ĐẠI HỌC CÔNG NGHIỆP TP.HCM
Khoa Công nghệ Thông tin

THIẾT KẾ
CÁC VẤN ĐỀ LIÊN QUAN

TS. Huỳnh Hữu Nghĩa

huynhhuunghia@iuh.edu.vn

Nội dung:

- Các thuộc tính chất lượng
- Các mối quan tâm xuyên suốt
- Giao tiếp và Tin nhắn
- Các tầng vật lý và Triển khai

Các
Thuộc tính Chất lượng

(Quality Attributes)

Tổng quan

- Các thuộc tính chất lượng là những yếu tố ảnh hưởng đến hành vi thời gian chạy, thiết kế hệ thống, và trải nghiệm người dùng, có khả năng tác động rộng khắp các lớp và tầng.

- Một số liên quan đến thiết kế hệ thống tổng thể, trong khi các thuộc tính khác dành riêng cho thời gian chạy, thời gian thiết kế, hay các vấn đề tập trung vào người dùng.

Tổng quan

- Mức độ ứng dụng sở hữu một sự kết hợp các thuộc tính chất lượng mong muốn như: tính khả dụng, hiệu suất, độ tin cậy, và bảo mật cho thấy sự thành công của thiết kế và chất lượng tổng thể của ứng dụng phần mềm.

- Khi thiết kế các ứng dụng để đáp ứng bất cứ các yêu cầu thuộc tính chất lượng, cần xem xét tác động tiềm năng trên các yêu cầu khác.

Tổng quan

- Bạn phải phân tích sự đánh đổi giữa nhiều thuộc tính chất lượng. Tầm quan trọng hay mức độ ưu tiên của từng thuộc tính chất lượng khác nhau từ hệ thống đến hệ thống;

- Ví dụ, khả năng tương tác thường sẽ ít quan trọng trong một ứng dụng bán lẻ được đóng gói sử dụng duy nhất so với một hệ thống kinh doanh (LOB).

Các thuộc tính chất lượng phổ biến

- Các thuộc tính được phân chia thành 4 loại gồm các chất lượng thiết kế, thời gian chạy, hệ thống, và người dùng.

- **Chất lượng thiết kế** (Design Qualities):

  - ✓ **Tính toàn vẹn khái niệm** (Conceptual Integrity). *Xác định tính nhất quán và sự gắn kết thiết kế tổng thể. Gồm cách mà các thành phần hay mô đun được thiết kế, cũng như các yếu tố như kiểu mã hóa và đặt tên biến.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng thiết kế** (Design Qualities):
  - ✓ **Khả năng bảo trì** (Maintainability). *Là khả năng của hệ thống trải qua những thay đổi với mức độ dễ dàng. Các thay đổi này có thể ảnh hưởng đến các thành phần, dịch vụ, đặc trưng, và giao diện khi bổ sung hay thay đổi chức năng, sửa lỗi, và đáp ứng các yêu cầu nghiệp vụ mới.*
  - ✓ **Khả năng tái sử dụng** (Reusability). *Xác định khả năng cho các thành phần và hệ thống con phù hợp để sử dụng trong các ứng dụng khác và trong các kịch bản khác. Khả năng tái sử dụng giảm thiểu sự trùng lấp các thành phần và cũng như thời gian triển khai*

Các thuộc tính chất lượng phổ biến

- **Chất lượng thời gian chạy** (Run-time Qualities):
  - ✓ **Tính khả dụng** (Availability). *xác định tỷ lệ thời gian mà hệ thống hoạt động và làm việc. Có thể được đo bằng một tỷ lệ % của tổng hệ thống ngừng hoạt động trên khoảng thời gian xác định trước. Tính khả dụng sẽ bị ảnh hưởng bởi các lỗi hệ thống, các vấn đề cơ sở hạ tầng, các cuộc tấn công độc hại và tải hệ thống.*
  - ✓ **Khả năng tương tác** (Interoperability). *là khả năng của hệ thống hay các hệ thống khác nhau hoạt động thành công bằng cách giao tiếp và trao đổi thông tin với các hệ thống bên ngoài khác được viết và chạy bởi các thành phần bên ngoài. Hệ thống có thể tương tác làm cho nó dễ dàng để trao đổi và tái sử dụng thông tin nội bộ cũng như bên ngoài.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng thời gian chạy** (Run-time Qualities):
  - ✓ **Tính quản trị** (Manageability). *xác định mức độ dễ dàng cho các quản trị viên hệ thống quản lý ứng dụng, thường thông qua các công cụ đầy đủ và hữu ích được sử dụng trong các hệ thống giám sát và để gỡ lỗi và điều chỉnh hiệu quả.*
  - ✓ **Hiệu suất** (Performance). *là một dấu hiệu cho thấy khả năng đáp ứng của hệ thống để thực hiện bất kỳ hoạt động nào trong khoản thời gian nhất định. Nó có thể được đo lường về độ trễ hay thông lượng. Độ trễ là thời gian thực hiện đến phải hồi bất kỳ sự kiện. Thông lượng là số lượng sự kiện diễn ra trong một khoản thời gian nhất định.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng thời gian chạy** (Run-time Qualities):

