai đoạn thiết kế hệ thống.**
  - **Thiết kế các *user acceptance test-case*.**
  - ***Các phương pháp thu thập yêu cầu*****: phỏng vấn (*interview*), bảng câu hỏi (*questionnaire*), phân tích tài liệu (*document analysis*), quan sát (*observation*), *throw-away prototype*, *use-case* theo góc nhìn của người sử dụng.**
  - ***Lập User Acceptance Test Plan*****.**
  - **Chưa xác định thiết kế, xây dựng hệ thống.**

Verification Phases

System Design

- ***Kỹ sư hệ thống*** (***system engineer***) phân tích và hiểu nghiệp vụ của hệ thống bằng cách nghiên cứu tài liệu yêu cầu người sử dụng: ***tìm các khả năng và kỹ thuật để thực hiện các yêu cầu của người sử dụng***.
- Thông báo các yêu cầu không khả thi cho người sử dụng.
- Chỉnh sửa tài liệu yêu cầu người sử dụng.
- ***Lập tài liệu đặc tả phần mềm*** (***software specification document***) cho giai đoạn phát triển (***development phase***): tổ chức hệ thống (***system organization***), cấu trúc *menu*, cấu trúc dữ liệu, *business scenarios*, *sample windows* …

Verification Phases

- **System Design**
  - ***Lập tài liệu kỹ thuật*** **(*technical document*): *entity diagram, data dictionary* …**
  - **Thiết kế các *system test-case*.**
  - ***Lập System Test Plan*** **bởi nhóm nghiệp vụ khách hàng (*client's business team*).**

Verification Phases

- **Architecture Design**
  - ***Thiết kế kiến trúc phần mềm*** **(*software architecture*).**
  - ***Xác định các module*****: chức năng (*functionality*) của mỗi *module*, *interface relationships*, *dependencies*, *database tables*, *architecture diagrams*, *technology details* …**
  - ***Thiết kế các integration test-case*****.**
  - ***Lập Integration Test Plan*****.**

Verification Phases

- **Module Design**
  - Hệ thống được chia thành các *unit* hoặc *module* nhỏ hơn.
  - *Giải thích mỗi module* để người viết chương trình có thể bắt đầu viết chương trình.
  - *Functional logic* được viết bằng mã giả (*pseudocode*):
    - Database tables
    - All interface details with complete API references
    - All dependency issues
    - Error message listings
    - Complete inputs and outputs for a module.
  - *Thiết kế các unit test-case*.
  - *Lập Unit Test Plan*.

Verification Phases

Unit Test Plan

	Requirements	Typical Components	Detailed Description
1	Introduction	a. Test Strategy and Approach	
		b. Test Scope	
		c. Test Assumptions	
2	Walkthrough (Static Testing)	a. Defects Discovered and Corrected	
		b. Improvement Ideas	
		c. Structured Programming Compliance	
		d. Languages Standards	
		e. Development Documentation Standards	
3	Test-cases (Dynamic Testing)	a. Input Test Data	
		b. Initial Conditions	
		c. Expected Results	
		d. Test Log Status	
4	Environment Requirements	a. Test Strategy and Approach	
		b. Platform	
		c. Libraries	
		d. Tools	
		e. Test Procedures	
		f. Status Reporting	

Validation Phases

Unit Tesing

- Các kế hoạch kiểm tra đơn vị (*UTP – Unit Test Plan*) được lập trong giai đoạn thiết kế *module*.
- *Thực hiện các UTP* để loại bỏ các *bug* của đoạn mã lệnh hoặc *module*.
- Một *unit* là thực thể nhỏ nhất có thể tồn tại độc lập, ví dụ *program module*.
- Kiểm thử đơn vị xác minh rằng thực thể nhỏ nhất có thể hoạt động đúng khi nó được thực hiện độc lập với các *unit* khác.

Validation Phases

❖ Integration Tesing

- ***Thực hiện các Integration test-case*** **trong *Integration Test Plan*.**
- **Các *test-case* này xác minh rằng các *unit* được thực hiện độc lập có thể cùng tồn tại và giao tiếp với nhau.**
- **Kết quả kiểm tra được thông báo cho khách hàng.**

Validation Phases

System Tesing

- Kiểm thử hệ thống đảm bảo phần mềm thỏa mãn những mong muốn của người sử dụng.
- Kiểm thử hệ thống xác thực rằng *các yêu cầu chức năng và phi chức năng đã được thỏa mãn*.
- Toàn bộ phần mềm được kiểm thử về *chức năng* (*functionality*), *sự phụ thuộc lẫn nhau* (*interdependency*) và *giao tiếp* (*communication*).
- *Kiểm thử tải* (*load*) và *hiệu suất* (*performance*), *kiểm thử căng thẳng* (*stress testing*), *kiểm thử hồi quy* (*regression testing*).

Validation Phases

User Acceptance Testing (UAT)

- Người sử dụng nghiệp vụ (*business user*) lập kế hoạch kiểm tra trong giai đoạn phân tích yêu cầu.
- UAT được thực hiện trong môi trường người sử dụng (*user environment*), giống với môi trường sản xuất (*production environment*), sử dụng dữ liệu thực tế (*realistic data*).
- *UAT xác minh phần mềm thỏa mãn yêu cầu của người sử dụng* và đã sẵn sàng để sử dụng trong thời gian thực (*real time*).