  - ✓ **Độ tin cậy** (Reliability). *là khả năng của hệ thống duy trình hoạt động theo thời gian. Độ tin cậy được đo bằng xác suất mà một hệ thống sẽ không bị sai xót khi thực hiện các chức năng được dự định của nó trong khoảng thời gian cụ thể.*

  - ✓ **Khả năng mở rộng** (Scalability). *là khả năng của một hệ thống có thể xử lý tăng tải mà không ảnh hưởng đến hiệu suất của hệ thống, hay khả năng dễ dàng mở rộng.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng thời gian chạy** (Run-time Qualities):
  - ✓ **Bảo mật** (Security). *là khả năng của một hệ thống để ngăn ngừa các hoạt động độc hại hoặc vô tình bên ngoài của việc sử dụng được thiết kế, và để ngăn chặn tiết lộ và mất thông tin. Một hệ thống an toàn nhằm mục tiêu để bảo vệ tài sản và ngăn chặn sử đổi thông trái phép.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng hệ thống** (System Qualities):
  - ✓ **Khả năng hỗ trợ** (Supportability). *là khả năng của một hệ thống cung cấp thông tin hữu ích để xác định và giải quyết các vấn đề khi không hoạt động chính xác.*
  - ✓ **Khả năng kiểm tra** (Testability). *là cách đo mức độ dễ dàng tạo ra tiêu chính kiểm tra hệ thống và các thành phần của nó, và thực thi các kiểm tra đó theo thứ tự xác định các tiêu chí có đúng không. Khả năng kiểm tra tốt mà cho nhiều khả năng lỗi trong hệ thống có thể được phát hiện kịp thời và hiệu quả.*

Các thuộc tính chất lượng phổ biến

- **Chất lượng người dùng** (User Qualities):
  - ✓ **Khả năng sử dụng** (Usability)**.** *xác định mức độ ứng dụng đáp ứng các yêu cầu của người dùng và người tiêu thụ bằng cách trực quan, dễ cá nhân hóa và tổng quát hóa, cung cấp quyền truy cập tốt cho những người dùng tàn tật, và mang lại trải nghiệm người dùng tổng quát tốt.*

Các
Mối Quan Tâm Xuyên Suốt

(Crosscutting Concerns)

Tổng quan

- Các hoạt động như: công nhận, ủy quyền, bộ nhớ đệm, giao tiếp, quản lý ngoại lệ, nhật ký và thiết bị, và xác thực được xem là các vấn đề quan tâm xuyên suốt vì nó ảnh hưởng đến toàn bộ ứng dụng.

Cân nhắc thiết kế chung

- Kiểm tra các chức năng được yêu cầu trong mỗi lớp, và tìm kiếm các trường hợp trong đó có thể trừu tượng chức năng đó thành các thành phần phổ biến, thậm chí có thể là các thành phần mục đích chung định cấu hình phụ thuộc vào các yêu cầu cụ thể của mỗi lớp của ứng dụng. Có khả năng các loại thành phần này sẽ được tái sử dụng trong các ứng dụng khác..

Cân nhắc thiết kế chung

- Phụ thuộc cách phân tán vật lý các thành phần và các lớp ứng dụng, có thể cài đặt các thành phần xuyên suốt trên nhiều hơn một tầng vật lý.
- Xem xét sử dụng mẫu Dependency Injection để tiêm các thể hiện của các thành phần xuyên suốt vào ứng dụng dựa trên thông tin cấu hình. Điều này cho phép thay đổi các thành phần xuyên suốt từng phần sử dụng dễ dàng, mà không yêu cầu biên dịch lại và triển khai lại ứng dụng. Các mẫu và thực hành thư viên Unity cung cấp hỗ trợ toàn diện cho mẫu Dependency Injection. Các thư viện Dependency Injection phổ biến khác bao gồm StructureMap, Ninject, và Castle Windsor.

Cân nhắc thiết kế chung

- Xem xét sử dụng thư viện các thành phần bên thứ ba có cấu hình cao và có thể giảm thời gian phát triển. Ví dụ là các mẫu và thực hành Enterprise Library, nó chứa các khối ứng dụng được thiết kế giúp triển khai bộ nhớ đệm, xử lý ngoại lệ, công nhận và ủy quyền, ghi nhật ký, xác thực, và chức năng mã hóa. Nó cũng chứa các cơ chế triển khai tiêm chính sách và một thùng chứa tiêm phụ thuộc tạo sự dễ dàng để triển khai các giải pháp một loạt các mối quan tâm xuyên suốt.

Cân nhắc thiết kế chung

- Xem xét sử dụng kỹ thuật AOP (Aspect Oriented Programming) để đưa các mối quan tâm xuyên suốt vào ứng dụng, hơn là có các cuộc gọi rõ ràng trong mã hóa. Các mẫu và thực hành cơ chế Unity và Enterprise Library Policy Injection Application Block hỗ trợ hướng tiếp cận này. Một ví dụ khác bao gồm Castle Windsor and PostSharp.

Các vấn đề thiết kế cụ thể

- ✓ **Authentication** (xác thực)
- ✓ **Authorization** (ủy quyền)
- ✓ **Caching** (bộ nhớ đệm)
- ✓ **Communication** (giao tiếp)
- ✓ **Configuration** Management (quản lý cấu hình)
- ✓ **Exception Management** (quản lý ngoại lệ)
- ✓ **Logging** and **Instrumentation** (ghi nhật ký và thiết bị)
- ✓ **State Management** (quản lý trạng thái)
- ✓ **Validation** (xác nhận)

Giao Tiếp và Nhắn Tin

(Communication and Messaging)

Tổng quan

- Một yếu tố quan trọng ảnh hưởng thiết kế một ứng dụng (đặc biệt là ứng dụng phân tán) là cách thiết kế cơ sở hạ tầng giao tiếp cho từng phần của ứng dụng.

- Các thành phần giao tiếp với nhau như: *gửi đầu vào người dùng đến lớp nghiệp vụ, và sau đó cập nhật kho dữ liệu thông qua lớp dữ liệu*.

Tổng quan

- Khi các thành phần được bố trí trên cùng tầng vật lý, thường có thể dựa vào giao tiếp trực tiếp giữa các thành phần này.

- Tuy nhiên, nếu triển khai các thành phần và các lớp trên các máy server và client riêng biệt vật lý (có khả năng trong hầu hết các kịch bản) phải xem cách các thành phần trong các lớp này sẽ giao tiếp với nhau một cách hiệu quả và đáng tin cậy.

Tổng quan

- Nói chung, phải chọn giữa giao tiếp trực tiếp (như phương thức gọi giữa các thành phần) và giao tiếp dựa trên tin nhắn.

- Có nhiều lợi thế khi sử dụng giao tiếp dựa trên tin nhắn, như khả năng tách rời các thành phần từ những cái khác.

Tổng quan

- Các thành phần tách rời không chỉ cải thiện khả năng bảo trì mà còn cung cấp tính linh hoạt tạo thuận lợi để thay đổi chiến lược phát triển trong tương lai.

- Tuy nhiên, giao tiếp dựa trên tin nhắn đặt ra các vấn đề phải xem xét như hiệu suất, độ tin cậy và bảo mật.

Hướng dẫn thiết kế chung

- Khi thiết kế một chiến lược giao tiếp cho ứng dụng, xem xét tác động hiệu suất của giao tiếp tiếp giữa các lớp, cũng như giữa các tầng.

- Bởi vì mỗi giao tiếp thông qua một ranh giới logic hay vật lý làm tăng chi phí xử lý, thiết kế để giao tiếp hiệu quả bằng cách giảm các chuyến đi khứ hồi và giảm thiểu khối lượng dữ liệu gửi qua mạng.

Hướng dẫn thiết kế chung

- Xem xét các chiến lược giao tiếp khi xuyên qua các ranh giới. Hiểu từng ranh giới và cách chúng ảnh hưởng đến hiệu suất giao tiếp. *Ví dụ, quá trình xử lý máy tính, máy móc, và chuyển đổi mã được quản lý đến không quản lý thể hiện trên tất cả các ranh giới có thể vượt qua khi giao tiếp với các thành phần của ứng dụng hay các dịch vụ và ứng dụng bên ngoài.*

- Xem xét sử dụng giao tiếp dựa trên tin nhắn khi vượt qua các ranh giới xử lý. Sử dụng WCF với giao thức TCP hay các đường ống xác định tên cho hiệu quả cao nhất.

Hướng dẫn thiết kế chung

- Xem xét sử dụng giao tiếp dựa trên tin nhắn khi xuyên qua các ranh giới vật lý. Sử dụng WCF để giao tiếp với máy từ xa qua các ranh giới vật lý. Sử dụng MSSQ cho chỉ một lần gửi tin nhắn đáng tin cậy.

- Tối đa hóa hiệu suất và khả năng đáp ứng khi truy cập các lớp từ xa. Khi giao tiếp với các lớp từ xa, hãy giảm các yêu cầu giao tiếp bằng cách sử dụng các phương thức giao tiếp dựa trên tin nhắn chi tiết thô và sử dụng giao tiếp bất đồng bộ nếu có thể để tránh bị chặn hay đóng băng UI.

Hướng dẫn thiết kế chung

- Xem xét khả năng tuần tự hóa những định dạng dữ liệu được truyền qua các ranh giới. Nếu yêu cầu khả năng tương tác với các hệ thống khác, xem xét tuần tự hóa XML. Nhớ rằng tuần tự hóa XML áp đặt chi phí gia tăng. Nếu hiệu suất là quan trọng, xem xét tuần tự hóa nhị phân vì nó nhanh hơn và dữ liệu được tuần tự hóa kết quả thì nhỏ hơn tương đương XML.

Hướng dẫn thiết kế chung

- Đảm bảo rằng việc bảo vệ các tin nhắn và dữ liệu nhạy cảm trong khi giao tiếp. Xem xét sử dụng mã hóa, các chứng nhận kỹ thuật số, và các tính năng bảo mật kênh.

- Triển khai các cơ chế để thực thi tính tự do và giao hoán. Đảm bảo rằng mã hóa ứng dụng có thể phát hiện và quản lý các tin nhắn đến nhiều lần (tính không thay đổi) và nhiều tin nhắn không theo thứ tự (giao hoán).

Tầng Vật Lý và Triển Khai

(Physical Tiers and Deployment)

Tổng quan

- Ứng dụng được triển khai trong môi trường vật lý, trong đó những giới hạn cơ sở hạ tầng có thể phủ nhận một số quyết định kiến trúc.

- Do đó, phải xem xét kịch bản phát triển được đề xuất và cơ sở hạ tầng là một phần của quá trình thiết kế ứng dụng.

Tổng quan

- Khi triển khai ứng dụng cần chọn các kiểu phân tán và không phân tán; các cách để mở rộng ứng dụng; và hướng dẫn và các mẫu cho các vấn đề hiệu suất, độ tin cậy và bảo mật.

- Xem xét các kịch bản phát triển có thể có cho ứng dụng như một phần của quá trình thiết kế, để ngăn chặn tình huống trong đó ứng dụng không thể được phát triển thành công, hay không thực hiện theo yêu cầu thiết kế của nó do các giới hạn cơ sở hạ tầng kỹ thuật.

Tổng quan

- Chiến lược triển khai yêu cầu cần bằng thiết kế.

- Khi chọn chiến lược triển khai phải:
  - ✓ Hiểu môi trường vật lý mục tiêu triển khai.
  - ✓ Hiểu các ràng buộc kiến trúc và thiết kế dựa trên môi trường triển khai.
  - ✓ Hiểu các tác động bảo mật và hiệu suất của môi trường triển khai.

Các vấn đề triển khai

- Triển khai phân tán hay không phân tán
- Các mẫu triển khai phân tán
- Các mẫu hiệu suất
- Các mẫu độ tin cậy
- Các mẫu bảo mật
- Xem xét an ninh hạ tầng mạng
- Xem xét khả năng quản lý


Q&A